ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ
ATHENS UNIVERSITY OF ECONOMICS AND BUSINESS

ΣΧΟΛΗ ΕΠΙΣΤΗΜΩΝ & ΤΕΧΝΟΛΟΓΙΑΣ ΤΗΣ ΠΛΗΡΟΦΟΡΙΑΣ
SCHOOL OF INFORMATION SCIENCES & TECHNOLOGY

ΤΜΗΜΑ ΣΤΑΤΙΣΤΙΚΗΣ
DEPARTMENT OF STATISTICS

ΣΧΟΛΗ ΕΠΙΣΤΗΜΩΝ & ΤΕΧΝΟΛΟΓΙΑΣ ΤΗΣ ΠΛΗΡΟΦΟΡΙΑΣ

ΤΜΗΜΑ ΣΤΑΤΙΣΤΙΚΗΣ

Operator Learning for Integral Operators

Leonte Gabriel Robert - Λεόντε Γκ. Ρόμπερτ

Διπλωματική Εργασία

που υποβλήθηκε στο Τμήμα Στατιστικής
του Οικονομικού Πανεπιστημίου Αθηνών στο
πλαίσιο του Προπτυχιακού Προγράμματος Σπουδών

Αθήνα

Σεπτέμβριος 2026

Acknowledgments

First, I thank God for giving me the strength and wisdom to complete this journey.

I am deeply grateful to my supervisor, Prof. Yannacopoulos Athanasios, for their invaluable guidance, encouragement, and continuous support. Their insight and expertise shed light upon a field of machine learning and computational mathematics I would not have encountered in other circumstances.

Finally, thank you to my family and friends. Your constant encouragement and belief in me kept me going when things got tough|I could not have done this without you.

Περίληψη

Τα αντίστροφα προβλήματα που περιγράφονται μέσω ολοκληρωτικών εξισώσεων Fredholm παρουσιάζουν σημαντικές δυσκολίες επίλυσης, καθώς μικρά σφάλματα στις μετρήσεις μπορούν να προκαλέσουν μεγάλες αστάθειες στη λύση. Παρότι η βαθιά μάθηση προσφέρει μια προσέγγιση βασισμένη στα δεδομένα, η εκπαίδευση βαθιών δικτύων τελεστών απαιτεί συχνά πολλές ώρες υπολογισμών μέσω οπισθοδιάδοσης και μπορεί να οδηγήσει στην εμφάνιση μη φυσικών τεχνουργημάτων.

Στην παρούσα εργασία παρακάμπτουμε τη διαδικασία βαθιάς εκπαίδευσης αξιοποιώντας τα Extreme Learning Machines, ELMs). Η αντικατάσταση των βαθιών νευρωνικών δικτύων από σταθερές τυχαίες βάσεις μετατρέπει την προσέγγιση του συνεχούς μετασχηματισμού σε ένα απλό πρόβλημα γραμμικών ελαχίστων τετραγώνων. Στο πλαίσιο αυτό, εισάγονται δύο μοντέλα: το LEOR, το οποίο βασίζεται στην απεικόνιση λανθάνοντος υποχώρου, και η Άμεση Προσέγγιση του Πυρήνα (Direct Kernel Approximation), η οποία χρησιμοποιείται για την προσεγγιστική μοντελοποίηση δισδιάστατων συναρτήσεων Green.

Για τη διατήρηση της φυσικής συνέπειας του ανακατασκευασμένου πυρήνα, ενσωματώνονται στη συνάρτηση απώλειας όροι κανονικοποίησης που επιβάλλουν χωρική εξομάλυνση, συμμετρία και κατάλληλη συμπεριφορά στα όρια του πεδίου. Τα αποτελέσματα των υπολογιστικών πειραμάτων δείχνουν ότι η προτεινόμενη προσέγγιση απαιτεί σημαντικά μικρότερο υπολογιστικό χρόνο σε σύγκριση με τα συμβατικά δίκτυα DeepONet, ενώ παράλληλα αποτυπώνει με μεγαλύτερη ακρίβεια τις φυσικές ιδιότητες και τη φασματική φθορά του υπό μελέτη τελεστή.

Abstract

Inverse problems governed by Fredholm integral equations are hard to solve because small measurement errors can cause huge instabilities. While deep learning offers a data-driven way forward, training deep operator networks requires hours of backpropagation and often yields non-physical artifacts.

In this thesis, we bypass deep training altogether using Extreme Learning Machines (ELMs). Replacing deep networks with fixed random bases turns the continuous mapping into a basic linear least-squares problem. We introduce two models under this framework: LEOR for latent subspace mapping and Direct Kernel Approximation for modeling 2D Green's functions. To keep the reconstructed kernel physically plausible, we add spatial smoothing, symmetry, and boundary regularizers to the loss function. Benchmarks show that this approach runs significantly faster than standard DeepONets while better capturing the true physical properties and spectral decay of the target operator.

The Lord will rescue me from every evil attack and will bring me safely to his heavenly kingdom. To him be glory for ever and ever. Amen

2 Timothy 4:18

Contents

1 Introduction **1**

1.1 Motivation: The Ubiquity of Inverse Problems 1

2 Preliminaries **7**

2.1 Elements of Functional Analysis 7

2.2 Integral Operators and Fredholm Equations 10

2.3 Spectral Theory of Compact Operators 12

3 Operator Learning via Randomized Architectures **15**

3.1 The Operator Learning Framework 15

3.2 Galerkin Projection and Randomized Bases 17

3.3 Advanced Neural Operators 20

3.4 Formulation of Investigated models 22

3.5 The Ill-Posedness of Kernel Learning and Regularization 24

4 Proposed Operator Learning Models and Applications **27**

4.1 Objective 27

4.2 Experimental Setup and Methodology 27

4.3 Baseline and Latent Subspace Mappings 30

4.4 Direct Kernel Approximation 31

4.5 Deep Operator Networks (DeepONets) 34

5 Conclusions and Future Research **37**

5.1 Conclusions 37

5.2 Future Research 38

6 Appendix **39**

6.1 Results from Naive ELM 39

6.2 Results from LEOR 42

6.3 Results from Kernel Reconstruction using Bi-ELM 42
6.4 Results from DeepONets 61

Bibliography **65**

List of Figures

1.1 Schematic representation of Forward and Inverse Problems. The dashed line indicates the inherently unstable nature of the inverse mapping. 2

1.2 A graph showing the smoothing of an oscillatory function under the kernel $k(x, y) = \exp(-(x - y)^2/(2\sigma^2))$ 3

1.3 Comparison of classical Machine Learning, which maps strictly between fixed-dimensional vectors, and Operator Learning, which learns the continuous transformation between infinite-dimensional function spaces. 4

6.1 Bootstrapp Boxplots for NaiveELM evaluated at Exponential Decay kernel 40

6.2 Bootstrapp Boxplots for NaiveELM evaluated at Minimum $\min()$ kernel 40

6.3 Bootstrapp Boxplots for NaiveELM evaluated at Polynomial kernel 41

6.4 Bootstrapp Boxplots for NaiveELM evaluated at 1D Poisson's kernel 41

6.5 Comparison of estimated kernel functions $\hat{K}(x, y)$ across different penalization techniques against the true kernel $K(x, y)$. Panels display two-dimensional heatmaps over the spatial domain $[0, 1]^2$. Color intensity represents kernel value magnitude. 45

6.6 Estimated eigenvalue spectra $\hat{\lambda}_i$ across all estimation techniques compared against the true eigenvalue spectrum λ_i. The horizontal axis displays the eigenvalue rank $i = 1, \ldots, 15$, while the vertical axis shows the eigenvalue magnitude on a logarithmic scale. 46

6.7 Spatial profiles of the first five estimated eigenfunctions $\hat{\phi}_1, \ldots, \hat{\phi}_5$ compared with the true eigenfunctions of the integral operator. 48

6.8 Comparison of estimated kernel functions $\hat{K}(x, y)$ across different penalization techniques against the true kernel $K(x, y)$. Panels display two-dimensional heatmaps over the spatial domain $[0, 1]^2$. Color intensity represents kernel value magnitude. 52

6.9 Estimated eigenvalue spectra $\hat{\lambda}_i$ across all estimation techniques compared against the true eigenvalue spectrum λ_i. The horizontal axis displays the eigenvalue rank $i = 1, \ldots, 15$, while the vertical axis shows the eigenvalue magnitude on a logarithmic scale. 53
6.10 Spatial profiles of the first five estimated eigenfunctions $\hat{\phi}_1, \ldots, \hat{\phi}_5$ compared with the true eigenfunctions of the integral operator. 54
6.11 Comparison of estimated kernel functions $\hat{K}(x, y)$ across different penalization techniques against the true kernel $K(x, y)$. Panels display two-dimensional heatmaps over the spatial domain $[0, 1]^2$. Color intensity represents kernel value magnitude. 57
6.12 Estimated eigenvalue spectra $\hat{\lambda}_i$ across all estimation techniques compared against the true eigenvalue spectrum λ_i. The horizontal axis displays the eigenvalue rank $i = 1, \ldots, 15$, while the vertical axis shows the eigenvalue magnitude on a logarithmic scale. 58
6.13 Spatial profiles of the first five estimated eigenfunctions $\hat{\phi}_1, \ldots, \hat{\phi}_5$ compared with the true eigenfunctions of the integral operator. 60
6.14 Visual output and kernel reconstruction of the **Model**: Ridge on Trunk, Ridge on Branch and Tanh activation functions 61
6.15 Visual output and kernel reconstruction of the **Model**: Ridge on Trunk, Lasso on Branch and Tanh activation functions 62
6.16 Visual output and kernel reconstruction of the **Model**: Ridge on Trunk, Lasso on Branch and Sine activation functions 62
6.17 Visual output and kernel reconstruction of the **Model**: Ridge on Trunk, Lasso on Branch, Sine activasio functions, Soft Boundary conditions 63

List of Tables

6.1 Performance Metrics for Naive ELM 39

6.2 Performance Metrics for Model B (Relative Errors in Percentages) 42

6.3 Out-of-bag bootstrap measures for kernel estimation using Ridge penalization. . 42

6.4 Out-of-bag bootstrap measures for kernel estimation using Lasso penalization. . 43

6.5 Out-of-bag bootstrap measures for kernel estimation using Elastic Net penalization. 43

6.6 Out-of-bag bootstrap measures for kernel estimation using Ridge penalization with Soft Symmetry. 43

6.7 Out-of-bag bootstrap measures for kernel estimation using Lasso penalization with Soft Symmetry. 43

6.8 Out-of-bag bootstrap measures for kernel estimation using Ridge penalization with Laplacian regularization. 44

6.9 Out-of-bag bootstrap measures for kernel estimation using Ridge penalization with Soft Symmetry and Soft Physics. 44

6.10 Out-of-bag bootstrap measures for kernel estimation using Ridge penalization with Soft Symmetry and Soft Boundary Conditions. 44

6.11 Relative error of estimated eigenvalues. Values with relative error $> 50\%$ (0.50) are dimmed in light gray to highlight valid estimates. 46

6.12 Relative error of estimated eigenfunctions compared to true eigenfunctions. Values with relative error $> 50\%$ (0.50) are dimmed in light gray to highlight accurate predictions. 47

6.13 Out-of-bag bootstrap measures for kernel estimation using Ridge penalization. . 49

6.14 Out-of-bag bootstrap measures for kernel estimation using Lasso penalization. . 49

6.15 Out-of-bag bootstrap measures for kernel estimation using Elastic Net penalization. 49

6.16 Out-of-bag bootstrap measures for kernel estimation using Ridge penalization with Soft Symmetry. 49

6.17 Out-of-bag bootstrap measures for kernel estimation using Lasso penalization with Soft Symmetry. 50

6.18 Out-of-bag bootstrap measures for kernel estimation using Ridge penalization with Soft Symmetry and Laplacian regularization. 50
6.19 Out-of-bag bootstrap measures for kernel estimation using Ridge penalization with Soft Symmetry and Soft Physics. 50
6.20 Out-of-bag bootstrap measures for kernel estimation using Ridge penalization with Soft Symmetry and Boundary Conditions. 51
6.21 Relative error of the top estimated eigenvalues compared to true eigenvalues. Values with relative error $> 50\%$ (0.50) are dimmed in light gray to highlight accurate predictions. 51
6.22 Relative error of the top estimated eigenfunctions compared to true eigenfunctions. Values with relative error $> 50\%$ (0.50) are dimmed in light gray to highlight accurate predictions. 53
6.23 Out-of-bag bootstrap measures for kernel estimation using Ridge penalization. . 55
6.24 Out-of-bag bootstrap measures for kernel estimation using Lasso penalization. . 55
6.25 Out-of-bag bootstrap measures for kernel estimation using Ridge penalization with Soft Symmetry. 55
6.26 Out-of-bag bootstrap measures for kernel estimation using Ridge penalization with Soft Symmetry and Laplacian Regularization. 56
6.27 Out-of-bag bootstrap measures for kernel estimation using Ridge penalization with Soft Symmetry and Soft Physics. 56
6.28 Out-of-bag bootstrap measures for kernel estimation using Ridge penalization with Soft Symmetry and Boundary Conditions. 56
6.29 Relative error of the top 10 estimated eigenvalues compared to true eigenvalues. Values with relative error $> 50\%$ (0.50) are dimmed in light gray to highlight accurate predictions. 58
6.30 Relative error of the top 15 estimated eigenfunctions compared to true eigenfunctions. Values with relative error $> 50\%$ (0.50) are dimmed in light gray to highlight accurate predictions. 59
6.31 Out-of-sample prediction and kernel estimation metrics across different system determination regimes (Mean $\pm$ 95% CI). **Model**: Ridge on Trunk, Ridge on Branch and Tanh activation functions 61
6.32 Out-of-sample prediction and kernel estimation metrics across different system determination regimes (Mean $\pm$ 95% CI). **Model**: Ridge on Trunk, Lasso Branch and Tanh activation functions. 61

6.33 Out-of-sample prediction and kernel estimation metrics across different system determination regimes (Mean ± 95% CI). **Model**: Ridge on Trunk, Lasso on Branch and Sine activation functions. 62
6.34 Out-of-sample prediction and kernel estimation metrics across different system determination regimes (Mean ± 95% CI). **Model**: Ridge on Trunk, Lasso on Branch, Sine activasio functions, Soft Boundary conditions 63

Chapter 1

Introduction

1.1 Motivation: The Ubiquity of Inverse Problems

Mathematical modeling is at the core of all physical sciences, as it involves the establishment of well-defined relationships between the cause behind certain phenomena and its effects. If the governing laws of physics and the initial state of the process are fully understood, forecasting future measurements is considered a *forward problem*. Formally, this requires applying a forward operator K to some input state $v(y)$, resulting in an output effect $u(x)$.

In practice, however, there are several important situations where the roles are reversed. Instead of observing the system's state and predicting its evolution, we take measurements and try to determine what caused them. This is known as an *inverse problem*. Solving an inverse problem requires using an inverse operator K^{-1} applied to observations $u(x)$, resulting in identifying some unknown cause $v(y)$.

Inverse problems are widespread in modern science. Here are two examples:

- **Medical Imaging:** In Computed Tomography (CT), the forward problem describes the scattering and attenuation of X-rays as they pass through human tissue. The inverse problem—which is the actual diagnostic task—requires reconstructing the internal density map of the tissues from external, lower-dimensional sensor readings.

- **Thermodynamics:** Calculating the future temperature distribution of a material given a known heat source is a standard forward diffusion problem. Conversely, determining the precise location, duration, and intensity of an initial heat source based solely on a snapshot of the current temperature distribution constitutes a complex inverse problem.

While forward problems mapping $v \mapsto u$ are typically well-posed and computationally stable, the corresponding inverse problems are notoriously elusive. Physical processes such as diffu-

sion or scattering inherently consume energy and smooth out local information. Consequently, attempting to mathematically reverse this process via K^{-1} is highly unstable.

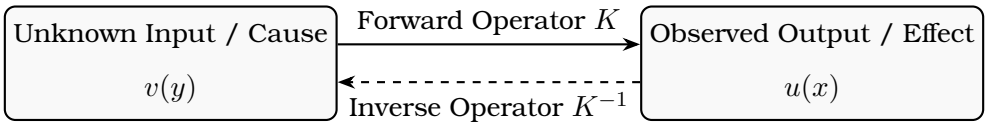

Figure 1.1: Schematic representation of Forward and Inverse Problems. The dashed line indicates the inherently unstable nature of the inverse mapping.

1.1.1 The Mathematical Problem: Integral Operators

In order to analyze such systems in depth, they have to be formulated mathematically. For many types of physical problems, such as signal processing and heat loss, the forward mapping process is defined by the 1D Fredholm integral equation of the first kind.

Let $\Omega = [0, 1]$ be our physical domain. We define the forward operator $K : \mathcal{V} \to \mathcal{U}$ mapping an input function $v(y)$ to an output function $u(x)$ via the integral transformation:

$$u(x) = \int_0^1 k(x, y)v(y)dy \tag{1.1}$$

or, in simplified operator notation, $Kv = u$. The bivariate function $k(x, y)$ is known as the *kernel* of the operator. The physical properties of the system—such as the rate of diffusion or the geometry of sensor scattering—are entirely encoded within this kernel.

While computing the forward integral (finding u given v) is computationally straightforward, the inverse problem (finding v given u) introduces a severe mathematical bottleneck: the inherent ill-posedness of the operator.

To understand this intuitively, one can view the integral operator K as a "filter". Integration is fundamentally an averaging, smoothing process. When the kernel $k(x, y)$ is a continuous, smooth function, the operator aggressively dampens high-frequency oscillations present in the input $v(y)$. Consequently, the resulting output $u(x)$ emerges perfectly smooth, completely stripping away the high-frequency structural details of the original cause.

This smoothing behavior is mathematically formalized by the compactness of the operator. Because the forward mapping destroys high-frequency information, attempting to reverse the process via the inverse operator K^{-1} has catastrophic consequences. In any real-world scenario, the observed data $u(x)$ will contain microscopic, high-frequency measurement noise. The inverse operator cannot distinguish between true high-frequency physical signals and sensor noise; thus, it amplifies this noise exponentially, causing the reconstructed solution to

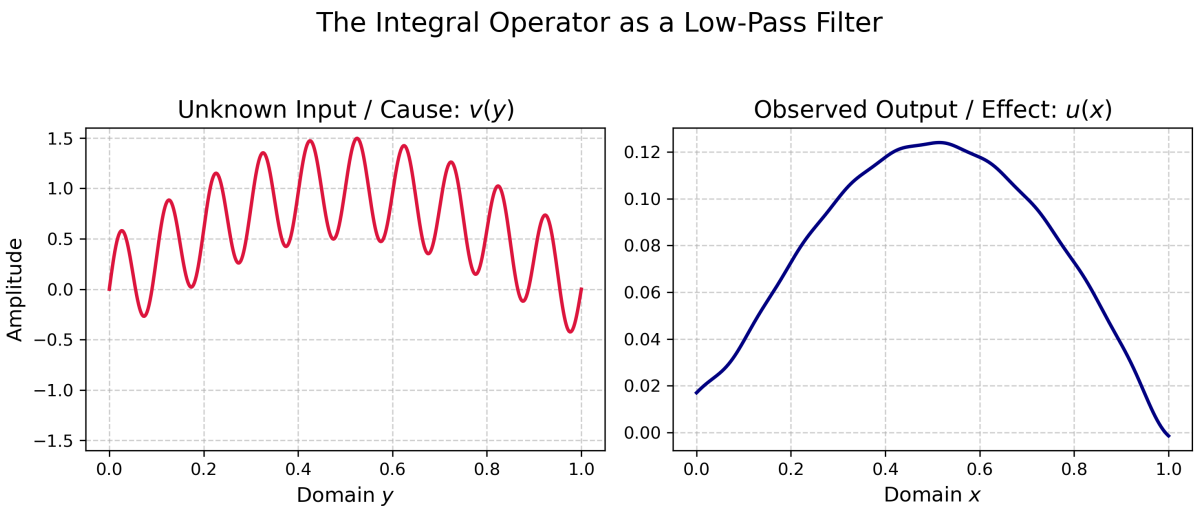

Figure 1.2: A graph showing the smoothing of an oscillatory function under the kernel $k(x, y) = \exp(-(x-y)^2/(2\sigma^2))$

mathematically explode.

This loss of information is visually demonstrated in Figure 1.2. The highly oscillatory input function is smoothed into a gentle curve, proving that standard numerical inversion is an impossible task without advanced regularization or data-driven learning frameworks.

1.1.2 The Paradigm Shift: Operator Learning and Randomized Architectures

While traditional Machine Learning (ML) has achieved remarkable success in various domains, its application to infinite-dimensional physical systems reveals a fundamental structural limitation. Standard neural networks—such as Multi-Layer Perceptrons (MLPs) or Convolutional Neural Networks (CNNs)—are designed to learn mappings between finite-dimensional Euclidean spaces, denoted as $f_\theta : \mathbb{R}^n \to \mathbb{R}^m$.

In the context of evaluating functions, this means the network is strictly tied to a fixed spatial grid. If an MLP is trained to map an input function evaluated at 50 sensor points to an output function evaluated at 50 points, the network architecture is permanently fixed to those dimensions. If we attempt to evaluate the trained network on a finer grid of 100 sensor points, the model inherently crashes, requiring a complete retraining of the parameters.

The Operator Learning Framework

To overcome this grid-dependency, the field has experienced a paradigm shift toward *Operator Learning*. Instead of mapping finite arrays of discrete data, operator networks are designed to learn the mapping between continuous, infinite-dimensional function spaces, $\mathcal{F}_\theta : \mathcal{V} \to \mathcal{U}$.

Because the neural operator learns the underlying functional transformation rather than

a specific point-to-point interpolation, it is inherently *mesh-free*. Once trained, an operator network can evaluate the output function at any arbitrary continuous coordinate, regardless of the resolution of the training data. This continuous-to-continuous mapping is visualized in Figure 1.3.

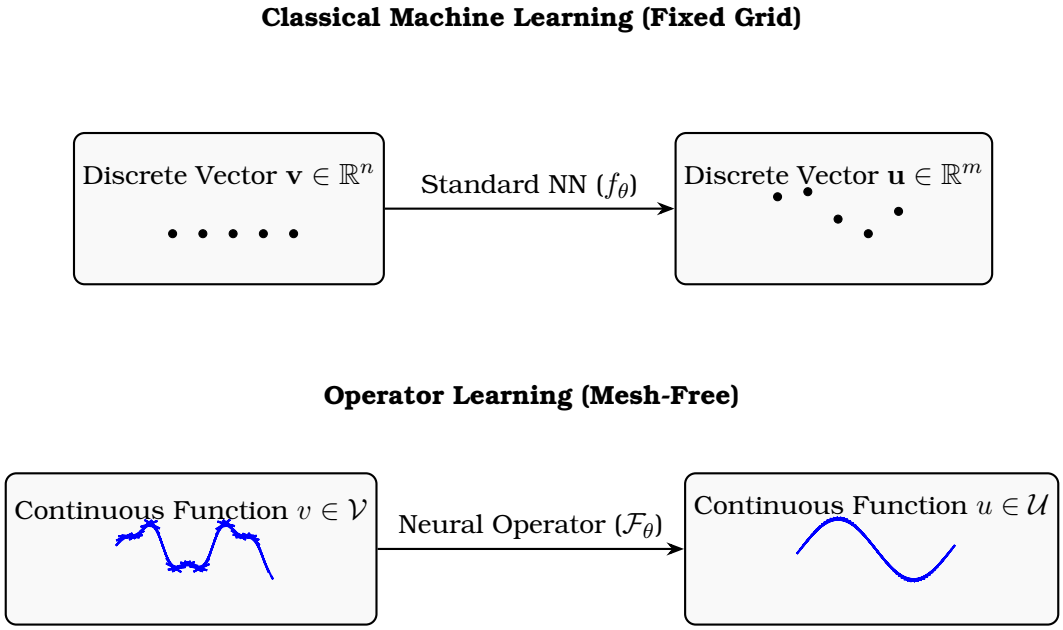

Figure 1.3: Comparison of classical Machine Learning, which maps strictly between fixed-dimensional vectors, and Operator Learning, which learns the continuous transformation between infinite-dimensional function spaces.

The Computational Barrier and Randomized Architectures

While Operator Learning provides the theoretical framework required to solve Inverse Problems, it introduces a severe computational barrier. State-of-the-art architectures, such as Deep Operator Networks (DeepONets), achieve their expressivity by training dual, deep neural networks (the Branch and the Trunk) simultaneously. Optimizing these dual networks requires navigating a highly non-convex loss landscape via backpropagation. For infinite-dimensional problems requiring dense sensor sampling, this process is computationally massive, prone to vanishing gradients, and demands extensive training times.

To shatter this computational bottleneck, this thesis prezents the integration of *Randomized Architectures*, specifically utilizing the paradigm of Extreme Learning Machines (ELMs).

Rather than iteratively training deep feature extraction layers, an ELM generates a dense, randomized functional basis by drawing its hidden layer weights from a fixed probability distribution and permanently freezing them. By replacing the deep networks in operator archi-

tectures with these randomized feature expansions, the highly non-linear, infinite-dimensional optimization problem collapses gracefully into a finite-dimensional, linear least-squares problem. This allows the neural operator to be trained without a single epoch of backpropagation, preserving theoretical accuracy while accelerating computational training times by several orders of magnitude.

1.1.3 Thesis Objectives and Contributions

The primary objective of this thesis is to investigate, develop, and evaluate machine learning methodologies for the robust approximation of compact integral operators, with a specific focus on solving ill-posed Inverse Problems. By leveraging the computational efficiency of randomized architectures, this work aims to bypass the severe computational bottlenecks associated with standard deep operator networks.

Specifically, the core contributions of this thesis are as follows:

- **Latent Space Operator Regression:** We formulate and test the Latent Extreme Operator Regression (LEOR) model, demonstrating how the complex task of learning an infinite-dimensional mapping can be reduced to a linear transformation between randomized functional bases.

- **Physics-Informed Kernel Approximation:** We develop a Direct Kernel Approximation architecture. By posing structural priors—such as Tikhonov regularization for boundedness, Laplace smoothing for spatial coherence, symmetry constraints, and boundary conditions—we explicitly collapse the non-physical null space and recover the true underlying Green's functions.

- **Comparison:** We compare these models across different sets of problems to assess their predictive ability, then compare the spectral approximation under the kernel approximation.

1.1.4 Thesis Outline

The remainder of this thesis is structured as such: Chapeter 2 establishes the rigorous functional analysis foundation required for operator learning. It defines the relevant Hilbert spaces, details the spectral theory of compact operators, and mathematically proves the inherent ill-posedness of Inverse Problems, thus justifying the necessity of data-driven regularization, Chapter 3 transitions from pure mathematics to machine learning. It formulates the continuous-to-continuous empirical risk minimization framework, introduces the Galerkin projection mapping

via Extreme Learning Machines (ELMs) and state-of-the-art operator learning models, and provides the rigorous mathematical derivations for the specific models and composite loss functions investigated in this study, Chapter 4 presents the core empirical findings. It establishes the benchmark Fredholm equations, demonstrates the failure of grid-based baselines, and evaluates the Latent Extreme Operator Regression (LEOR) model. It then conducts an ablation study on Direct Kernel Approximation to show how structural and physical regularizers isolate the true kernel, and concludes with a comparative benchmark against Deep Operator Networks (DeepONets). Chapter 5 summarizes the key findings of the research, discusses some of the theoretical and practical limitations of the proposed methods, and outlines potential ideas for further research.

Chapter 2

Preliminaries

The study of operator learning, particularly for integral operators, requires an understanding of the structure of function spaces. This comes up as natural extension of the union of linear algebra and real analysis, known as *Functional Analysis*. In this section, we will provide some necessary information concerning for the most part the foundation of this thesis and order them by abstractness, from most abstract to most special. We will introduce the fundamental spaces and concepts that will be used throughout the thesis to justify the models created. For further reading, the reader is advised to consult [1, 2, 3] for introductory knowledge, [4] for the topic of Compact Operators, and [5] for an advanced, in-depth analysis.

2.1 Elements of Functional Analysis

The concepts of distance, convergence, and geometry in infinite-dimensional spaces are foundational for defining both the integral equations and the objective functions used to train operator networks.

2.1.1 Normed and Banach Spaces

To train a machine learning model to approximate an unknown operator, we must first establish a mathematical definition of "error" or "distance" between the true output function and our model's prediction.

Let X be a linear vector space over a field $\mathbb{K}$ (typically $\mathbb{R}$ or $\mathbb{C}$).

A norm is a function $\|\cdot\| : X \to \mathbb{R}$ that satisfies the following properties for all $x, y \in X$ and scalars $\alpha \in \mathbb{K}$:

- Positivity: $\|x\| \geq 0$, and $\|x\| = 0 \iff x = 0$
- Absolute Homogeneity: $\|\alpha x\| = |\alpha| \|x\|$

- Triangle Inequality: $\|x + y\| \leq \|x\| + \|y\|$

A vector space equipped with a norm is called a *normed* space. Every norm naturally induces a metric $d(x, y) = \|x - y\|$, which will serves as a theoretical loss function under different types of norms.

When training neural networks, we deal with sequences of functions that converge toward a target. We require that the space is complete - meaning that every Cauchy sequence of functions converges to a limit that is also contained within the space. A *complete normed vector* space is called a **Banach** space.

Beyond completeness, machine learning theoretically relies on the concept of *density*. A subset $\mathcal{M}$ is said to be *dense* in a Banach space X if its closure is the entire space ($\overline{\mathcal{M}} = X$). Practically, this means that for any target function $f \in X$ and any $\epsilon > 0$, there exists an element $g \in \mathcal{M}$ such that $\|f - g\|_X < \epsilon$.

Note on Machine Learning : The Universal Approximation Theorem essentially establishes that the hypothesis space generated by specific neural network architectures (like ELMs) forms a dense subset within target Banach spaces, such as $C(\Omega)$ or $L^p(\Omega)$.

2.1.2 Inner Product and Hilbert Spaces

While Banach spaces provide a framework for convergence, they lack geometric structure. Analyzing the spectral properties of operators and perform orthogonal projections requires the notion of angles and orthogonality.

An inner product space is a linear space $\mathcal{H}$ equipped with an inner product $\langle \cdot, \cdot \rangle : \mathcal{H} \times \mathcal{H} \to \mathbb{C}$ (or $\mathbb{R}$), satisfying symmetry, linearity in the first argument, and positive-definiteness.The inner product naturally induces a norm on $\mathcal{H}$ defined by:

$$\|x\| = \sqrt{\langle x, x \rangle}$$

A **Hilbert** space is an *inner product* space that is complete with respect to the norm induced by its inner product. Thus, every Hilbert space is a Banach space, but with the added power of geometric operations like orthogonal projections.

Hilbert spaces grant us two foundational mathematical tools:

- The Cauchy-Schwarz Inequality: $|\langle x, y \rangle| \leq \|x\| \|y\|$. This is instrumental in proving the boundedness of integral operators.

- Orthogonal Projections and Bases: In a separable Hilbert space, any element $x \in \mathcal{H}$ can be represented by a Fourier-type series expansion over an orthonormal basis $\{e_n\}_{n=1}^{\infty}$:

$$x = \sum_{n=1}^{\infty} \langle x, e_n \rangle e_n$$

By truncating this infinite series, we can project infinite-dimensional functions into finite-dimensional latent representations.

The Space $L^2(\Omega)$:For the scope of this thesis, the most important Hilbert space is $L^2(\Omega)$, the space of all square-integrable functions on Ω. This is the natural setting for 1D Fredholm integral equations. For two functions $f, g \in L^2(\Omega)$, the inner product is defined as:

$$\langle f, g \rangle_{L^2} = \int_{\Omega} f(x)\overline{g(x)}dx$$

with the corresponding induced norm:

$$\|f\|_{L^2} = \left(\int_{\Omega} |f(x)|^2 dx \right)^{1/2}$$

Because our empirical observations and sensor data map discretely to this continuous domain, the L^2 norm will serve as the primary metric for evaluating the predictive capability and spectral approximations

2.1.3 Linear Operators and Boundedness

With our infinite-dimensional spaces defined, we can formalize the mappings between them.

A mapping $T : X \to Y$ between two normed spaces is a *linear* operator if it preserves vector addition and scalar multiplication:

$$T(\alpha x + \beta y) = \alpha T(x) + \beta T(y)$$

An operator T is *bounded* if there exists a real constant $C \geq 0$ such that for all $x \in X$:

$$\|Tx\|_Y \leq C\|x\|_X$$

Boundedness is mathematically equivalent to *continuity* for linear operators.

The infimum such constant C is called the operator norm, defined as:

$$\|T\| = \sup_{x \neq 0} \frac{\|Tx\|_Y}{\|x\|_X}$$

Finally, in a Hilbert space $\mathcal{H}$, for every bounded linear operator T, there exists a unique adjoint operator $T^* : \mathcal{H} \to \mathcal{H}$ such that for all $x, y \in \mathcal{H}$:

$$\langle Tx, y \rangle = \langle x, T^* y \rangle$$

An operator is called self-adjoint if $T = T^*$. This property will be critical when we later analyze the spectral properties (eigenvalues and eigenfunctions).

2.2 Integral Operators and Fredholm Equations

2.2.1 Compact Operators

In Banach and Hilbert spaces, the unit ball is not compact. This loss of topological compactness means bounded sequences do not strictly have convergent subsequences. *Compact* operators are mathematically vital because they restore this property, acting as "smoothing" mappings that compress infinite-dimensional spaces into finite-dimensional-like structure.

Definition 2.2.1 (Sequential Compactness)**.** Let $\mathcal{X}$ and $\mathcal{Y}$ be Banach spaces. A linear operator $K : \mathcal{X} \to \mathcal{Y}$ is *compact* if, for every bounded sequence $\{x_n\}_{n=1}^{\infty}$ in $\mathcal{X}$, the image sequence $\{Kx_n\}_{n=1}^{\infty}$ contains a convergent subsequence in $\mathcal{Y}$.

Definition 2.2.2 (Topological Compactness)**.** Equivalently, K is *compact* if the image of the closed unit ball $B_{\mathcal{X}} = \{x \in \mathcal{X} : \|x\|_{\mathcal{X}} \leq 1\}$ is relatively compact in $\mathcal{Y}$ (meaning its closure $\overline{K(B_{\mathcal{X}})}$ is compact).

Compact operators essentially act as "low-pass filters" on infinite-dimensional spaces, aggressively smoothing out high-frequency oscillations. A foundational theorem that justifies our machine learning models and kernel approximations, is the following:

Theorem 2.2.3 (Approximation by Finite-Rank Operators)**.** *Let $\mathcal{H}$ be a Hilbert space. An operator $K : \mathcal{H} \to \mathcal{H}$ is compact if and only if it is the limit of a sequence of finite-rank operators.*

This theorem guarantees that any compact integral operator can be approximated to arbitrary precision by a finite-dimensional matrix projection.

2.2.2 Fredholm Integral Operators and Hilbert-Schmidt kernels

The primary focus of our numerical experiments involves learning operators defined by integral transformations.

Let Ω be a measurable set. A Fredholm integral operator $K : L^2(\Omega) \to L^2(\Omega)$ acts on a function $u \in L^2(\Omega)$ via the rule:

$$Ku(x) = \int_{\Omega} k(x, y)u(y)dy$$

where the bivariate function $k(x, y)$ is known as the *kernel* of the operator.

The properties of the operator K are entirely dictated by the properties of its kernel.

A particularly important class of kernels are the **Hilbert-Schmidt kernels**, which satisfy:

$$\int_{\Omega}\int_{\Omega} |k(x, y)|^2 dxdy < \infty$$

meaning $k \in L^2(\Omega \times \Omega)$. If an integral operator possesses a Hilbert-Schmidt kernel, it is mathematically guaranteed to be a compact operator on $L^2(\Omega)$

Theorem 2.2.4. *Let Ω be a measurable space and $K : L^2(\Omega) \to L^2(\Omega)$ be an integral operator defined by a kernel $k \in L^2(\Omega \times \Omega)$. The standard operator norm $\|K\|_{op}$, defined as the maximum stretch the operator applies to any function $u \in L^2(\Omega)$, is bounded by the L^2 norm of its kernel:*

$$\|K\|_{op} = \sup_{u \neq 0} \frac{\|Ku\|_{L^2(\Omega)}}{\|u\|_{L^2(\Omega)}} \leq \|k\|_{L^2(\Omega \times \Omega)}$$

Consequently, a bounded kernel strictly guarantees a bounded operator.

Proof. By the definition of the integral operator, the squared absolute value of the evaluated function is given by:

$$|Ku(x)|^2 = \left| \int_\Omega k(x,y)u(y)dy \right|^2$$

Applying the Cauchy-Schwarz inequality to the integral with respect to y, we obtain:

$$|Ku(x)|^2 \leq \left(\int_\Omega |k(x,y)|^2 dy \right) \left(\int_\Omega |u(y)|^2 dy \right)$$

Which simplifies to:

$$|Ku(x)|^2 \leq \left(\int_\Omega |k(x,y)|^2 dy \right) \|u\|^2_{L^2(\Omega)}$$

To find the L^2 norm of the resulting function Ku, we integrate both sides with respect to x:

$$\|Ku\|^2_{L^2(\Omega)} = \int_\Omega |Ku(x)|^2 dx \leq \int_\Omega \left(\int_\Omega |k(x,y)|^2 dy \right) dx \cdot \|u\|^2_{L^2(\Omega)}$$

Recognizing the double integral on the right-hand side as the squared norm of the kernel, we have:

$$\|Ku\|^2_{L^2(\Omega)} \leq \|k\|^2_{L^2(\Omega \times \Omega)} \|u\|^2_{L^2(\Omega)}$$

Taking the square root of both sides and dividing by $\|u\|_{L^2(\Omega)}$ (for $u \neq 0$) yields the foundational bound:

$$\|K\|_{op} \leq \|k\|_{L^2(\Omega \times \Omega)}$$

□

2.2.3 Fredholm Equations of the First and Second Kind

In the context of Inverse Problems, we are often given an observed output function $f(x)$ and must determine the unknown input function $u(y)$ that produced it.

Depending on the physical system being modeled, this results in two distinct types of integral equations.

Fredholm Equations of the Second Kind A Fredholm equation of the second kind takes the form:

$$u(x) - \lambda \int_{\Omega} k(x,y)u(y)dy = f(x)$$

where λ is a scalar parameter. Equivalently, this is written as $(I - \lambda K)u = f$. These equations are generally well-posed. Because the identity operator I is not compact in infinite dimensions, the combined operator $(I - \lambda K)$ is bounded away from zero, meaning its inverse is continuous. Small errors in the sensor measurements of $f(x)$ will only result in proportionally small errors in the reconstructed $u(x)$.

Fredholm Equations of the First Kind Conversely, a Fredholm equation of the first kind is written as:

$$\int_{\Omega} k(x,y)u(y)dy = f(x)$$

or simply $Ku = f$. Because K is a compact operator, its inverse K^{-1} (if it exists) is strictly unbounded. Consequently, Fredholm equations of the first kind are inherently ill-posed.The compact nature of K means it aggressively dampens high-frequency components of $u(y)$.

When attempting to invert the operator, any high-frequency noise present in the data $f(x)$ | such as inevitable measurement noise from discrete sensors | will be infinitely amplified by K^{-1}, completely destroying the prediction of $u(y)$. It is this inherent ill-posedness that necessitates the use of Machine Learning as a data-driven regularization framework, and justifies the explicit use of Ridge and Laplace smoothing techniques when directly approximating the kernel.

2.3 Spectral Theory of Compact Operators

To understand how architectures and operator networks approximate unknown integral operators, we must first analyze the internal structure of these operators. Spectral theory allows us to decompose a complex, infinite-dimensional operator into a series of independent, one-dimensional mappings defined by its eigenvalues and eigenfunctions. But first we introduce the notion of positive-definitevity

Let $\mathcal{H}$ be a Hilbert space. An operator K is *positive-definite* if for all non-zero $x \in \mathcal{H}$:

$$\langle Kx, x \rangle > 0$$

Note When an integral operator models a physical system (such as heat diffusion or structural mechanics), its governing kernel often satisfies both of positive-definitevity and self-adjointness. naturally.

2.3.1 The Spectral Theorem for Compact Operators

In finite-dimensional linear algebra, symmetric matrices can be diagonalized using an orthonormal basis of eigenvectors. The Spectral Theorem extends this powerful result to infinite-dimensional Hilbert spaces.

Theorem 2.3.1 (The Spectral Theorem)**.** *Let $K : \mathcal{H} \to \mathcal{H}$ be a compact, self-adjoint operator on a Hilbert space $\mathcal{H}$. Then, there exists an orthonormal basis of $\mathcal{H}$ consisting entirely of the eigenfunctions of K.Mathematically, there exists a sequence of real eigenvalues $\{\lambda_n\}_{n=1}^{\infty}$ and a corresponding set of orthonormal eigenfunctions $\{\phi_n\}_{n=1}^{\infty}$ such that:*

$$K\phi_n = \lambda_n \phi_n \quad \textit{for } n = 1, 2, \dots$$

This theorem implies that the action of the integral operator K on any arbitrary function $u \in \mathcal{H}$ can be perfectly represented as a weighted sum of its projections onto these eigenfunctions:

$$Ku = \sum_{n=1}^{\infty} \lambda_n \langle u, \phi_n \rangle \phi_n$$

2.3.2 The Decay of Eigenvalues and Ill-posedness

While the Spectral Theorem allows us to represent the operator neatly, the compactness of K imposes a severe restriction on its eigenvalues, which fundamentally dictates the difficulty of solving Inverse Problems.

For any compact operator on an infinite-dimensional space, the sequence of eigenvalues must decay to zero:

$$\lim_{n \to \infty} \lambda_n = 0$$

This decay is the precise mathematical source of ill-posedness in Fredholm equations of the first kind ($Ku = f$). If we attempt to construct the inverse operator to solve for the unknown input u, we must divide by the eigenvalues:

$$u = K^{-1} f = \sum_{n=1}^{\infty} \frac{\langle f, \phi_n \rangle}{\lambda_n} \phi_n$$

However, in practical machine learning and experimental setups, we observe noisy data $f^{\delta} = f + \delta$, where δ represents high-frequency measurement or sensor noise. The noise δ generally projects onto the high-frequency eigenfunctions (where n is large). Because $\lambda_n \to 0$ rapidly as $n \to \infty$, dividing the noise projection by a vanishingly small λ_n causes the term to explode toward infinity. Consequently, the inverse mapping is strictly unbounded, and standard numerical inversion immediately fails. This theoretical limitation can be overcome by robust functional approximations provided by randomized neural networks.

2.3.3 Mercer's Theorem and Network Architectures

The spectral decomposition of the operator naturally extends to the kernel itself, providing the ultimate theoretical justification for advanced operator networks like DeepONets and RandONets.

Theorem 2.3.2 (Mercer's Theorem)**.** *Suppose $k(x, y)$ is a continuous, symmetric, positive-definite kernel on $\Omega \times \Omega$. Then the kernel can be expanded in a uniformly and absolutely convergent series in terms of the eigenvalues λ_n and orthonormal eigenfunctions ϕ_n of its corresponding integral operator:*

$$k(x, y) = \sum_{n=1}^{\infty} \lambda_n \phi_n(x) \overline{\phi_n(y)}$$

Connection to Operator Learning

Mercer's Theorem guarantees that the continuous surface of the kernel can be perfectly reconstructed via its eigenfunctions. By truncating this infinite series at a finite rank N, we obtain a highly accurate approximation of the operator:

$$(Ku)(x) \approx \sum_{n=1}^{N} \lambda_n \langle u, \phi_n \rangle \phi_n(x)$$

This finite-rank truncation is the exact mathematical architecture underpinning operator networks. In frameworks like DeepONets, the architecture is split into two pathways: The Branch Network evaluates the input function u and learns to approximate the coefficients $(\lambda_n \langle u, \phi_n \rangle)$.The Trunk Network evaluates the spatial domain x and learns to approximate a latent basis spanning the eigenfunctions $\phi_n(x)$.By combining these networks via a dot product, DeepONets and RandONets are fundamentally performing a learned, finite-dimensional Mercer expansion.

Chapter 3

Operator Learning via Randomized Architectures

3.1 The Operator Learning Framework

Classical machine learning models | such as Multi-Layer Perceptrons (MLPs) | are designed to learn mappings between finite-dimensional Euclidean spaces, $f_\theta : \mathbb{R}^{d_{in}} \to \mathbb{R}^{d_{out}}$. However, learning the solution operator to a Fredholm integral equation requires a fundamentally different paradigm. We seek a parameterized model $\mathcal{F}_\theta$ that maps an entire input function to an entire output function, $\mathcal{F}_\theta : \mathcal{V} \to \mathcal{U}$, where $\mathcal{V}$ and $\mathcal{U}$ are infinite-dimensional Banach or Hilbert spaces. As it seems this may look like an easy task coming up with machine learning structures that are complex, however the complexity rises even more in the setting of infiite-dimensions since the true challenge is not the models as is fighting the ill-posedness of such problems.

3.1.1 Continuous vs. Discrete Formulation

In the theoretical setting, an unknown compact integral operator $\mathcal{G} : \mathcal{V} \to \mathcal{U}$ maps an input function $v(y)$ to an output function $u(x)$. In a practical, data-driven setting, we do not have access to the full continuous functions or the analytical form of $\mathcal{G}$.

Instead, we are provided with a finite dataset of paired observations:

$$\mathcal{D} = \{(v_i, u_i)\}_{i=1}^{D}$$

where each $v_i \in \mathcal{V}$ is drawn from a specific measure μ supported on the input space, and $u_i = \mathcal{G}(v_i) \in \mathcal{U}$ is the corresponding image.

Discretization via Sensor Locations Computers cannot directly process v_i and u_i as continuous entities. To represent these functions computationally, they must be evaluated at discrete

spatial coordinates. Let the input domain Ω_v be sampled at m_v discrete points (often termed "sensor locations"), and the output domain Ω_u be sampled at m_u points:

$$\mathbf{y} = [y_1, y_2, \ldots, y_{m_v}]^T \subset \Omega_v$$

$$\mathbf{x} = [x_1, x_2, \ldots, x_{m_u}]^T \subset \Omega_u$$

The continuous functions are therefore realized computationally as finite-dimensional vectors representing point-wise evaluations:

$$\mathbf{v}_i = [v_i(y_1), v_i(y_2), \ldots, v_i(y_{m_v})]^T \in \mathbb{R}^{m_v}$$

$$\mathbf{u}_i = [u_i(x_1), u_i(x_2), \ldots, u_i(x_{m_u})]^T \in \mathbb{R}^{m_u}$$

Discrete Approximation of the L^2 Error Because our theoretical framework resides in the Hilbert space $L^2(\Omega)$, the canonical metric for evaluating the distance between a predicted function $\hat{u}$ and the true function u is the relative L^2 error:

$$\mathcal{E}_{rel} = \frac{\|\hat{u} - u\|_{L^2(\Omega)}}{\|u\|_{L^2(\Omega)}} = \frac{\left(\int_\Omega |\hat{u}(x) - u(x)|^2 dx\right)^{1/2}}{\left(\int_\Omega |u(x)|^2 dx\right)^{1/2}}$$

To compute this during training and evaluation, the continuous integral is approximated discretely over the grid $\mathbf{x}$. If the grid points x_j are uniformly spaced with distance Δx, the continuous norm is approximated via Riemann sums or the trapezoidal rule. For instance, a scaled Euclidean norm serves as a discrete surrogate for the L^2 norm:

$$\|u\|_{L^2}^2 \approx \sum_{j=1}^{m_u} |u(x_j)|^2 \Delta x = \Delta x \|\mathbf{u}\|_2^2$$

Crucially, a well-formulated operator learning architecture—such as DeepONets or the ELM-based expansions utilized in this study—must be mesh-free. This means the parameters θ of the network $\mathcal{F}_\theta$ should remain independent of the specific grid resolution m_v or m_u. The network must learn the underlying continuous operator $\mathcal{G}$, allowing it to be trained on one grid resolution and evaluated on an entirely different one without retraining.

3.1.2 Empirical Risk Minimization for Operators

With the functional data discretized, we must formalize the optimization task. In standard machine learning, Empirical Risk Minimization (ERM) minimizes a loss function over a finite-dimensional parameter space. In operator learning, we extend this minimization to function spaces.Let $\mathcal{F}_\theta : \mathcal{V} \to \mathcal{U}$ be our neural operator parameterized by θ (which, in the case of our randomized architectures, encompasses the output weights W). The theoretical goal is to find

the optimal parameters θ^* that minimize the expected risk over the entire distribution of input functions μ:

$$\mathcal{L}(\theta) = \mathbb{E}_{v\sim\mu}\left[\|\mathcal{F}_\theta(v) - \mathcal{G}(v)\|_{\mathcal{U}}^2\right]$$

Because we only possess a finite dataset $\mathcal{D}$ of size D, we cannot compute this exact expectation. Instead, we minimize the empirical risk, formulated as the mean squared functional error over the training set:

$$\hat{\mathcal{L}}(\theta) = \frac{1}{D}\sum_{i=1}^{D} \|\mathcal{F}_\theta(v_i) - u_i\|_{\mathcal{U}}^2$$

Substituting our discrete L^2 approximation into the empirical risk functional yields the strictly computable loss function:

$$\hat{\mathcal{L}}_{discrete}(\theta) = \frac{1}{D}\sum_{i=1}^{D}\left(\frac{1}{m_u}\sum_{j=1}^{m_u} |\mathcal{F}_\theta(v_i)(x_j) - u_i(x_j)|^2\right)$$

This formulation represents a profound shift from standard deep learning. We are no longer regressing a vector $\mathbf{v}_i$ to a vector $\mathbf{u}_i$; we are training the parameters θ to regress the evaluation of the mapping $\mathcal{F}_\theta(v_i)$ at any arbitrary continuous coordinate x_j against the true functional evaluation $u_i(x_j)$. This continuous-to-continuous ERM framework serves as the exact optimization objective for the Galerkin projections and Latent Extreme Operator Regressions developed in the subsequent sections.

3.2 Galerkin Projection and Randomized Bases

To computationally solve integral equations, the infinite-dimensional spaces governing the continuous operators must be reduced to finite-dimensional subspaces. In classical numerical analysis, this is achieved via the *Galerkin method* using fixed analytic bases (such as Fourier series or orthogonal polynomials). In our operator learning framework, we replace these classical analytical bases with data-driven, randomized functional bases generated by neural networks.

3.2.1 Galerkin Projection Operator

Let $K : \mathcal{V} \to \mathcal{U}$ be a compact integral operator. We seek to learn the mapping $Kv = u$. Because the functional spaces $\mathcal{V}$ and $\mathcal{U}$ are infinite-dimensional, we must project them onto finite-dimensional subspaces to render the problem computable.

Let $\mathcal{V}_N = \text{span}\{\psi_1, \psi_2, \ldots, \psi_N\} \subset \mathcal{V}$ be an N-dimensional subspace of input functions, and let $\mathcal{U}_M = \text{span}\{\phi_1, \phi_2, \ldots, \phi_M\} \subset \mathcal{U}$ be an M-dimensional subspace of output functions. An input function $v \in \mathcal{V}$ and its corresponding output $u \in \mathcal{U}$ can be approximated as finite linear

combinations of these basis functions:

$$v(y) \approx v_N(y) = \sum_{j=1}^{N} c_j \psi_j(y)$$

$$u(x) \approx u_M(x) = \sum_{i=1}^{M} d_i \phi_i(x)$$

where $\mathbf{c} = [c_1, \ldots, c_N]^T$ and $\mathbf{d} = [d_1, \ldots, d_M]^T$ are the coordinate vectors in the latent basis space.When we apply the operator K to our approximate input v_N, we generate a residual error $R_N(x) = K(v_N)(x) - u(x)$ because the exact image of v_N may not lie perfectly within $\mathcal{U}_M$. The Galerkin condition strictly requires that this residual be orthogonal to the output test space $\mathcal{U}_M$. Taking the inner product with each test function ϕ_i:

$$\langle K(v_N) - u_M, \phi_i \rangle_{\mathcal{U}} = 0 \quad \text{for } i = 1, 2, \ldots, M$$

Using the linearity of K and substituting the expansions:

$$\sum_{j=1}^{N} c_j \langle K(\psi_j), \phi_i \rangle_{\mathcal{U}} = \sum_{i=1}^{M} d_i \langle \phi_i, \phi_i \rangle_{\mathcal{U}}$$

If the output basis $\{\phi_i\}$ is orthonormal, the right side simply becomes the coefficient d_i. We can define a transformation matrix $W \in \mathbb{R}^{M \times N}$, where the entries are given by $W_{ij} = \langle K(\psi_j), \phi_i \rangle_{\mathcal{U}}$.The infinite-dimensional integral operator mapping $v \mapsto u$ is thus perfectly translated into a finite-dimensional matrix-vector multiplication in the latent coordinate space:

$$\mathbf{d} = W\mathbf{c}$$

In classical methods, constructing the matrix W requires analytically integrating the kernel against fixed polynomials. In our operator learning framework, the basis functions $\{\psi_j\}$ and $\{\phi_i\}$ are entirely generated by the hidden layers of randomized neural networks, and W becomes the core matrix of trainable parameters.

3.2.2 Extreme Learning Machines (ELMs) as Randomized Bases

To generate these finite-dimensional subspaces, we utilize the Extreme Learning Machine (ELM) architecture. Originally proposed by Huang, Zhu, and Siew [6] as a computationally efficient alternative to gradient-descent-based training, the ELM transforms the highly non-convex landscape of deep learning into a strictly convex, linear algebra problem.

A standard single-hidden-layer feedforward neural network (SLFN) with M hidden neurons and a non-linear activation function σ evaluates a point $x \in \mathbb{R}^d$ via:

$$f(x) = \sum_{m=1}^{M} \beta_m \sigma(\mathbf{w}_m \cdot x + b_m)$$

where $\mathbf{w}_m$ are the input weights, b_m are the biases, and β_m are the output weights. In standard backpropagation, all parameters $(\mathbf{w}_m, b_m, \beta_m)$ are iteratively updated, which is computationally expensive and prone to vanishing gradients.

The Randomized Architecture The defining mathematical mechanism of the ELM is that the hidden layer parameters $(\mathbf{w}_m, b_m)$ are randomly assigned from a continuous probability distribution (such as a standard Gaussian $\mathcal{N}(0, 1)$) and are permanently fixed. The network does not learn its feature extraction; it generates a randomized latent space.By defining the hidden node output as $\sigma_m(x) = \sigma(\mathbf{w}_m \cdot x + b_m)$, the network's mapping becomes:

$$f(x) = \sum_{m=1}^{M} \beta_m \sigma_m(x) = \boldsymbol{\sigma}(x)^T \boldsymbol{\beta}$$

where $\boldsymbol{\sigma}(x) = [\sigma_1(x), \ldots, \sigma_M(x)]^T$ acts as a randomized feature vector.

Connection to Function Approximation By fixing the hidden weights, the ELM effectively generates a fixed, M-dimensional functional subspace $\mathcal{H}_M = \text{span}\{\sigma_1(x), \ldots, \sigma_M(x)\}$. The Universal Approximation Theorem for randomized networks guarantees that as $M \to \infty$, the span of these randomly parameterized functions becomes dense in L^2. Therefore, the sequence of random features $\{\sigma_m(x)\}$ serves as a mathematically valid basis for the Galerkin projection defined in Section 3.2.1.

Linear Least-Squares Optimization Because the hidden layer is fixed, the highly non-linear problem of training a neural network reduces to finding the optimal linear combination of these random basis functions. Given N observations $\mathbf{X} = [x_1, \ldots, x_N]^T$ and corresponding targets $\mathbf{T} = [t_1, \ldots, t_N]^T$, we construct the hidden-layer output matrix $\mathbf{H} \in \mathbb{R}^{N \times M}$:

$$\mathbf{H} = \begin{bmatrix} \boldsymbol{\sigma}(x_1)^T \\ \vdots \\ \boldsymbol{\sigma}(x_N)^T \end{bmatrix}$$

The learning problem immediately collapses to solving the overdetermined linear system:

$$\mathbf{H}\boldsymbol{\beta} = \mathbf{T}$$

Because $\mathbf{H}$ is generally not square, the exact solution is obtained via Empirical Risk Minimization using the Moore-Penrose pseudoinverse $\mathbf{H}^\dagger$:

$$\boldsymbol{\beta} = \mathbf{H}^\dagger \mathbf{T}$$

By employing Tikhonov regularization (Ridge regression) to penalize the L^2 norm of the output weights and prevent overfitting to high-frequency noise, the closed-form solution becomes:

$$\boldsymbol{\beta} = (\mathbf{H}^T\mathbf{H} + \alpha I)^{-1}\mathbf{H}^T\mathbf{T}$$

where $\alpha > 0$ is the regularization parameter.This formulation provides a massive computational advantage. Training the network requires no iterative epochs or backpropagation; it merely requires a single matrix inversion. In the following sections, we will deploy this randomized linear inversion to computationally approximate the latent operator matrix W described by the Galerkin projection.

3.3 Advanced Neural Operators

In this section, we present the two state-of-the-art neural operators that serve as the advanced benchmarks for our study: Deep Operator Networks (DeepONets) and Randomized Operator Networks (RandONets).

3.3.1 Deep Operator Networks (DeepONets)

Introduced by Lu et al. [7], the Deep Operator Network (DeepONet) fundamentally restructured neural architectures to process functional data. Standard neural networks inherently couple the processing of the input data with the spatial coordinates. DeepONets strictly uncouple these two domains by splitting the architecture into two parallel subnetworks: the **Branch** network and the **Trunk** network.

The Branch Network: This network is tasked with processing the input function $v \in \mathcal{V}$. Because v is observed discretely, the Branch takes the sensor evaluations $\mathbf{v} = [v(y_1), \ldots, v(y_{m_v})]^T$ as input and outputs a p-dimensional latent vector of coefficients:

$$\mathbf{b}(v) = [b_1(v), b_2(v), \ldots, b_p(v)]^T$$

The Trunk Network: This network is tasked with processing the continuous spatial coordinates. It takes a coordinate $x \in \Omega_u$ as input and outputs a p-dimensional latent vector of basis functions:

$$\mathbf{t}(x) = [t_1(x), t_2(x), \ldots, t_p(x)]^T$$

The final prediction of the operator acting on v evaluated at x is given by the inner product of these two latent representations, plus an optional bias b_0:

$$\mathcal{G}_\theta(v)(x) = \sum_{k=1}^{p} b_k(v)t_k(x) + b_0$$

Theoretical Justification and Mercer's Expansion This specific architecture is not arbitrary; it is the direct neural realization of the Operator Universal Approximation Theorem proposed by Chen and Chen [8]. The theorem guarantees that any continuous nonlinear operator can be uniformly approximated by the dot product of two neural networks.Furthermore, this uncoupled dot product is the computational analogue to Mercer's Theorem Section 2.3.3 and the spectral decomposition established in Section 2.3.1. The Trunk network effectively acts as a universal approximator learning the continuous data-driven eigenfunctions $\{\phi_n(x)\}$ of the true integral operator, while the Branch network learns the projection coefficients corresponding to the input function. By training both networks simultaneously via backpropagation, DeepONets learn a highly expressive, finite-rank functional expansion.

3.3.2 Randomized Operator Networks (RandONets)

While DeepONets are incredibly expressive, they suffer from the traditional drawbacks of deep learning: training two deep subnetworks simultaneously via backpropagation is computationally expensive, requires massive datasets, and is highly non-convex (prone to vanishing gradients and local minima).To resolve this computational bottleneck, Fabiani et al. [9] proposed the RandONet, which brilliantly fuses the structural elegance of the DeepONet with the randomized efficiency of Extreme Learning Machines.In the RandONet framework, the deep, fully trainable Branch and Trunk networks are entirely replaced by randomized feature expansions (Random Vector Functional Links or ELMs).

Randomized Trunk: The spatial coordinates x are mapped to a latent space via fixed, randomly parameterized hidden neurons, instantly generating a randomized basis $t_k(x) = \sigma(\mathbf{w}_k^{trunk} \cdot x + c_k^{trunk})$.

Randomized Branch: Similarly, the discrete input function $\mathbf{v}$ is mapped via a separate set of fixed, random hidden neurons to generate the feature coefficients $b_k(v) = \sigma(\mathbf{w}_k^{branch} \cdot \mathbf{v} + c_k^{branch})$.

Because the hidden feature extraction layers are randomized and frozen, the only trainable parameters in the RandONet are the final output weights connecting these expanded spaces.This architectural shift profoundly alters the optimization landscape. Just as we derived in Section 3.2.2, substituting the deep nonlinear training with fixed random bases reduces the entire operator learning task to a strictly convex, linear least-squares problem.

RandONets preserve the essential Mercer-like expansion capability | learning operators as the inner product of functional coefficients and spatial bases | but execute the training orders of magnitude faster than standard DeepONets, requiring only a singular matrix inversion rather than thousands of iterative epochs.

3.4 Formulation of Investigated models

Building upon the theoretical foundations of Galerkin projections and randomized feature expansions, this section details the mathematical formulations of the core models investigated in this study. The primary objective across these architectures is to bypass the computationally expensive backpropagation associated with standard deep learning, leveraging the Extreme Learning Machine (ELM) paradigm to solve operator mappings via direct linear inversion.

3.4.1 Latent Extreme Operator Regression (LEOR)

The Latent Extreme Operator Regression (LEOR) model elegantly circumvents the need to approximate the integral kernel directly. Instead, it frames operator learning strictly as a linear transformation between the latent feature spaces of the input and output functions.

Given a dataset of D discretely observed function pairs $\mathcal{D} = \{(v_d, u_d)\}_{d=1}^{D}$, the LEOR algorithm proceeds in three distinct stages:

Stage 1: Randomized Basis Expansion By the Universal Approximation Theorem for ELMs, we construct a rich, randomized span for both the input space $\mathcal{V}$ and the output space $\mathcal{U}$. We define two independent sets of fixed, random hidden features: $\{\sigma_m^{(v)}(y)\}_{m=1}^{M_v}$ for the input domain and $\{\sigma_m^{(u)}(x)\}_{m=1}^{M_u}$ for the output domain.Every function in the dataset is independently projected onto these bases via standard ELM least-squares fitting:

$$v_d(y) \approx \sum_{m=1}^{M_v} \beta_m^{(v,d)} \sigma_m^{(v)}(y)$$

$$u_d(x) \approx \sum_{m=1}^{M_u} \beta_m^{(u,d)} \sigma_m^{(u)}(x)$$

Stage 2: Latent Coefficient Extraction Because the basis functions are fixed, the continuous functions v_d and u_d are now entirely characterized by their optimal coordinate vectors in the latent space:

$$v_d \mapsto \boldsymbol{\beta}_d^{(v)} \in \mathbb{R}^{M_v} \quad \text{and} \quad u_d \mapsto \boldsymbol{\beta}_d^{(u)} \in \mathbb{R}^{M_u}$$

By collecting these coefficient vectors for all D samples, we construct the global latent matrices:

$$B_v = \left[\boldsymbol{\beta}_1^{(v)}, \ldots, \boldsymbol{\beta}_D^{(v)}\right] \in \mathbb{R}^{M_v \times D}$$

$$B_u = \left[\boldsymbol{\beta}_1^{(u)}, \ldots, \boldsymbol{\beta}_D^{(u)}\right] \in \mathbb{R}^{M_u \times D}$$

Stage 3: Operator Regression Because the underlying integral operator is linear, its action in the latent space must also be linear. We assume the existence of a latent transition matrix $W \in \mathbb{R}^{M_u \times M_v}$ such that:

$$B_u = W B_v$$

The operator learning problem is thus reduced to finding the matrix W that minimizes the Frobenius norm $\|B_u - WB_v\|_F^2$. Using Tikhonov regularization with penalty α, the closed-form solution is derived via the regularized Moore-Penrose pseudoinverse:

$$W = B_u B_v^T (B_v B_v^T + \alpha I)^{-1}$$

To evaluate the operator on a completely unseen function v_{new}, one simply extracts its latent vector $\boldsymbol{\beta}_{new}^{(v)}$, multiplies by W to obtain $\boldsymbol{\beta}_{new}^{(u)}$, and reconstructs the continuous output function via the $\mathcal{U}$ basis.

3.4.2 Direct Kernel Approximation (ELM on the Kernel)

While LEOR operates implicitly in the latent space, this model explicitly targets the physical properties of the Fredholm equation. Because a Fredholm operator is entirely defined by its kernel $k(x, y)$, learning the operator is mathematically equivalent to learning this 2D surface.

We parameterize the kernel as the cross-product of two independent spatial ELMs | one acting on the output coordinate x, and one acting on the integration variable y. Let $\boldsymbol{\sigma}_x(x) \in \mathbb{R}^{M_x}$ and $\boldsymbol{\sigma}_y(y) \in \mathbb{R}^{M_y}$ be the fixed, randomized feature vectors for the respective domains. The kernel is approximated by the bilinear form:

$$k(x, y) \approx \boldsymbol{\sigma}_x(x)^T \mathbf{W} \boldsymbol{\sigma}_y(y)$$

where $\mathbf{W} \in \mathbb{R}^{M_x \times M_y}$ is the trainable weight matrix.The mathematical brilliance of this formulation reveals itself when we substitute it back into the Fredholm equation of the first kind:

$$u(x) = \int_\Omega k(x, y) v(y) dy \approx \int_\Omega \left(\boldsymbol{\sigma}_x(x)^T \mathbf{W} \boldsymbol{\sigma}_y(y) \right) v(y) dy$$

Because the spatial feature vector $\boldsymbol{\sigma}_x(x)$ and the weight matrix $\mathbf{W}$ do not depend on the integration variable y, they can be factored out of the integral:

$$u(x) \approx \boldsymbol{\sigma}_x(x)^T \mathbf{W} \left(\int_\Omega \boldsymbol{\sigma}_y(y) v(y) dy \right)$$

Let $\mathbf{c}_v = \int_\Omega \boldsymbol{\sigma}_y(y) v(y) dy \in \mathbb{R}^{M_y}$. This vector represents the exact numerical projection of the input function $v(y)$ onto the y-domain random features, computable via standard numerical quadrature (e.g., trapezoidal rule). The prediction of the model thus simplifies to:

$$u(x) \approx \boldsymbol{\sigma}_x(x)^T \mathbf{W} \mathbf{c}_v$$

By vectorizing the outputs over discrete spatial grids and D training samples, we once again reduce the highly complex kernel reconstruction problem to a sparse, linear least-squares optimization for the elements of $\mathbf{W}$, natively supporting the inclusion of spatial derivatives and boundary condition penalties via Tikhonov/Laplace smoothing.

3.5 The Ill-Posedness of Kernel Learning and Regularization

While the randomized architectures and linear least-squares formulations derived in the previous section provide profound computational acceleration, they do not inherently solve the mathematical ill-posedness of *Inverse* Problems. In fact, attempting to learn an infinite-dimensional operator from a finite dataset introduces a secondary layer of ill-posedness: the non-uniqueness of the kernel itself. To guarantee convergence to physically meaningful solutions, we must construct a rigorously regularized composite loss function.

3.5.1 The Null Space of Finite Training Data

As established in Section 3.1, our empirical knowledge of the true operator K_{true} is strictly limited to a finite dataset of D discrete function pairs $\mathcal{D} = \{(v_d, u_d)\}_{d=1}^{D}$.

Let $\mathcal{V}_D = \text{span}\{v_1, \ldots, v_D\}$ be the finite-dimensional subspace of input functions spanned by our training data. Because $\mathcal{V}$ is an infinite-dimensional Hilbert space, the orthogonal complement $\mathcal{V}_D^{\perp}$ is infinitely vast. Consequently, there exists an infinite set of non-trivial integral operators | which we define as the null operators K_{null}-that perfectly map every function in our training set to zero:

$$K_{null} v_d = 0 \quad \text{for all } d = 1, \ldots, D$$

If we train our neural network (e.g., the ELM on the Kernel) relying exclusively on the empirical data loss, the network is free to learn any estimated operator K_{est} of the form:

$$K_{est} = K_{true} + K_{null}$$

When evaluated on the training dataset, this estimated operator yields an empirical error of exactly zero, because $(K_{true} + K_{null})v_d = K_{true} v_d = u_d$. However, when the network is tested on a novel function v_{new} containing frequency components outside the training span $\mathcal{V}_D$, the K_{null} term becomes highly active. The network will output wildly inaccurate predictions.Empirical risk minimization alone cannot distinguish between K_{true} and $K_{true} + K_{null}$. To force the network to select the true physical operator, we must systematically collapse this null space by injecting prior mathematical and physical knowledge directly into the loss function.

3.5.2 Physics-Informed and Structural Regularizers

To constrain the hypothesis space, we abandon standard least-squares in favor of a composite optimization objective. For Model 2 (Direct Kernel Approximation), the final parameters (such as the matrix W) are obtained by minimizing the total regularized loss:

$$\mathcal{L}_{total} = \mathcal{L}_{data} + \mathcal{L}_{ridge} + \mathcal{L}_{laplace} + \mathcal{L}_{sym} + \mathcal{L}_{BC}$$

1. The Data Fidelity Term The foundation of the loss is the discrete empirical error, which measures how well the approximated operator maps the inputs to the outputs over the sampled spatial grid:

$$\mathcal{L}_{data} = \frac{1}{D}\sum_{d=1}^{D} \|K_{est}v_d - u_d\|^2_{L^2(\Omega)}$$

2. Ridge Penalty for Parameter Bounding As proved in Section 2.2.2, the operator norm is bounded by the L^2 norm of its kernel. To prevent the network from learning a K_{null} with massive, unbounded oscillations that amplify high-frequency noise, we must strictly bound the norm of the estimated operator. Because our randomized bases are fixed, the operator norm is entirely dictated by the magnitude of the trainable weights W. We enforce this via Tikhonov (Ridge) regularization, heavily penalizing the Frobenius norm of the weight matrix:

$$\mathcal{L}_{ridge} = \alpha\|W\|^2_F$$

where α is a tunable hyperparameter. This guarantees that K_{est} remains a strictly bounded, continuous mapping.

3. Laplace Smoothing for Spatial Coherence Compact integral operators are mathematically defined by their smoothing properties. A true physical kernel $k_{true}(x, y)$ typically represents diffusion or inverse-differential mechanics, meaning the 2D surface of the kernel must be spatially smooth. K_{null} kernels, conversely, often manifest as high-frequency "checkerboard" artifacts that exploit the discrete grid spacing.To eliminate these artifacts, we apply a Laplace smoothing penalty that penalizes severe spatial gradients across the learned kernel surface:

$$\mathcal{L}_{laplace} = \lambda \int_{\Omega}\int_{\Omega} |\Delta k(x, y)|^2 dxdy$$

where $\Delta = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}$ is the continuous Laplacian operator, computationally approximated via discrete finite differences. By scaling the hyperparameter λ, we explicitly force the neural network to learn a low-pass filtering kernel.

4. Kernel Symmetry for Self-Adjoint Operators Many Fredholm integral equations arise as the inverses of self-adjoint differential operators. As established in Section 2.3.1, a self-adjoint operator on $L^2(\Omega)$ mathematically requires its kernel to be strictly symmetric, such that $k(x, y) = k(y, x)$.If the training data $\mathcal{D}$ does not perfectly uniformly sample the function space, the empirical loss $\mathcal{L}_{data}$ may inadvertently allow the network to learn an asymmetric K_{null} that satisfies the data points but violates the fundamental physics of the system (e.g., implying that a source at y affects x differently than a source at x affects y). To eradicate this non-physical

null space, we introduce a soft symmetry constraint that penalizes the variance between the kernel and its transpose across the domain:

$$\mathcal{L}_{sym} = \beta \int_{\Omega} \int_{\Omega} |k(x,y) - k(y,x)|^2 dxdy$$

For architectures like Model 2, if the spatial features for x and y are drawn from the same random distribution, this penalty effectively forces the latent weight matrix to become symmetric ($\|W - W^T\|_F^2 \to 0$).

5. Boundary and Initial Conditions (Green's Function Priors) Finally, when the integral operator acts as a Green's function for a specific differential boundary value problem, the kernel must strictly adhere to the boundary conditions of that underlying system. For example, if the physical system is bound by Dirichlet conditions where $u(0) = u(1) = 0$, the kernel must satisfy $k(0,y) = k(1,y) = 0$ for all y. We enforce this structural prior by evaluating the network exclusively at the spatial boundaries and heavily penalizing any deviation from zero:

$$\mathcal{L}_{BC} = \gamma \int_{\Omega} \left(|k(0,y)|^2 + |k(1,y)|^2 \right) dy$$

By minimizing this carefully constructed $\mathcal{L}_{total}$, the optimization landscape is constrained so tightly by boundedness, smoothness, symmetry, and physical boundaries that the non-physical null space K_{null} is entirely eradicated, allowing the network to successfully reconstruct K_{true} even from limited, noisy data.

Summarizing this subsection, we can see that ill-posedness even though it requires regularizations techiques, provides us with the abilty to formulate under different metrics or different norms a plethora of loss function structures, according to our needs and our aims. Also to note that many of the regularization terms do not change drastically the convexity of the optimization problem under the Data Fidelity term, thus we benefit by linear algebra closed form solutions or iterative subspace shcemes

Chapter 4

Proposed Operator Learning Models and Applications

4.1 Objective

In this chapter, we will present a set of models that were used for experiments on a set of inverse problems. We will study their empirical behavior and assess their weaknesses and strengths. NOte that further results and plots are solely shown in the Appendix section of this thesis.

4.2 Experimental Setup and Methodology

To rigorously test the models against varying degrees of mathematical complexity and ill-posedness, we must define a standardized computational environment. This includes the selection of benchmark operators, the generation of functional data, and the statistical metrics utilized for evaluation.

4.2.1 The Benchmark Fredholm Equations

All experiments in this study are conducted on 1D Fredholm integral equations of the first kind over the domain $\Omega = [0, 1]$. The forward mapping is defined as:

$$u(x) = \int_0^1 k(x, y)v(y)dy$$

To test the adaptability of the neural operators, we define four distinct ground-truth kernels $k(x, y)$, chosen specifically for their diverse analytical and structural properties:

- **Exponential Decay Kernel ($k_1(x, y) = e^{-|x-y|}$)**:This kernel represents a standard, strictly positive, symmetric diffusion process. It is continuous everywhere but exhibits a sharp

peak along the diagonal $x = y$, making it an excellent benchmark for measuring a model's ability to capture localized spatial correlations. Associated with the mixed boundary condtions $u(0) = u'(0)$ & $u(1) + u'(1) = 0$

- **Polynomial Kernel ($k_2(x, y) = x^2 - y^2$)**:Unlike typical physical kernels, this kernel is highly smooth, separable, and non-symmetric. Because it is separable (a finite sum of products of x and y), the underlying integral operator has a strictly finite rank. This tests whether the models can natively compress their latent representations to match low-rank, non-physical mappings.

- **Brownian Covariance Kernel ($k_3(x, y) = \min(x, y)$)**:This is a classic covariance kernel associated with Brownian motion and acts as the inverse of the Laplacian operator with mixed boundary conditions ($u(0) = 0$, $u'(1) = 0$). While continuous, its first derivative is discontinuous along the diagonal, testing the network's ability to approximate non-differentiable manifolds.

- **1D Poisson Green's Function ($k_4(x, y)$)**:The most physically rigorous benchmark is the Green's function corresponding to the 1D Poisson equation ($-\frac{d^2u}{dx^2} = v$) subject to homogeneous Dirichlet boundary conditions ($u(0) = u(1) = 0$). The kernel is defined piecewise as:

$$k_4(x, y) = \begin{cases} x(1 - y), & \text{if } x \leq y \\ y(1 - x), & \text{if } y < x \end{cases}$$

  Equivalently written as $k_4(x, y) = \min(x, y) - xy$. This benchmark tests the absolute limits of the architectures, as a successful model must capture the boundary constraints, the diagonal non-smoothness, and the strict symmetry inherent to the physical system.

4.2.2 Data Generation and Discretization

Because operator learning models map entire functions, our dataset $\mathcal{D} = \{(v_d, u_d)\}_{d=1}^{D}$ consists of functional pairs rather than standard scalar vectors.

Input Function Generation: To ensure the neural operators learn a generalized mapping rather than memorizing specific curves, the input functions $v(y)$ are generated as random trajectories. For the standard analytical kernels (Kernels 1{3), inputs are sampled from Gaussian Random Fields (GRFs) to provide smooth, continuous variation. However, specifically for the 1D Poisson equation benchmark, the forcing functions $v(y)$ are generated using finite sums of random sines and cosines.

Spatial Discretization: Computers must evaluate these continuous functions over a discrete grid. The domain $\Omega = [0, 1]$ is uniformly discretized into M sensor points (e.g., $M = 100$), yielding a spatial resolution of $\Delta x = 1/M$.For each generated input vector $\mathbf{v}_d \in \mathbb{R}^M$, the corresponding true output vector $\mathbf{u}_d \in \mathbb{R}^M$ is computed via simple discretization scheme of the integral applied to the ground-truth kernels:

$$\mathbf{u}_d(x_i) = \sum_{j=1}^{M} k(x_i, y_j)\mathbf{v}_d(y_j)\Delta y$$

The dataset of D pairs is then randomly partitioned into a training set for model optimization and an unseen test set for predictive evaluation.

4.2.3 Statistical Evaluation Metrics

To quantify the predictive capability of the models, we evaluate the discrete predictions $\hat{\mathbf{u}}$ against the true observations $\mathbf{u}$ over the test set using standard functional norms. The primary metrics include the Mean Squared Error (MSE), alongside the relative L^1, L^2, and L^∞ (maximum absolute) errors:

$$\text{Rel } L^2 = \frac{\|\mathbf{u} - \hat{\mathbf{u}}\|_2}{\|\mathbf{u}\|_2} = \left(\frac{\sum_i |u_i - \hat{u}_i|^2}{\sum_i |u_i|^2} \right)^{1/2}$$

$$\text{Rel } L^\infty = \frac{\|\mathbf{u} - \hat{\mathbf{u}}\|_\infty}{\|\mathbf{u}\|_\infty} = \frac{\max_i |u_i - \hat{u}_i|}{\max_i |u_i|}$$

Bootstrapping for Robustness: A core contribution of this experimental setup is the rigorous statistical validation of model stability. Because Extreme Learning Machines rely on randomized weight initialization, a single run may misrepresent the model's true capability. Therefore, performance is evaluated using statistical bootstrapping.The models are initialized, trained, and tested across numerous independent randomized iterations. From this bootstrap distribution, we extract the statistical mean, standard deviation, minimum, and maximum for every error metric. The distributions of these errors are visualized using boxplots to explicitly demonstrate not just the accuracy, but the stability and variance of each architecture across the differing kernels.

Spectral Evaluation: Beyond predictive error, models that explicitly reconstruct the integral kernel are evaluated on their spectral accuracy. The matrix representation of the learned kernel K_{est} is subjected to eigendecomposition. The extracted eigenvalues $\{\hat{\lambda}_n\}$ are then compared directly against the true analytical eigenvalues $\{\lambda_n\}$ of the Fredholm operator, providing another measure of how well the neural networks have captured the underlying topological physics of the system.

4.3 Baseline and Latent Subspace Mappings

Here we will evaluate the capicity and accuracy the simplest Naive ELM and first-operator-theoretic model Latent Extreme Operator Regression. We experimented with these models across all the different types of simulated data and performed bootstapp evaluations of the models to assess their capabilities. We then computed the appropriate metrics such MSE and the Relative Errors.

4.3.1 The Naive ELM Baseline: Proof of Ill-Posedness

Before deploying operator-specific networks, we evaluate a Naive Extreme Learning Machine (ELM). In this architecture, the network ignores the underlying continuous functional nature of the data. It attempts to map the discrete input vector $\mathbf{v} \in \mathbb{R}^{m_v}$ directly to the discrete output vector $\mathbf{u} \in \mathbb{R}^{m_u}$ using a single hidden layer of randomized neurons, optimizing exclusively for the point-wise empirical loss.

The bootstrap evaluation of the Naive ELM reveals a catastrophic failure to generalize. Across all four tested kernels, the model yielded a minimum relative L^2 error of approximately 40%. Furthermore, the bootstrap boxplots (detailed in Appendix) demonstrate a massive spread in the error distributions, indicating extreme predictive instability.

This failure is not a flaw in the training mechanism, but a proof of mathematical ill-posedness. Because the Naive ELM operates entirely on the fixed spatial grid and lacks any mechanism to project the data into a continuous function space, it is utterly defenseless against the smoothing properties of the integral operator. It severely overfits to the high-frequency measurement noise of the training data. This empirical collapse explicitly proves that standard finite-dimensional neural architectures are fundamentally insufficient for solving continuous Inverse Problems, necessitating the shift to latent functional mappings.

4.3.2 Latent Extreme Operator Regression (LEOR): Performance on Analytical Kernels

Having established the baseline, we implement the Latent Extreme Operator Regression (LEOR) model. Rather than mapping discrete grid points, LEOR projects the input and output functions onto independent, randomized continuous functional bases generated by the ELM hidden layers. The continuous integral operator is thus gracefully reduced to a discrete linear transformation strictly within the latent coordinate space ($B_u = W B_v$).

The predictive metrics for LEOR show an immediate and profound improvement over the Naive ELM. When evaluated on the analytical kernels | specifically the smooth $e^{-|x-y|}$, the poly-

nomial $x^2 - y^2$, and the continuous non-differentiable covariance $\min(x, y)$ - LEOR successfully captures the underlying continuous mappings.

The bootstrap error distributions for these three kernels demonstrate highly stable predictions with remarkably tight standard deviations. Because LEOR solves the operator mapping as a global, regularized linear system in the latent space, it successfully filters out grid-level noise and respects the continuous topology of the operator. Furthermore, by bypassing backpropagation, LEOR achieves this high predictive accuracy in a fraction of the computational time required by standard deep learning architectures.

4.3.3 LEOR: Limitations and the Green's Function

While LEOR performs exceptionally well on standard analytical kernels, pushing the model to approximate the 1D Poisson equation reveals the fundamental limitation of a purely data-driven latent approach.

Unlike the previous benchmarks, the integral kernel corresponding to the 1D Poisson boundary value problem acts as a physical Green's function. It possesses strict structural constraints: its spatial derivative exhibits a sharp jump discontinuity along the diagonal $x = y$, and it must strictly evaluate to zero at the boundaries ($x = 0$ and $x = 1$). Furthermore, as established in Section 4.2.2, the input functions for this dataset were generated using dense sums of random sines and cosines, injecting highly oscillatory harmonic frequencies into the system.

When tested against this rigorous physical benchmark, LEOR's predictive stability completely collapses. The bootstrap evaluation reveals that the relative L^2 error on the Poisson dataset spikes to a mean of approximately **0.05**, accompanied by a massive standard deviation of **0.3**.This extreme variance indicates that while LEOR can occasionally guess the correct output for highly specific input trajectories, it is violently sensitive to high-frequency harmonic inputs and frequently predicts entirely non-physical mappings. Also another thing to notice is that a crucial role are the choices of base functions upon which we expand our estimated functions.

4.4 Direct Kernel Approximation

While Latent Extreme Operator Regression (LEOR) successfully captures functional mappings in the latent space, it lacks the structural capacity to explicitly reconstruct the physical kernel surface. To solve this, we deploy Direct Kernel Approximation , parameterizing the integral kernel as the bilinear cross-product of randomized spatial ELMs.

Because the reconstruction of the operator from finite data is fundamentally ill-posed, relying solely on empirical data loss leads to an infinite null space of non-physical solutions. To systematically investigate how structural constraints collapse this null space, we performed a comprehensive ablation study, sequentially applying diverse regularization strategies and matching them with specialized numerical solvers.

4.4.1 Optimization Algorithms and Penalty Formulations

The complexity of the regularization terms dictates the required optimization algorithm. For standard L^2 (Ridge) regularization, the loss landscape is strictly convex and differentiable, allowing for an exact, closed-form solution via *Eigen-decomposition*. However, to enforce sparsity, L^1 penalties (Lasso and Elastic Net) were introduced. Due to the non-differentiability of the L^1 norm at zero, these models were optimized using the **Fast Iterative Shrinkage-Thresholding Algorithm (FISTA)**. Finally, for highly composite loss functions involving multiple structural soft constraints, the system was solved iteratively using the **Conjugate Gradient (CG)** method.

The investigation began with the exponential decay kernel $k(x, y) = e^{-|x-y|}$ and progressed through the following sequential regularization:

Bounding the Operator Standard regularization were applied to bound the parameter matrix Θ. This included Ridge (Eigen-decomposition), Lasso (FISTA), and Elastic Net (FISTA). These penalties successfully bounded the operator norm. However due the density of the base functions Lasso and Elastic Net, under-performed. One reason is the following: Empirically the Lasso term under-performs Ridge when the 'variables' are highly correlated. This is what happens with the cross-product of randomized base functions. Due their randomization and their non-linearity, these show a high level collinearity thus Lasso and Elastic Net under perform.

$$\min_{\Theta} \frac{1}{2} \left\| \Phi^T \Theta (\Psi v \cdot \Delta y) - u \right\|_F^2 + \alpha \|\Theta\|_F^2$$

$$\min_{\Theta} \frac{1}{2} \left\| \Phi^T \Theta (\Psi v \cdot \Delta y) - u \right\|_F^2 + \alpha \|\Theta\|_1$$

$$\min_{\Theta} \frac{1}{2} \left\| \Phi^T \Theta (\Psi v \cdot \Delta y) - u \right\|_F^2 + \alpha_1 \|\Theta\|_1 + \frac{1}{2} \alpha_2 \|\Theta\|_F^2$$

Enforcing Physical Symmetry Because diffusion kernels are mathematically self-adjoint, the estimated kernel must be symmetric. We enforced a soft symmetry penalty on the latent parameter matrix Θ via CG optimization:

$$\min_{\Theta} \frac{1}{2} \left\| \Phi^T \Theta \Phi v \cdot \Delta y - u \right\|_F^2 + \frac{\alpha_1}{2} \|\Theta\|_F^2 + \frac{\alpha_2}{2} \|\Theta - \Theta^T\|_F^2$$

This formulation was also successfully combined with Lasso (L^1 + Symmetry) utilizing FISTA.

Enforcing Spatial Coherence To ensure the learned kernel surface remained physically smooth, a smoothing penalty was applied directly to the parameters. The intuition relies on the basis functions Φ being continuous; if the coefficient matrix Θ is constrained such that parameters are similar to their neighbors, the resulting 2D functional expansion is guaranteed to act as a low-pass filter without high-frequency artifacts.

Boundary Conditions (Green's Function Priors) For systems representing specific differential boundaries, the kernel must explicitly evaluate to zero at the spatial limits. This was formulated as an additional penalty term minimized via CG:

$$\min_{\Theta} \frac{1}{2} \left\| \Phi^T \Theta \Phi v \cdot \Delta y - u \right\|_F^2 + \frac{\alpha_1}{2} \|\Theta\|_F^2 + \frac{\alpha_2}{2} \|\Theta - \Theta^T\|_F^2 + \frac{\alpha_3}{2} \left(\|v_0^T \Theta \Phi\|_F^2 + \|v_1^T \Theta \Phi\|_F^2 \right)$$

The Oracle Prior (True Kernel Guiding) As a theoretical benchmark to measure the maximum expressivity of the architecture, an "oracle" penalty was introduced. This explicitly penalized deviations from the true known kernel, acting as a structural upper bound for the model's capabilities:

$$\min_{\Theta} \cdots + \frac{\alpha_3}{2} \|\Phi^T \Theta \Phi - K_{true}\|_F^2$$

4.4.2 Predictive and Structural Evaluation

For every combination of regularization and solver, the model's performance was evaluated through rigorous bootstrapping. Crucially, the evaluation was split into two distinct categories:

- **Predictive Accuracy**: How well the model mapped novel inputs to outputs (u_{pred}), measured via Relative L^2 error and MSE.
- **Structural Accuracy**: How accurately the model reconstructed the exact physical surface of the kernel (K_{est} vs K_{true}), measured via Frobenius norm deviations.

4.4.3 Spectral Approximation and Pseudo-Spectrum Divergence

The ultimate metric of an operator network is its ability to learn the underlying spectrum of the continuous integral equation. To evaluate this, we extracted the eigendecomposition of the fully regularized estimated kernel K_{est} and plotted its eigenvalue decay curve against the exact analytical spectrum of K_{true}.

The spectral plots yield another mathematical insight. For the leading, dominant eigenvalues, the learned network aligns near-perfectly with the true spectrum. However, as the eigenvalue index n increases, the true eigenvalues decay rapidly toward zero.

At a specific threshold, the estimated spectrum abruptly diverges from the true decay curve, entering what is known as the *pseudo-spectrum*. At this point, the eigenvalues of the neural approximation are no longer capturing true physical modes.

This divergence is heavily dependent on the analytical nature of the true kernel. For example, the polynomial kernel benchmark ($x^2 - y^2$) represents a strictly finite-rank operator, meaning its true eigenvalues decay to zero almost instantaneously. Consequently, the neural estimation of the polynomial eigenvalues breaks down extremely early in the expansion, confirming the spectral theory of ill-posedness established in Chapter 2.

4.5 Deep Operator Networks (DeepONets)

Moving on while previous frameworks show some potential, in the field of Operator Learning literature is dominated by DeepONets [7]. Thus we benchmark our methodologies against this model.

We have to note here that this is a *bounded* attempt. That is because compared to the randomized, closed form and convex optimization path of ELM-based models, DeepONets rely heavily on backpropagation. It becomes difficult to perform a comparative study since the architecture complexity, optimization instability and huge hyperparameter space are part of it. Consequently, and for the purposes of this thesis, our experiments were restricted to the exponential decay kernel, $k(x, y) = e^{-|x-y|}$.

4.5.1 Architextures and Parameter Regimes

To assess the deep network's behavior, we defined 4 distinct models, varying activation functions, regularization strategies and physical constraints:

- **Model 1 (Baseline):** Hyperbolic tangent ($\tanh$) activations for both the branch and trunk networks, utilizing standard Ridge (L_2) regularization on both components.
- **Model 2 (Mixed Regularization):** $\tanh$ activations, maintaining Ridge (L_2) on the trunk, but applying Lasso (L_1) regularization to the branch network.
- **Model 3 (Harmonic Activations):** Sine activations ($\sin$) to better capture frequency responses, with Ridge on the trunk and Lasso on the branch.
- **Model 4 (Physics-Informed):** Sine activations, Ridge on the trunk, Lasso on the branch, augmented by a custom physics-informed loss term explicitly enforcing boundary conditions.

For each of these four parent architectures, we tested three sub-models varying the number of hidden parameters. These were categorized into regimes based on the dimensionality of their hidden layers relative to the problem complexity:

- **Overdetermined:** Branch dimensions $[500, 325, 325, 325, P_{out}]$,
  Trunk dimensions $[1, 325, 325, 325, P_{out}]$
- **Well-Determined:** Branch dimensions $[500, 470, 470, 470, P_{out}]$,
  Trunk dimensions $[1, 470, 470, 470, P_{out}]$
- **+10% Determined:** Branch dimensions $[500, 490, 490, 490, P_{out}]$,
  Trunk dimensions $[1, 490, 490, 490, P_{out}]$

4.5.2 The Role of Branch Sparsity (L1 Regularization)

An empirical step was taken for MOdels 2,3 and 4 by applying the L_1 regularization to the branch network. That is due to the idea that the branch network processes discrete sensor points, by applying the L_1 regularization it somehow forces the network to choose an essential or minimal subset of sensors to characterize the operator. Also, that this sparsity improves data and computational efficiency.

4.5.3 Results: Predictive Accuracy vs. Kernel Reconstruction

The models after being trained were evaluated using OOB bootstrapp methodology. Results can be seen in the Appendix. In terms of predictive mapping of the functional data input, the results looked excellent. The DeepONets achieved highly competitive Mean Squared Error (MSE) on the unseen test data.

However, when attempted to reconstruct the kernel, the results were not that satisfying. Because the DeepONet is not natively a kernel reconstractor like in our Direct Kernel Approximation, we attempted to extract the learned kernel by passing a Gaussian bell curve approximation of the Dirac delta function ($\delta(x - y)$) through the trained network. This extraction method revealed that the DeepONets struggled to accurately reconstruct the structure of the kernel surface, resulting in a higher relative error in structural accuracy compared to our ELM-based methods.

This failure in exact kernel reconstruction does not imply that the DeepONet failed as a learning model. Instead, it serves as a powerful empirical demonstration of the mathematical ill-posedness inherent in these inverse problems. Because the mapping is learned from finite data, the network estimates a kernel of the form $K_{est} = K_{true} + K_{null}$, where K_{null} could be

an arbitrary component residing within the operator's null space. The DeepONet successfully optimized the predictive loss but without the strict physical and structural priors utilized in our previous methods, it could not collapse the infinite null space to isolate the single, true physical kernel.

To conclude, DeepONets are undeniably powerful functional approximators, but they are to be treated as an independent study for effective experimentation. The combinatorial explosion of decision-making | choosing the optimal number of parameters, selecting appropriate activation functions, balancing single versus mixed regularizations, and tuning physics-informed loss weights | makes them highly sensitive.

Chapter 5

Conclusions and Future Research

5.1 Conclusions

Operator Learning represents a new step of computational mathematics which provides a data-driven approach when trying to solve inverse problems of continoous operators. This thesis investigated randomized neural network architectures, the Extreme Learning Machines (ELMs), demonstrating that expensive and iterative numerical solves could be overcome by linear least-squares optimization, somethimes even convex.

A key insight of this research is that operator learning is heavily dependent on the nature and the spectrum of the available training data. Whether data are abundant or scarce, and whether they consist strictly of low-frequency smooth trends, high-frequency harmonic oscillations, or a hybrid of both, it highlights specific parts of the properties of operator, *what data stress what properties*. High-frequency data highlight spatial *gradient changes* and *sharp diagonal* behaviors, while low-frequency inputs isolate overall energy transport and *dominant spectral modes*. To note that, despite input data variations, the inherent compactness of Fredholm integral operators acts as low-pass filters and they eventually smooth out the noises while still preserving essential properties of the underlying physics.

Each model architecture investigated in this study exhibited distinct strengths and trade-offs:

- **Naive ELM**: Demonstrated a total collapse in predictive generalization, serving as proof that operating on discrete spatial grids without projecting into continuous function spaces cannot resolve ill-posed inverse problems.
- **Latent Extreme Operator Regression (LEOR)**: Achieved exceptional speed and accuracy on smooth analytical kernels, but its predictive stability collapsed when applied to physical Green's functions with strict boundary conditions and high-frequency harmonic inputs.

- **Direct Kernel Approximation**: Successfully reconstructed physical kernel surfaces and preserved spectral decay properties, though its performance relied heavily on appropriate L^2-based regularizers, as L^1 sparsity methods struggled under the severe collinearity of randomized feature bases.

- **DeepONets**: Matched the predictive mapping accuracy of randomized models for output trajectories, but struggled to accurately isolate the true bivariate kernel surface due to the infinite null space inherent in deep, unconstrained architectures.

5.2 Future Research

This work highlights the efficiency and structural interests of the randomized operator learning, thus leading to several directions for future research:

- **Extended DeepONet and RandONet Experiments**: Due to the massive hyperparameter space and computational overhead of backpropagation, DeepONet experiments in this thesis were limited. An extension would involve broader architecture studies across several more kernels while also comparing them to the randomized version the RandONets. Faithfully, maybe understanding the mapping between hyperparameter spaces as well.

- **Advanced Regularization in Deep Architectures**: The physics-informed, symmetry, and Boundary Conditions regularizers developed for the Direct Kernel Approximation model could be extended to the loss functions of DeepONets and RandONets to better collapse their non-physical null spaces.

- **Application to Volterra Processes and Stochastic Calculus**: An exciting theoretical extension is applying these operator frameworks to Volterra integral equations in stochastic calculus. Learning Volterra operators could enable the empirical reconstruction of non-stationary covariance functions and memory-dependent kernels directly from real-world financial or physical time-series data.

- **Bayesian Formulations for Uncertainty Quantification**: Incorporating Bayesian frameworks into Direct Kernel Approximation and DeepONets would transform deterministic predictions into probabilistic landscapes. Generating posterior distributions over learned parameters and kernel surfaces would allow for quantified uncertainty bounds, which are critical when solving real-world, highly unstable inverse problems.

Chapter 6

Appendix

6.1 Results from Naive ELM

Table 6.1: Performance Metrics for Naive ELM

Kernel	**Metric**	**Mean**	**Std**
Exponential Decay	MSE	8.21e-03	9.72e-04
	Rel L1	41%	4.97%
	Rel L2	38.7%	4.40%
	Rel L-inf	34%	3.33%
Minimum	MSE	2.80e-03	3.28e-04
	Rel L1	40.1%	4.37%
	Rel L2	38.7%	4.08%
	Rel L-inf	36.5%	3.57%
Polynomial	MSE	2.14e-03	1.87e-04
	Rel L1	37.3%	2.96%
	Rel L2	37%	2.91%
	Rel L-inf	38%	3.11%
1D Poisson's	MSE	2.50e-01	1.72e-02
	Rel L1	97.2%	6.13%
	Rel L2	93.3%	5.44%
	Rel L-inf	89.8%	4.47%

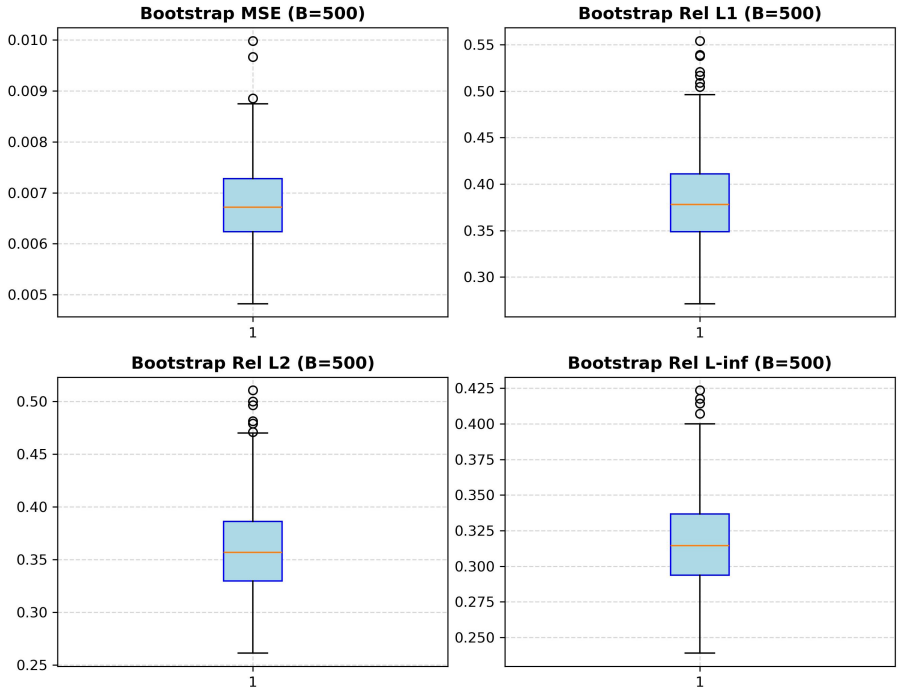

Figure 6.1: Bootstrapp Boxplots for NaiveELM evaluated at Exponential Decay kernel

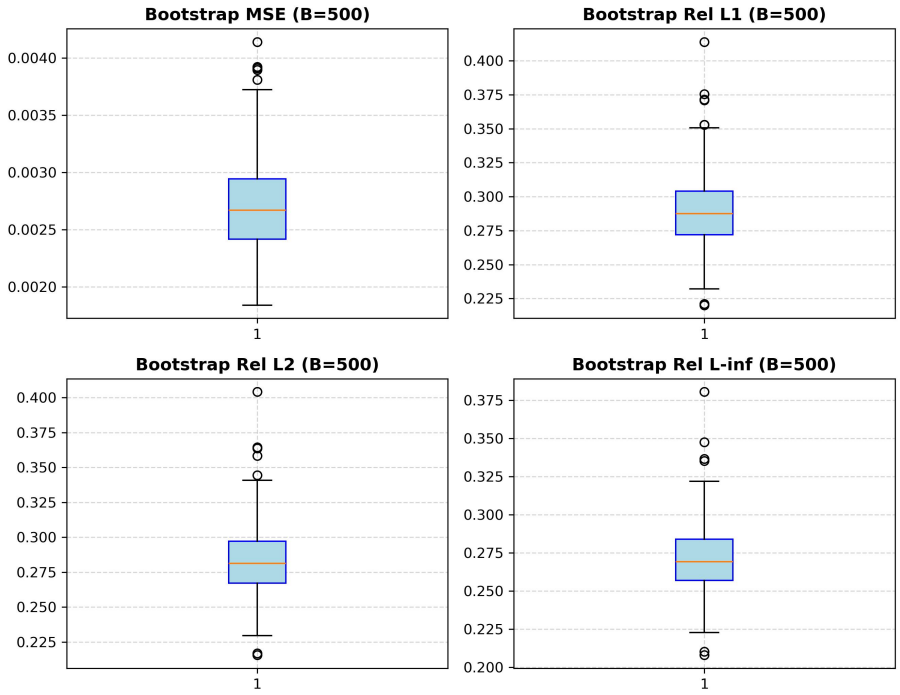

Figure 6.2: Bootstrapp Boxplots for NaiveELM evaluated at Minimum $\min()$ kernel

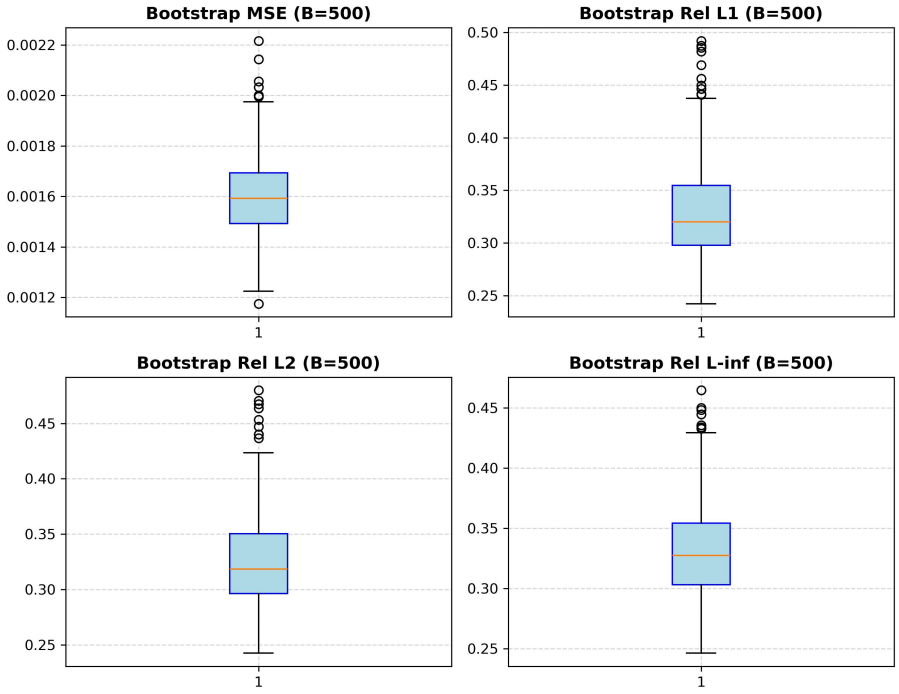

Figure 6.3: Bootstrapp Boxplots for NaiveELM evaluated at Polynomial kernel

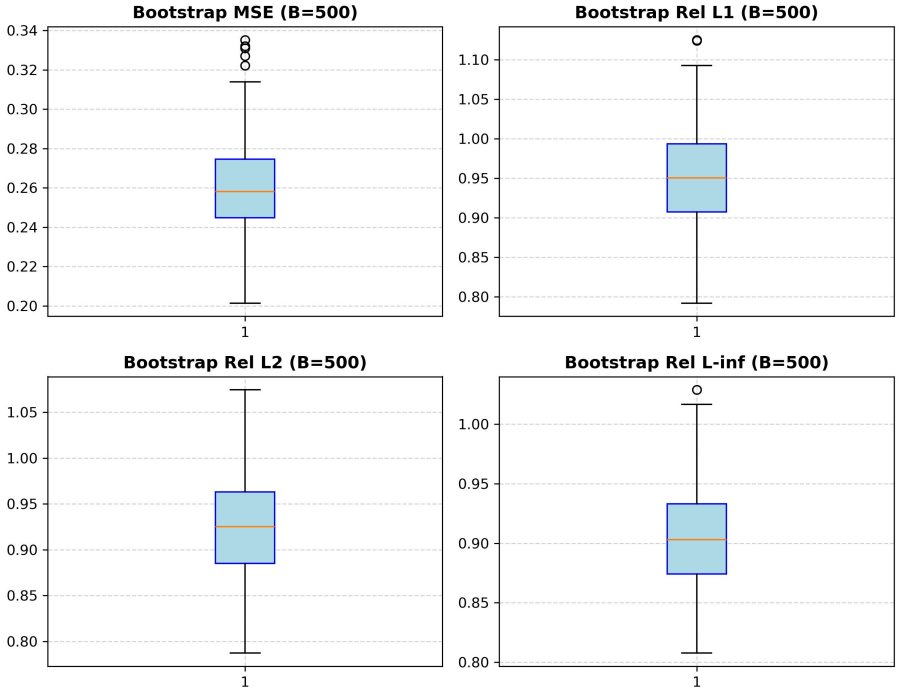

Figure 6.4: Bootstrapp Boxplots for NaiveELM evaluated at 1D Poisson's kernel

6.2 Results from LEOR

Table 6.2: Performance Metrics for Model B (Relative Errors in Percentages)

Problem	**Metric**	**Mean**	**Std Dev**	**95% Boot CI**
Problem 1	MSE	4.37e-06	3.31e-06	[1.40e-06, 1.35e-05]
	Rel L1	0.82%	0.36%	[0.40%, 1.82%]
	Rel L2	0.78%	0.33%	[0.39%, 1.72%]
	Rel L-inf	0.79%	0.27%	[0.47%, 1.54%]
Problem 2	MSE	1.90e-06	4.62e-06	[4.78e-07, 7.77e-06]
	Rel L1	0.91%	0.64%	[0.41%, 2.69%]
	Rel L2	0.88%	0.60%	[0.41%, 2.56%]
	Rel L-inf	0.88%	0.53%	[0.46%, 2.40%]
Problem 3	MSE	1.62e-06	4.31e-06	[3.93e-07, 7.76e-06]
	Rel L1	0.70%	0.45%	[0.34%, 1.94%]
	Rel L2	0.73%	0.48%	[0.35%, 2.06%]
	Rel L-inf	0.92%	0.69%	[0.41%, 2.80%]
Problem 4	MSE	1.82e+00	5.35e+01	[1.18e-02, 8.19e-02]
	Rel L1	5.87%	27.18%	[3.23%, 8.37%]
	Rel L2	5.86%	30.54%	[3.23%, 8.12%]
	Rel L-inf	6.39%	43.04%	[3.58%, 7.95%]

6.3 Results from Kernel Reconstruction using Bi-ELM

6.3.1 Results from estimating $K(x, y) = \exp|x - y|$

Table 6.3: Out-of-bag bootstrap measures for kernel estimation using Ridge penalization.

Kernel Grid	**MSE**(u) $(\times 10^{-7})$	**Rel** $L_2(u)$	**MSE**(K) $(\times 10^{-5})$	**Rel** $L_2(K)$
$(50, 50)$	4.967 ± 0.134	0.002 ± 0.000	11.03	0.014
$(100, 100)$	$\mathbf{1.347} \pm 0.021$	0.001 ± 0.000	8.062	0.012
$(500, 500)$	2.107 ± 0.052	0.001 ± 0.000	5.965	**0.010**
$(999, 999)$	1.629 ± 0.027	0.001 ± 0.000	8.386	0.012

Table 6.4: Out-of-bag bootstrap measures for kernel estimation using Lasso penalization.

Kernel Grid	**MSE**(u) $(\times 10^{-5})$	**Rel** $L_2(u)$	**MSE**(K) $(\times 10^{-3})$	**Rel** $L_2(K)$
$(50, 50)$	1.363 ± 0.032	0.009 ± 0.000	1.343	0.049
$(100, 100)$	$\mathbf{1.110} \pm 0.024$	0.008 ± 0.000	1.220	$\mathbf{0.046}$
$(500, 500)$	1.617 ± 0.039	0.009 ± 0.000	1.468	0.051
$(999, 999)$	1.854 ± 0.045	0.010 ± 0.000	1.655	0.054

Table 6.5: Out-of-bag bootstrap measures for kernel estimation using Elastic Net penalization.

Kernel Grid	**MSE**(u) $(\times 10^{-5})$	**Rel** $L_2(u)$	**MSE**(K) $(\times 10^{-3})$	**Rel** $L_2(K)$
$(50, 50)$	1.363 ± 0.032	0.009 ± 0.000	1.343	0.049
$(100, 100)$	$\mathbf{1.110} \pm 0.024$	0.008 ± 0.000	1.220	$\mathbf{0.046}$
$(500, 500)$	1.617 ± 0.039	0.009 ± 0.000	1.468	0.051
$(999, 999)$	1.854 ± 0.045	0.010 ± 0.000	1.655	0.054

Table 6.6: Out-of-bag bootstrap measures for kernel estimation using Ridge penalization with Soft Symmetry.

Kernel Grid	**MSE**(u) $(\times 10^{-5})$	**Rel** $L_2(u)$	**MSE**(K) $(\times 10^{-3})$	**Rel** $L_2(K)$
50	1.071 ± 0.027	0.008 ± 0.000	1.053	$\mathbf{0.043}$
100	$\mathbf{0.806} \pm 0.016$	0.007 ± 0.000	1.026	$\mathbf{0.043}$
500	1.054 ± 0.024	0.008 ± 0.000	1.107	0.044
999	1.077 ± 0.024	0.008 ± 0.000	1.123	0.044

Table 6.7: Out-of-bag bootstrap measures for kernel estimation using Lasso penalization with Soft Symmetry.

Kernel Grid	**MSE**(u) $(\times 10^{-5})$	**Rel** $L_2(u)$	**MSE**(K) $(\times 10^{-3})$	**Rel** $L_2(K)$
50	1.521 ± 0.038	0.009 ± 0.000	1.288	0.048
100	$\mathbf{0.821} \pm 0.016$	0.007 ± 0.000	1.022	$\mathbf{0.042}$
500	1.619 ± 0.039	0.009 ± 0.000	1.473	0.051
999	1.594 ± 0.038	0.009 ± 0.000	1.438	0.050

Table 6.8: Out-of-bag bootstrap measures for kernel estimation using Ridge penalization with Laplacian regularization.

Kernel Grid	**MSE**(u) $(\times 10^{-5})$	**Rel** $L_2(u)$	**MSE**(K) $(\times 10^{-3})$	**Rel** $L_2(K)$
50	1.075 ± 0.028	0.009 ± 0.000	1.063	0.044
100	$\mathbf{0.810} \pm 0.015$	0.007 ± 0.000	1.03	**0.043**
500	1.059 ± 0.026	0.008 ± 0.000	1.11	0.044
999	1.082 ± 0.026	0.009 ± 0.000	1.123	0.044

Table 6.9: Out-of-bag bootstrap measures for kernel estimation using Ridge penalization with Soft Symmetry and Soft Physics.

Kernel Grid	**MSE**(u) $(\times 10^{-7})$	**Rel** $L_2(u)$	**MSE**(K) $(\times 10^{-4})$	**Rel** $L_2(K)$
50	5.457 ± 0.122	0.002 ± 0.000	1.673	0.017
100	$\mathbf{1.985} \pm 0.036$	0.001 ± 0.000	0.769	**0.012**
500	3.427 ± 0.069	0.001 ± 0.000	1.220	0.015
999	3.409 ± 0.070	0.001 ± 0.000	1.240	0.015

Table 6.10: Out-of-bag bootstrap measures for kernel estimation using Ridge penalization with Soft Symmetry and Soft Boundary Conditions.

Kernel Grid	**MSE**(u) $(\times 10^{-6})$	**Rel** $L_2(u)$	**MSE**(K) $(\times 10^{-4})$	**Rel** $L_2(K)$
50	2.276 ± 0.044	0.003 ± 0.000	3.981	0.026
100	1.349 ± 0.030	0.003 ± 0.000	**2.961**	**0.023**
500	1.460 ± 0.032	0.003 ± 0.000	3.064	**0.023**
999	$\mathbf{1.337} \pm 0.030$	0.003 ± 0.000	3.035	**0.023**

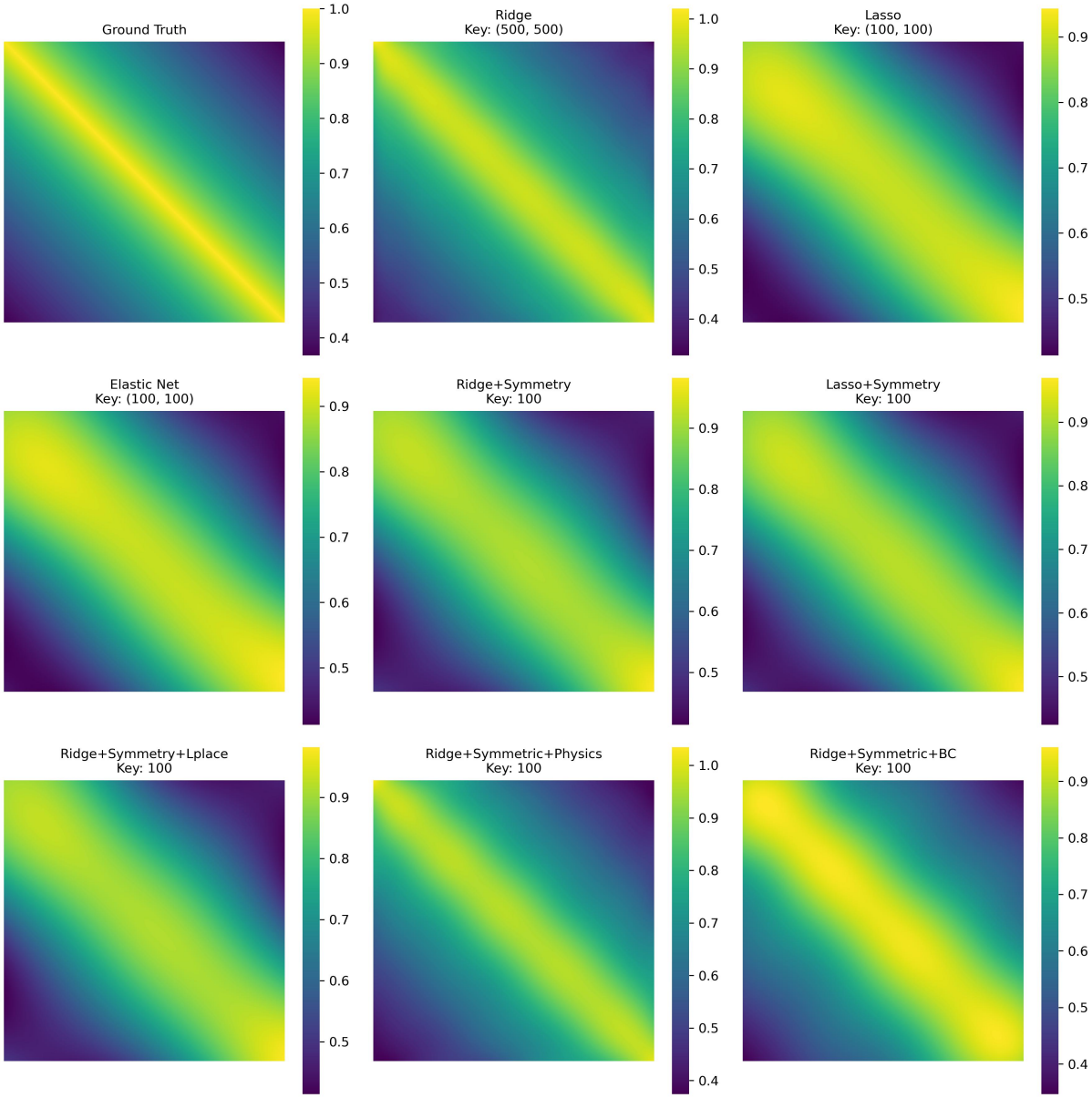

Figure 6.5: Comparison of estimated kernel functions $\hat{K}(x, y)$ across different penalization techniques against the true kernel $K(x, y)$. Panels display two-dimensional heatmaps over the spatial domain $[0, 1]^2$. Color intensity represents kernel value magnitude.

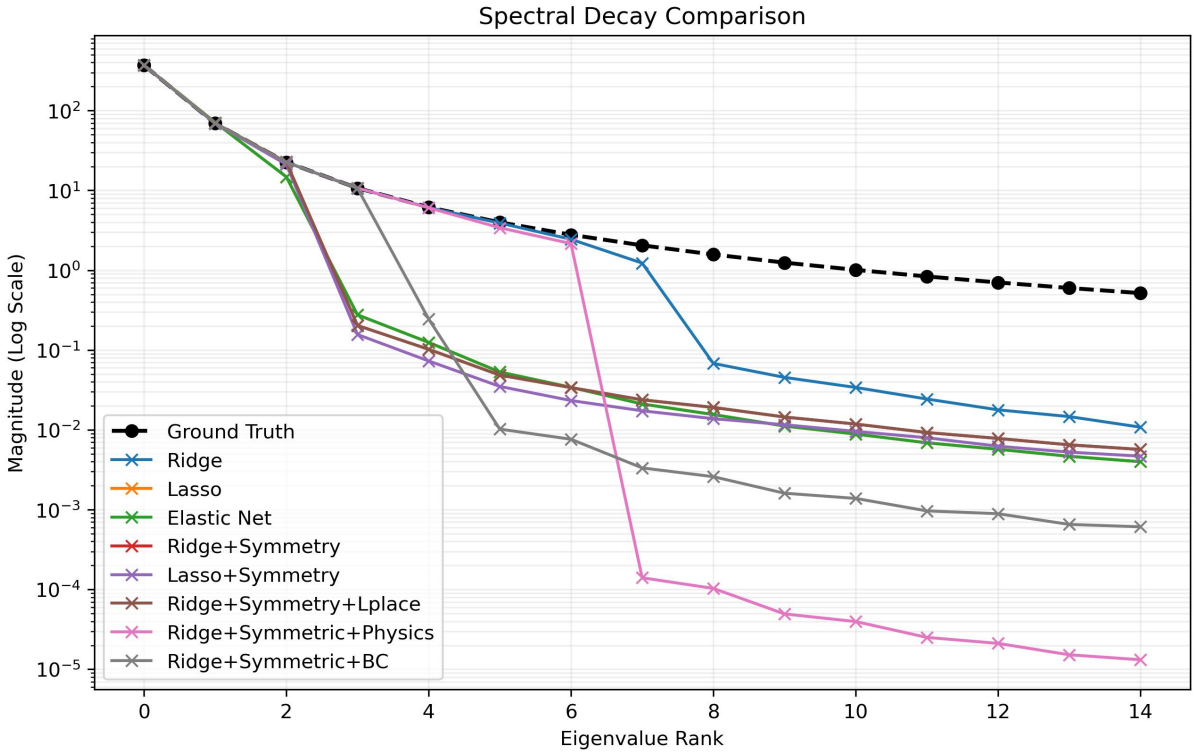

Figure 6.6: Estimated eigenvalue spectra $\hat{\lambda}_i$ across all estimation techniques compared against the true eigenvalue spectrum λ_i. The horizontal axis displays the eigenvalue rank $i = 1, \ldots, 15$, while the vertical axis shows the eigenvalue magnitude on a logarithmic scale.

Table 6.11: Relative error of estimated eigenvalues. Values with relative error $> 50\%$ (0.50) are dimmed in light gray to highlight valid estimates.

Rank	Elastic Net	Lasso	Lasso+Sym	Ridge	Ridge+Sym	Ridge+Sym+BC	Ridge+Sym+Phys	Ridge+Sym+Laplace
1	0.0028	0.0028	0.0001	0.0005	0.0004	0.0001	0.0000	0.0004
2	0.0271	0.0271	0.0009	0.0005	0.0074	0.0021	0.0000	0.0074
3	0.3453	0.3453	0.0637	0.0015	0.0022	0.0028	0.0001	0.0022
4	0.9742	0.9742	0.9854	0.0022	0.9810	0.0045	0.0019	0.9810
5	0.9797	0.9797	0.9881	0.0085	0.9834	0.9602	0.0168	0.9834
6	0.9867	0.9867	0.9912	0.0198	0.9878	0.9974	0.1446	0.9878
7	0.9877	0.9877	0.9916	0.1227	0.9878	0.9973	0.2243	0.9878
8	0.9897	0.9897	0.9915	0.4008	0.9884	0.9984	0.9999	0.9884
9	0.9901	0.9901	0.9912	0.9568	0.9879	0.9984	0.9999	0.9879
10	0.9910	0.9910	0.9907	0.9635	0.9884	0.9987	1.0000	0.9884
11	0.9912	0.9912	0.9905	0.9664	0.9883	0.9986	1.0000	0.9883
12	0.9918	0.9918	0.9905	0.9709	0.9889	0.9988	1.0000	0.9889
13	0.9919	0.9919	0.9911	0.9746	0.9889	0.9987	1.0000	0.9889
14	0.9922	0.9922	0.9912	0.9755	0.9892	0.9989	1.0000	0.9892
15	0.9923	0.9923	0.9909	0.9791	0.9890	0.9988	1.0000	0.9890

Table 6.12: Relative error of estimated eigenfunctions compared to true eigenfunctions. Values with relative error $> 50\%$ (0.50) are dimmed in light gray to highlight accurate predictions.

Rank	Elastic Net	Lasso	Lasso+Sym	Ridge	Ridge+Sym	Ridge+Sym+BC	Ridge+Sym+Phys	Ridge+Sym+Laplace
1	0.0079	0.0079	0.0037	0.0032	0.0052	0.0009	0.0004	0.0052
2	0.0121	0.0121	0.0218	0.0020	0.0310	0.0075	0.0002	0.0310
3	0.2799	0.2799	0.2331	0.0064	0.2345	0.0318	0.0053	0.2345
4	1.3645	1.3645	1.3247	0.0110	1.3249	0.1504	0.0252	1.3249
5	1.1379	1.1379	1.0336	0.0467	1.0157	0.3038	0.0905	1.0157
6	1.2656	1.2656	1.3939	0.1451	1.3459	1.1500	0.4389	1.3460
7	1.3660	1.3660	1.1295	0.5166	1.0783	1.1991	0.6981	1.0782
8	1.2998	1.2998	1.2674	0.9061	1.3141	1.3835	1.4069	1.3140
9	1.3501	1.3501	1.3644	1.2353	1.2076	1.3544	1.2939	1.2077
10	1.4052	1.4052	1.2035	1.3845	1.0901	1.1557	1.3442	1.0900
11	1.3666	1.3666	1.0996	1.1858	1.3328	1.1255	1.3699	1.3326
12	1.3579	1.3579	1.1434	1.2720	1.1528	1.3426	1.3779	1.1529
13	1.3662	1.3662	1.3827	1.1701	1.2219	1.3098	1.3800	1.2217
14	1.4059	1.4059	1.2290	1.3658	1.2511	1.2077	1.3808	1.2513
15	1.3691	1.3691	1.2339	1.4020	1.0948	1.2158	1.3609	1.0947

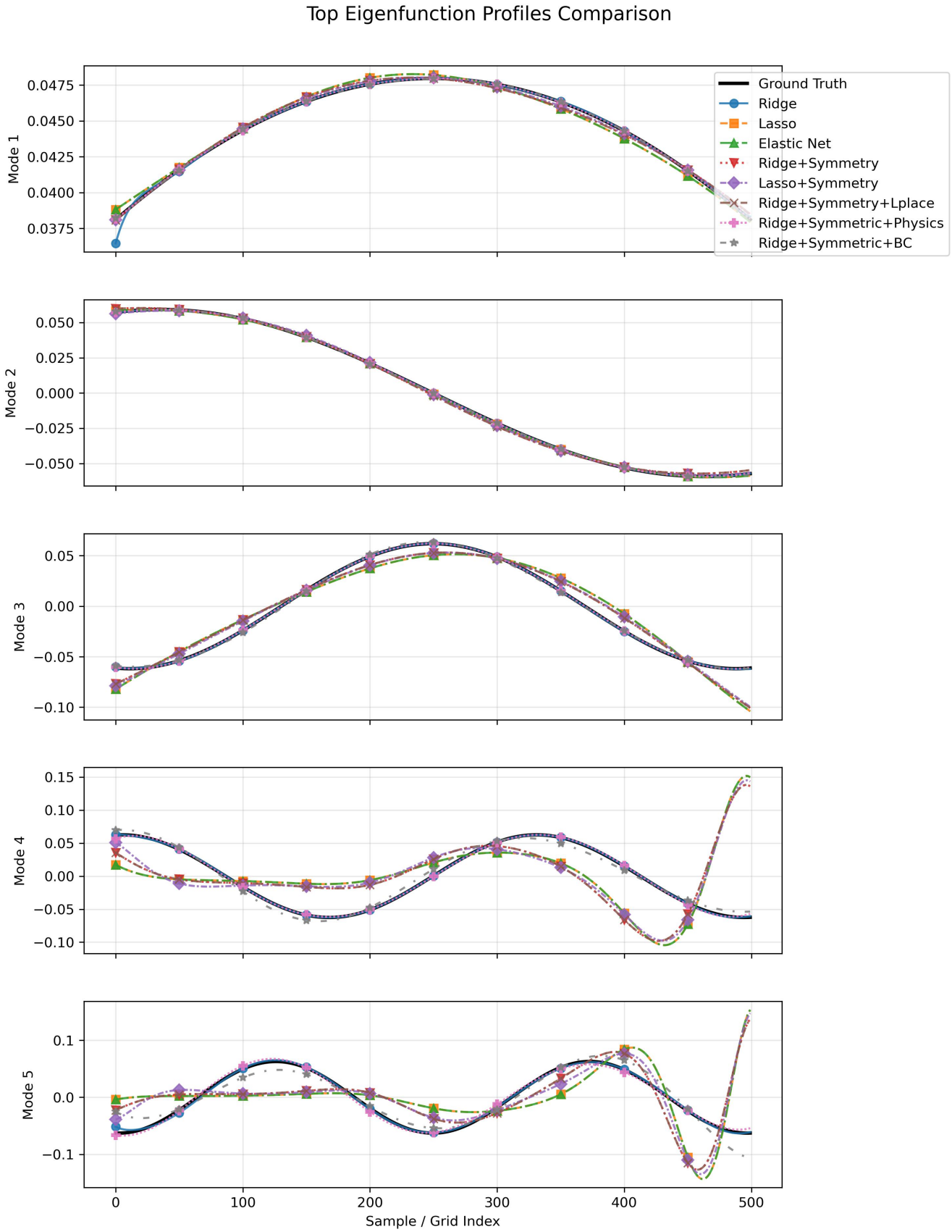

Figure 6.7: Spatial profiles of the first five estimated eigenfunctions $\hat{\phi}_1, \ldots, \hat{\phi}_5$ compared with the true eigenfunctions of the integral operator.

6.3.2 Results from estimating $K(x, y) = x^2 - y^2$

Table 6.13: Out-of-bag bootstrap measures for kernel estimation using Ridge penalization.

Kernel Grid	**MSE**(u) $(\times 10^{-16})$	**Rel** $L_2(u)$ $(\times 10^{-7})$	**MSE**(K) $(\times 10^{-13})$	**Rel** $L_2(K)$ $(\times 10^{-6})$
$(50, 50)$	4.249 ± 0.069	1.050 ± 0.017	3.917	1.481
$(100, 100)$	11.180 ± 0.292	1.703 ± 0.029	11.660	2.556
$(500, 500)$	**2.705** ± 0.047	0.838 ± 0.010	3.877	**1.474**
$(999, 999)$	55.290 ± 0.938	3.787 ± 0.042	56.470	5.624

Table 6.14: Out-of-bag bootstrap measures for kernel estimation using Lasso penalization.

Kernel Grid	**MSE**(u) $(\times 10^{-7})$	**Rel** $L_2(u)$ $(\times 10^{-3})$	**MSE**(K) $(\times 10^{-5})$	**Rel** $L_2(K)$ $(\times 10^{-2})$
$(50, 50)$	2.833 ± 0.064	2.711 ± 0.029	4.621	1.609
$(100, 100)$	1.344 ± 0.037	1.867 ± 0.034	3.890	1.476
$(500, 500)$	**1.139** ± 0.026	1.718 ± 0.012	1.341	**0.867**
$(999, 999)$	1.220 ± 0.027	1.779 ± 0.023	2.430	1.167

Table 6.15: Out-of-bag bootstrap measures for kernel estimation using Elastic Net penalization.

Kernel Grid	**MSE**(u) $(\times 10^{-7})$	**Rel** $L_2(u)$ $(\times 10^{-3})$	**MSE**(K) $(\times 10^{-5})$	**Rel** $L_2(K)$ $(\times 10^{-2})$
$(50, 50)$	2.833 ± 0.064	2.711 ± 0.029	4.621	1.609
$(100, 100)$	1.344 ± 0.037	1.867 ± 0.034	3.890	1.476
$(500, 500)$	**1.139** ± 0.026	1.718 ± 0.012	1.341	**0.867**
$(999, 999)$	1.220 ± 0.027	1.779 ± 0.023	2.430	1.167

Table 6.16: Out-of-bag bootstrap measures for kernel estimation using Ridge penalization with Soft Symmetry.

Kernel Grid	**MSE**(u) $(\times 10^{-8})$	**Rel** $L_2(u)$ $(\times 10^{-3})$	**MSE**(K) $(\times 10^{-6})$	**Rel** $L_2(K)$ $(\times 10^{-3})$
50	6.872 ± 0.184	1.335 ± 0.007	3.815	**4.623**
100	2.549 ± 0.051	0.813 ± 0.009	4.576	5.063
500	4.517 ± 0.094	1.082 ± 0.008	4.194	4.847
999	**1.232** ± 0.025	0.565 ± 0.009	3.833	4.633

Table 6.17: Out-of-bag bootstrap measures for kernel estimation using Lasso penalization with Soft Symmetry.

Kernel Grid	**MSE**(u) $(\times 10^{-7})$	**Rel** $L_2(u)$ $(\times 10^{-3})$	**MSE**(K) $(\times 10^{-5})$	**Rel** $L_2(K)$ $(\times 10^{-2})$
50	1.886 ± 0.046	2.211 ± 0.012	1.926	1.039
100	**0.276** ± 0.006	0.846 ± 0.006	0.306	**0.414**
500	1.245 ± 0.028	1.796 ± 0.015	1.774	0.997
999	1.056 ± 0.023	1.654 ± 0.019	1.826	1.011

Table 6.18: Out-of-bag bootstrap measures for kernel estimation using Ridge penalization with Soft Symmetry and Laplacian regularization.

Kernel Grid	**MSE**(u) $(\times 10^{-8})$	**Rel** $L_2(u)$ $(\times 10^{-3})$	**MSE**(K) $(\times 10^{-6})$	**Rel** $L_2(K)$ $(\times 10^{-3})$
50	6.850 ± 0.186	1.333 ± 0.007	3.876	4.659
100	2.548 ± 0.051	0.813 ± 0.009	4.563	5.056
500	2.951 ± 0.070	0.875 ± 0.007	4.794	5.182
999	**1.232** ± 0.025	0.565 ± 0.009	3.835	**4.634**

Table 6.19: Out-of-bag bootstrap measures for kernel estimation using Ridge penalization with Soft Symmetry and Soft Physics.

Kernel Grid	**MSE**(u) $(\times 10^{-10})$	**Rel** $L_2(u)$ $(\times 10^{-5})$	**MSE**(K) $(\times 10^{-9})$	**Rel** $L_2(K)$ $(\times 10^{-4})$
50	1.323 ± 0.040	5.857 ± 0.034	3.410	1.382
100	0.305 ± 0.010	2.812 ± 0.021	0.806	0.672
500	0.083 ± 0.003	1.464 ± 0.006	0.138	**0.278**
999	**0.034** ± 0.001	0.936 ± 0.001	0.138	**0.278**

Table 6.20: Out-of-bag bootstrap measures for kernel estimation using Ridge penalization with Soft Symmetry and Boundary Conditions.

Kernel Grid	**MSE**(u) $(\times 10^{-9})$	**Rel** $L_2(u)$ $(\times 10^{-4})$	**MSE**(K) $(\times 10^{-6})$	**Rel** $L_2(K)$ $(\times 10^{-3})$
50	1.927 ± 0.059	2.235 ± 0.046	1.372	2.772
100	0.116 ± 0.003	0.549 ± 0.010	0.079	0.664
500	$\mathbf{0.023} \pm 0.000$	0.245 ± 0.003	0.009	**0.225**
999	0.031 ± 0.000	0.285 ± 0.004	0.019	0.324

Table 6.21: Relative error of the top estimated eigenvalues compared to true eigenvalues. Values with relative error $> 50\%$ (0.50) are dimmed in light gray to highlight accurate predictions.

Rank	**Elastic Net**	**Lasso**	**Lasso+Sym**	**Ridge**	**Ridge+Sym**	**Ridge+Sym+BC**	**Ridge+Sym+Laplace**	**Ridge+Sym+Phys**
1	0.0030	0.0030	0.0007	0.0000	0.0015	0.0000	0.0014	0.0000
2	51.8162	51.8162	8.5524	0.0949	14.1281	1.9439	6.2234	0.3434
3	16.2581	16.2581	2.7024	0.0278	4.4219	0.6187	1.9562	0.0983
4	9.4779	9.4779	1.5515	0.0149	2.5343	0.3527	1.1107	0.0508
5	6.6205	6.6205	1.0576	0.0095	1.7283	0.2382	0.7469	0.0311
6	5.0469	5.0469	0.7828	0.0066	1.2805	0.1747	0.5444	0.0206
7	4.0511	4.0511	0.6081	0.0048	0.9954	0.1345	0.4158	0.0143
8	3.3645	3.3645	0.4876	0.0037	0.7983	0.1070	0.3274	0.0102
9	2.8626	2.8626	0.3999	0.0028	0.6540	0.0871	0.2633	0.0074
10	2.4799	2.4799	0.3335	0.0023	0.5441	0.0722	0.2150	0.0055
11	2.1786	2.1786	0.2818	0.0018	0.4578	0.0606	0.1776	0.0041
12	1.9354	1.9354	0.2405	0.0015	0.3885	0.0515	0.1480	0.0031
13	1.7350	1.7350	0.2071	0.0012	0.3318	0.0441	0.1242	0.0023
14	1.5672	1.5672	0.1796	0.0011	0.2846	0.0381	0.1049	0.0018
15	1.4247	1.4247	0.1567	0.0009	0.2450	0.0332	0.0890	0.0014

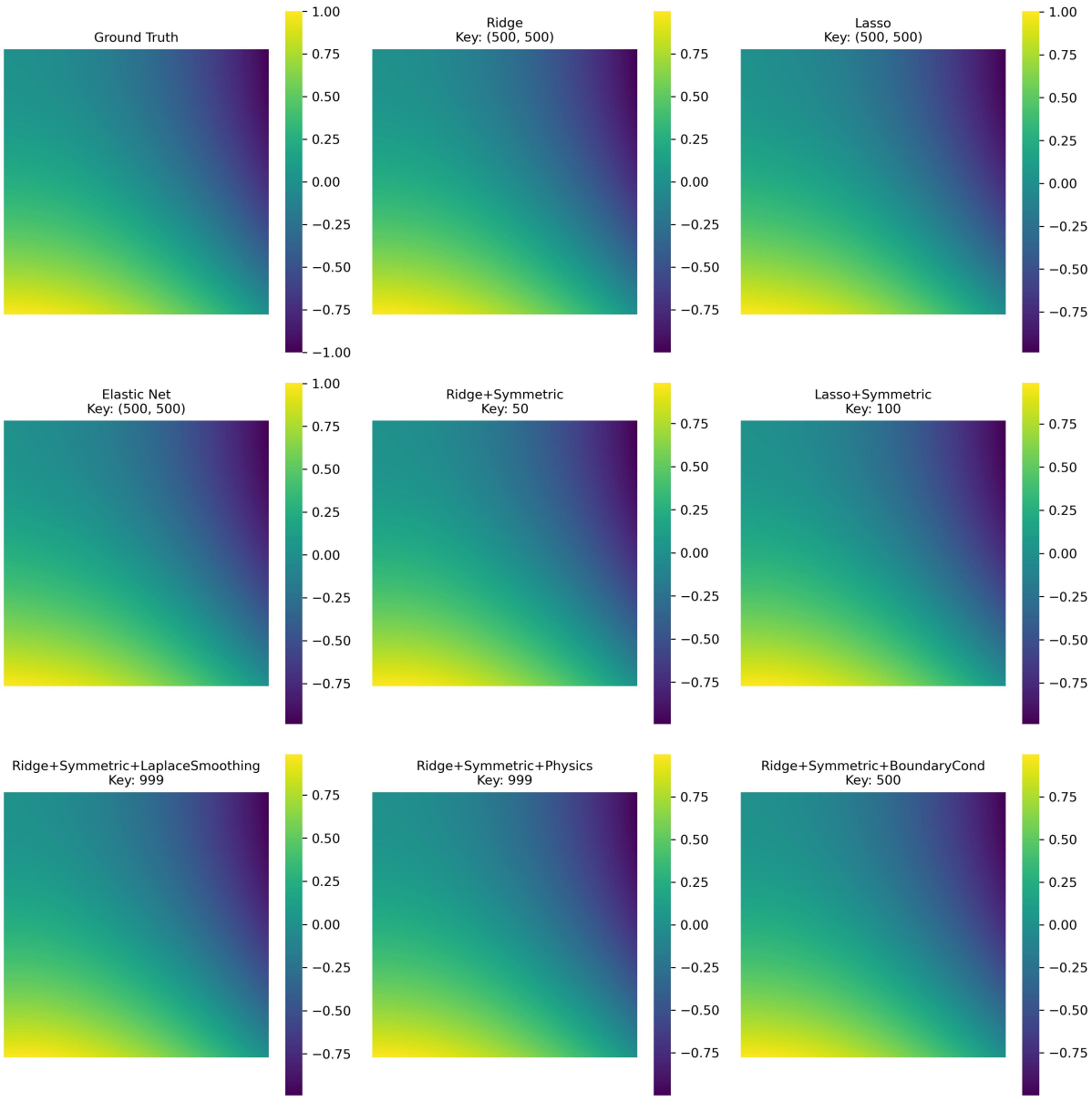

Figure 6.8: Comparison of estimated kernel functions $\hat{K}(x, y)$ across different penalization techniques against the true kernel $K(x, y)$. Panels display two-dimensional heatmaps over the spatial domain $[0, 1]^2$. Color intensity represents kernel value magnitude.

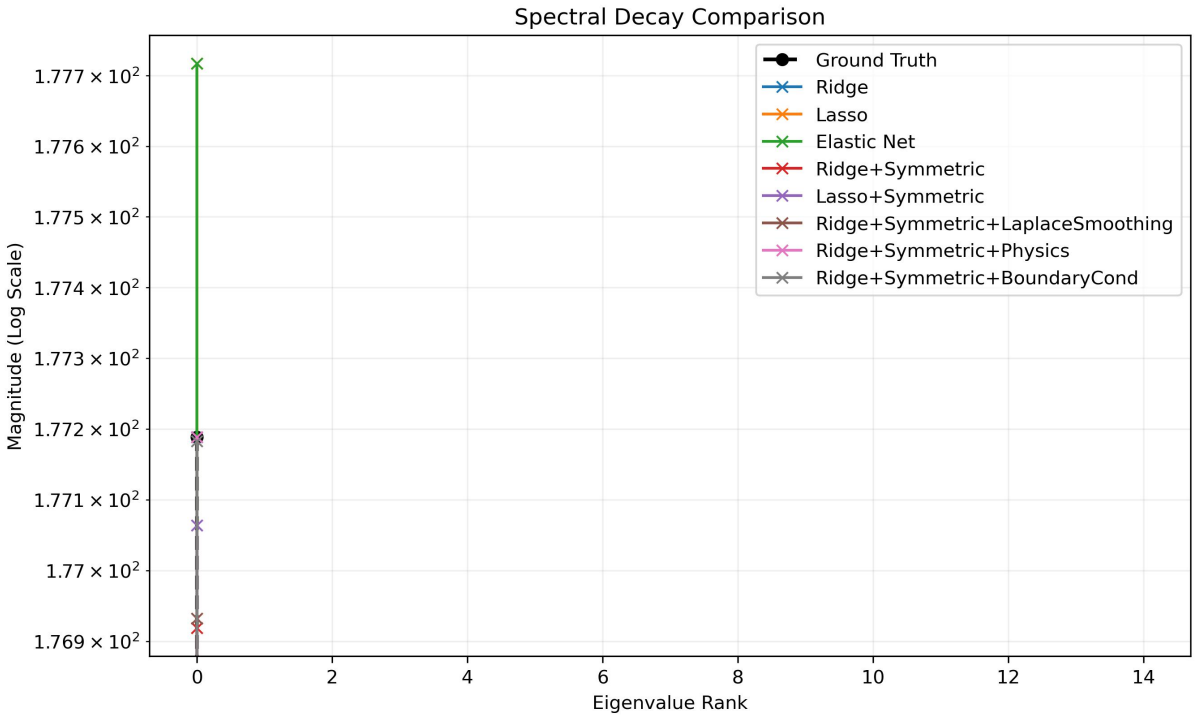

Figure 6.9: Estimated eigenvalue spectra $\hat{\lambda}_i$ across all estimation techniques compared against the true eigenvalue spectrum λ_i. The horizontal axis displays the eigenvalue rank $i = 1, \ldots, 15$, while the vertical axis shows the eigenvalue magnitude on a logarithmic scale.

Table 6.22: Relative error of the top estimated eigenfunctions compared to true eigenfunctions. Values with relative error $> 50\%$ (0.50) are dimmed in light gray to highlight accurate predictions.

Rank	Elastic Net	Lasso	Lasso+Sym	Ridge	Ridge+Sym	Ridge+Sym+BC	Ridge+Sym+Laplace	Ridge+Sym+Phys
1	0.0073	0.0073	0.0015	0.0000	0.0036	0.0001	0.0034	0.0000
2	1.2371	1.2371	0.7531	0.0168	0.9324	0.2844	0.6484	0.0610
3	1.1121	1.1121	1.2028	0.0237	1.3638	0.4304	1.0344	0.0871
4	1.3973	1.3973	1.3933	0.0254	1.1069	0.4937	1.2411	0.0938
5	1.1777	1.1777	1.2541	0.0261	1.0721	0.5301	1.3634	0.0964
6	1.3826	1.3826	1.1678	0.0263	1.1642	0.5533	1.3877	0.0968
7	1.2311	1.2311	1.1189	0.0262	1.2840	0.5685	1.3342	0.0961
8	1.2600	1.2600	1.0944	0.0260	1.3891	0.5784	1.2969	0.0944
9	1.3913	1.3913	1.0845	0.0256	1.3569	0.5845	1.2715	0.0920
10	1.2539	1.2539	1.0830	0.0251	1.2926	0.5877	1.2552	0.0892
11	1.2525	1.2525	1.0858	0.0245	1.2464	0.5885	1.2457	0.0860
12	1.3571	1.3571	1.0905	0.0238	1.2156	0.5875	1.2417	0.0825
13	1.3518	1.3518	1.0959	0.0231	1.1969	0.5849	1.2421	0.0788
14	1.2700	1.2700	1.1012	0.0224	1.1870	0.5811	1.2461	0.0750
15	1.2568	1.2568	1.1061	0.0217	1.1830	0.5762	1.2532	0.0711

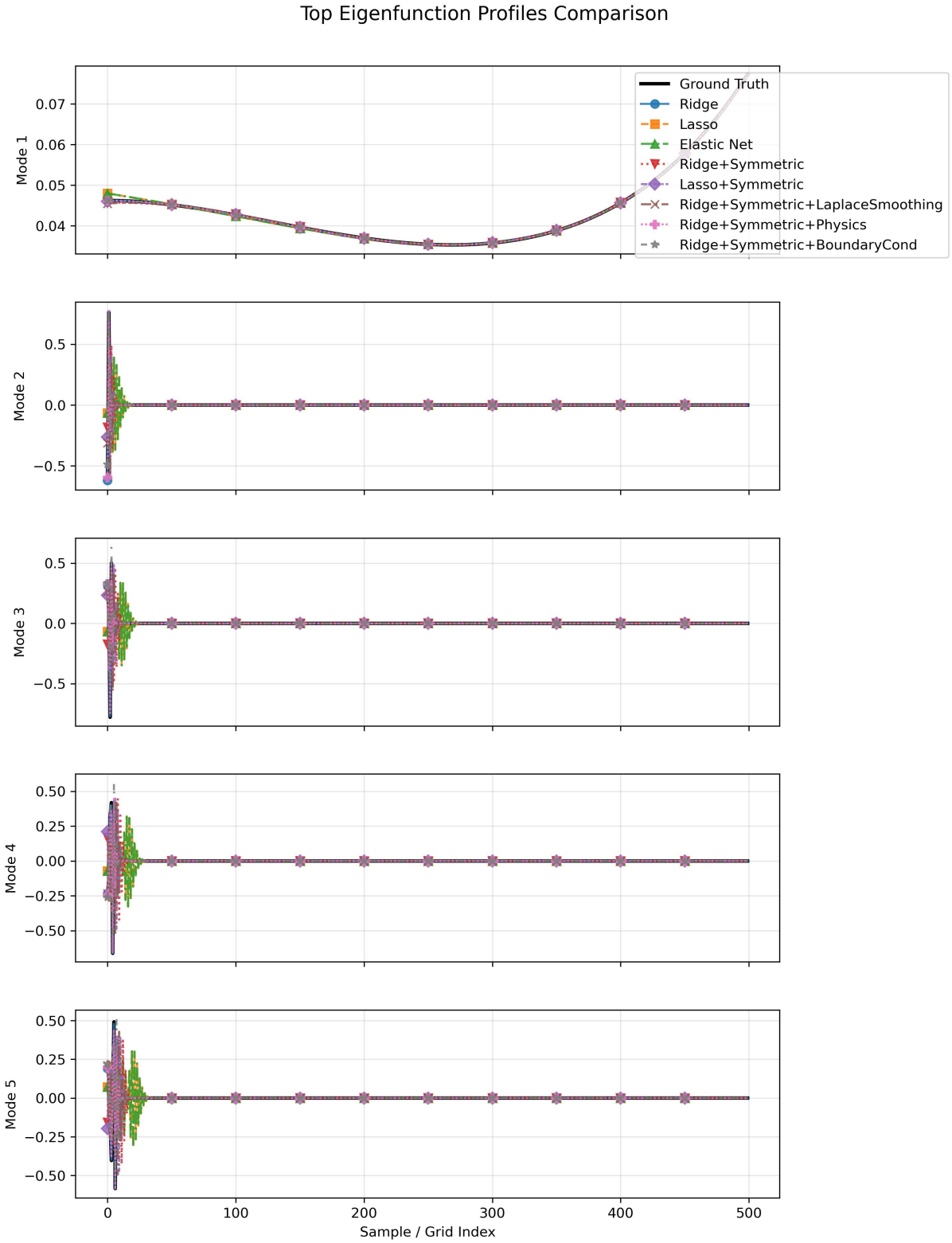

Figure 6.10: Spatial profiles of the first five estimated eigenfunctions $\hat{\phi}_1, \ldots, \hat{\phi}_5$ compared with the true eigenfunctions of the integral operator.

6.3.3 Results from estimating 1D Poisson's kernel $K(x, y) = \min(x, y)(1-\max(x, y))$

Table 6.23: Out-of-bag bootstrap measures for kernel estimation using Ridge penalization.

Kernel Grid	**MSE**(u) $(\times 10^{-6})$	**Rel** $L_2(u)$	**MSE**(K) $(\times 10^{-4})$	**Rel** $L_2(K)$
$(50, 50)$	7.491 ± 0.109	0.074 ± 0.001	3.677	0.182
$(100, 100)$	6.821 ± 0.091	0.070 ± 0.001	5.281	0.218
$(500, 500)$	$\mathbf{4.541} \pm 0.057$	0.057 ± 0.001	0.621	$\mathbf{0.075}$
$(999, 999)$	6.227 ± 0.083	0.067 ± 0.001	5.415	0.221

Table 6.24: Out-of-bag bootstrap measures for kernel estimation using Lasso penalization.

Kernel Grid	**MSE**(u) $(\times 10^{-5})$	**Rel** $L_2(u)$	**MSE**(K) $(\times 10^{-3})$	**Rel** $L_2(K)$
$(50, 50)$	8.793 ± 0.169	0.252 ± 0.004	5.502	0.705
$(100, 100)$	$\mathbf{5.826} \pm 0.100$	0.205 ± 0.003	5.317	0.693
$(500, 500)$	6.079 ± 0.111	0.209 ± 0.003	2.904	$\mathbf{0.512}$
$(999, 999)$	6.409 ± 0.117	0.215 ± 0.003	3.527	0.565

Table 6.25: Out-of-bag bootstrap measures for kernel estimation using Ridge penalization with Soft Symmetry.

Kernel Grid	**MSE**(u) $(\times 10^{-5})$	**Rel** $L_2(u)$	**MSE**(K) $(\times 10^{-5})$	**Rel** $L_2(K)$
50	2.518 ± 0.047	0.135 ± 0.002	5.166	0.068
100	$\mathbf{1.528} \pm 0.023$	0.105 ± 0.001	3.129	$\mathbf{0.053}$
500	1.536 ± 0.023	0.105 ± 0.001	3.373	0.055
999	1.642 ± 0.026	0.109 ± 0.001	5.985	0.074

Table 6.26: Out-of-bag bootstrap measures for kernel estimation using Ridge penalization with Soft Symmetry and Laplacian Regularization.

Kernel Grid	**MSE**(u) $(\times 10^{-5})$	**Rel** $L_2(u)$	**MSE**(K) $(\times 10^{-5})$	**Rel** $L_2(K)$
50	2.517 ± 0.047	0.135 ± 0.002	5.335	0.069
100	$\mathbf{1.528} \pm 0.023$	0.105 ± 0.001	3.894	**0.059**
500	1.563 ± 0.024	0.106 ± 0.001	9.807	0.094
999	1.635 ± 0.026	0.109 ± 0.001	41.260	0.193

Table 6.27: Out-of-bag bootstrap measures for kernel estimation using Ridge penalization with Soft Symmetry and Soft Physics.

Kernel Grid	**MSE**(u) $(\times 10^{-5})$	**Rel** $L_2(u)$	**MSE**(K) $(\times 10^{-5})$	**Rel** $L_2(K)$
100	$\mathbf{0.940} \pm 0.013$	0.082 ± 0.001	1.890	**0.041**
500	1.158 ± 0.016	0.091 ± 0.001	2.324	0.046
999	1.014 ± 0.014	0.086 ± 0.001	2.034	0.043

Table 6.28: Out-of-bag bootstrap measures for kernel estimation using Ridge penalization with Soft Symmetry and Boundary Conditions.

Kernel Grid	**MSE**(u) $(\times 10^{-5})$	**Rel** $L_2(u)$	**MSE**(K) $(\times 10^{-5})$	**Rel** $L_2(K)$
50	1.526 ± 0.024	0.105 ± 0.001	3.327	0.055
100	$\mathbf{0.939} \pm 0.013$	0.082 ± 0.001	2.288	**0.045**
500	1.027 ± 0.015	0.086 ± 0.001	2.864	0.051
999	0.951 ± 0.013	0.083 ± 0.001	6.361	0.076

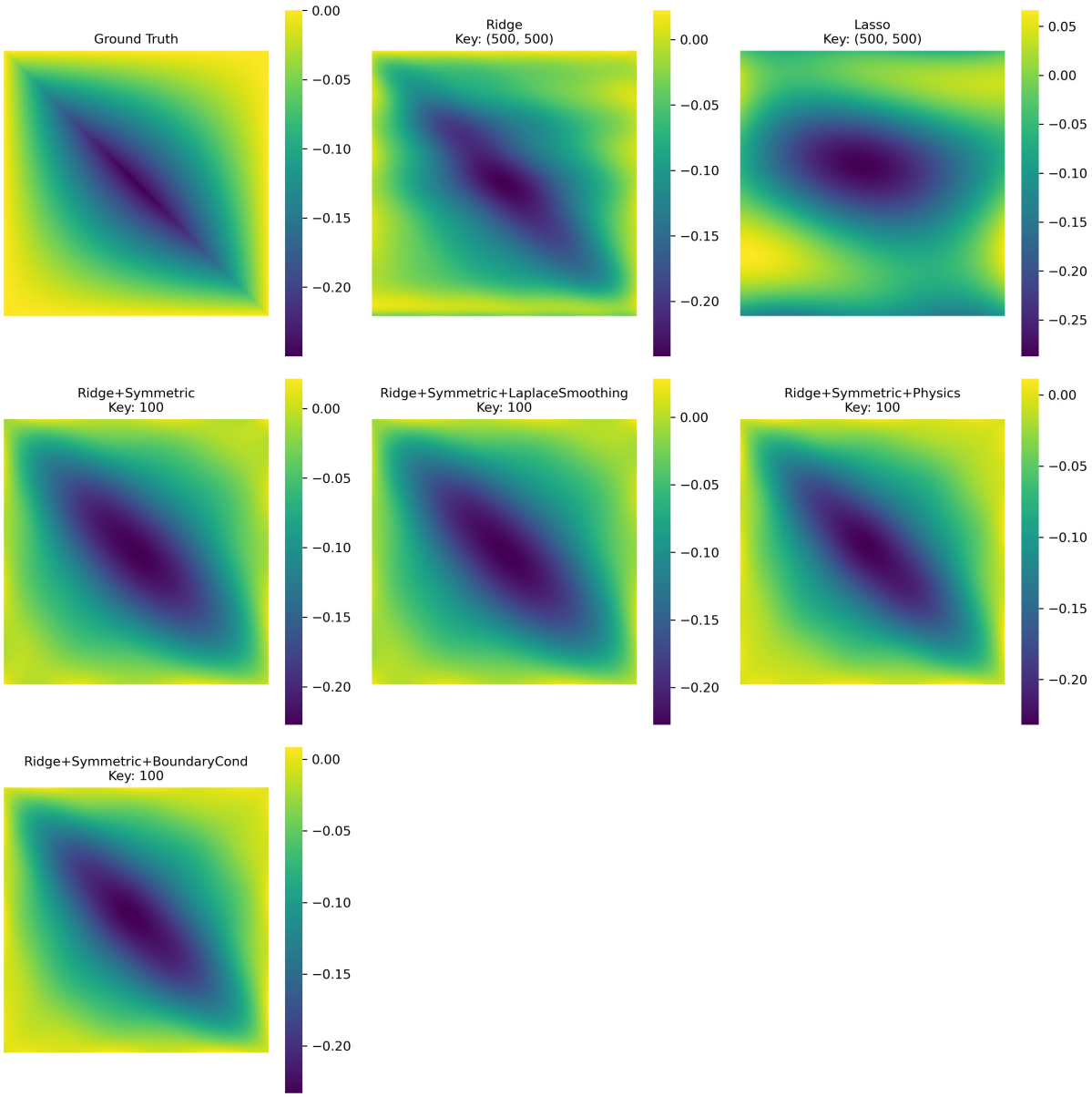

Figure 6.11: Comparison of estimated kernel functions $\hat{K}(x, y)$ across different penalization techniques against the true kernel $K(x, y)$. Panels display two-dimensional heatmaps over the spatial domain $[0, 1]^2$. Color intensity represents kernel value magnitude.

Table 6.29: Relative error of the top 10 estimated eigenvalues compared to true eigenvalues. Values with relative error $> 50\%$ (0.50) are dimmed in light gray to highlight accurate predictions.

Rank	Lasso	Ridge	Ridge+Sym	Ridge+Sym+BC	Ridge+Sym+Laplace	Ridge+Sym+Phys
1	0.0686	0.0031	0.0036	0.0126	0.0117	0.0001
2	0.3507	0.0136	0.0014	0.0109	0.0104	0.0003
3	0.7274	0.0147	0.0141	0.0208	0.0102	0.0001
4	0.5941	0.0968	0.0262	0.0047	0.0168	0.0070
5	0.3362	0.0793	0.3975	0.0075	0.4953	0.0151
6	0.3302	0.0283	0.6358	0.5695	0.6667	0.5175
7	0.4127	0.2224	0.6252	0.6914	0.5651	0.6485
8	0.3814	0.0790	0.9868	0.8285	0.9731	0.8197
9	0.2688	0.0457	0.9944	0.9294	0.9725	0.9362
10	0.1561	0.6660	0.9940	0.9938	0.9739	0.9969

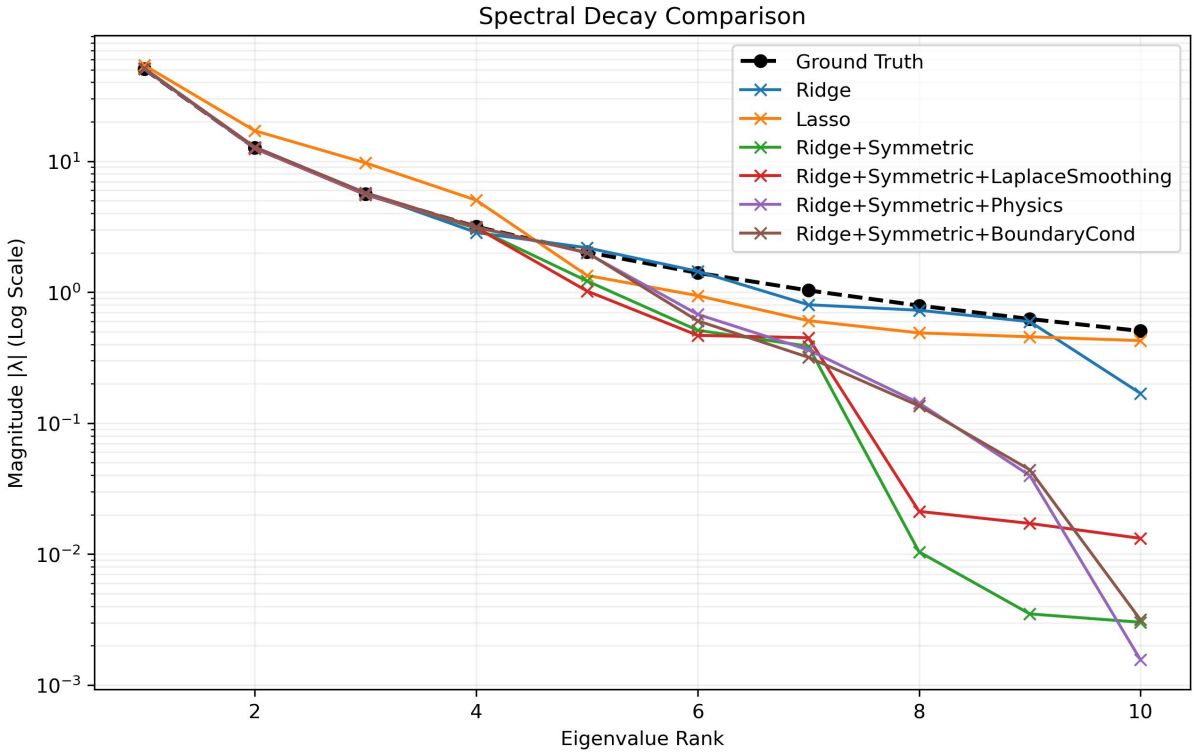

Figure 6.12: Estimated eigenvalue spectra $\hat{\lambda}_i$ across all estimation techniques compared against the true eigenvalue spectrum λ_i. The horizontal axis displays the eigenvalue rank $i = 1, \ldots, 15$, while the vertical axis shows the eigenvalue magnitude on a logarithmic scale.

Table 6.30: Relative error of the top 15 estimated eigenfunctions compared to true eigenfunctions. Values with relative error $> 50\%$ (0.50) are dimmed in light gray to highlight accurate predictions.

Rank	Lasso	Ridge	Ridge+Sym	Ridge+Sym+BC	Ridge+Sym+Laplace	Ridge+Sym+Phys
1	0.3949	0.0464	0.0046	0.0074	0.0172	0.0029
2	0.4193	0.0684	0.0111	0.0151	0.0142	0.0017
3	0.8601	0.1410	0.0513	0.0368	0.0711	0.0226
4	1.1310	0.1311	0.2481	0.1240	0.2395	0.1256
5	1.1946	0.4612	0.5380	0.3139	0.5120	0.3038
6	1.2783	0.5983	1.0352	0.7302	1.2904	0.7487
7	1.2210	0.9701	1.2432	0.9130	1.2995	0.9789
8	1.2162	1.1355	1.2334	1.3930	1.2432	1.4096
9	1.0620	1.3665	1.0534	1.3213	1.0629	1.2409
10	1.0865	1.3616	1.0255	1.3572	1.1523	1.4115
11	1.3818	1.2789	1.0706	1.3283	1.1166	1.3385
12	1.3160	1.3528	1.0702	1.1265	1.3370	1.0555
13	1.1175	1.3329	0.9723	1.3685	1.3929	1.1830
14	1.0015	1.2968	0.9761	1.1188	1.3914	1.1508
15	1.1034	1.1281	0.9758	1.0421	1.2709	0.9641

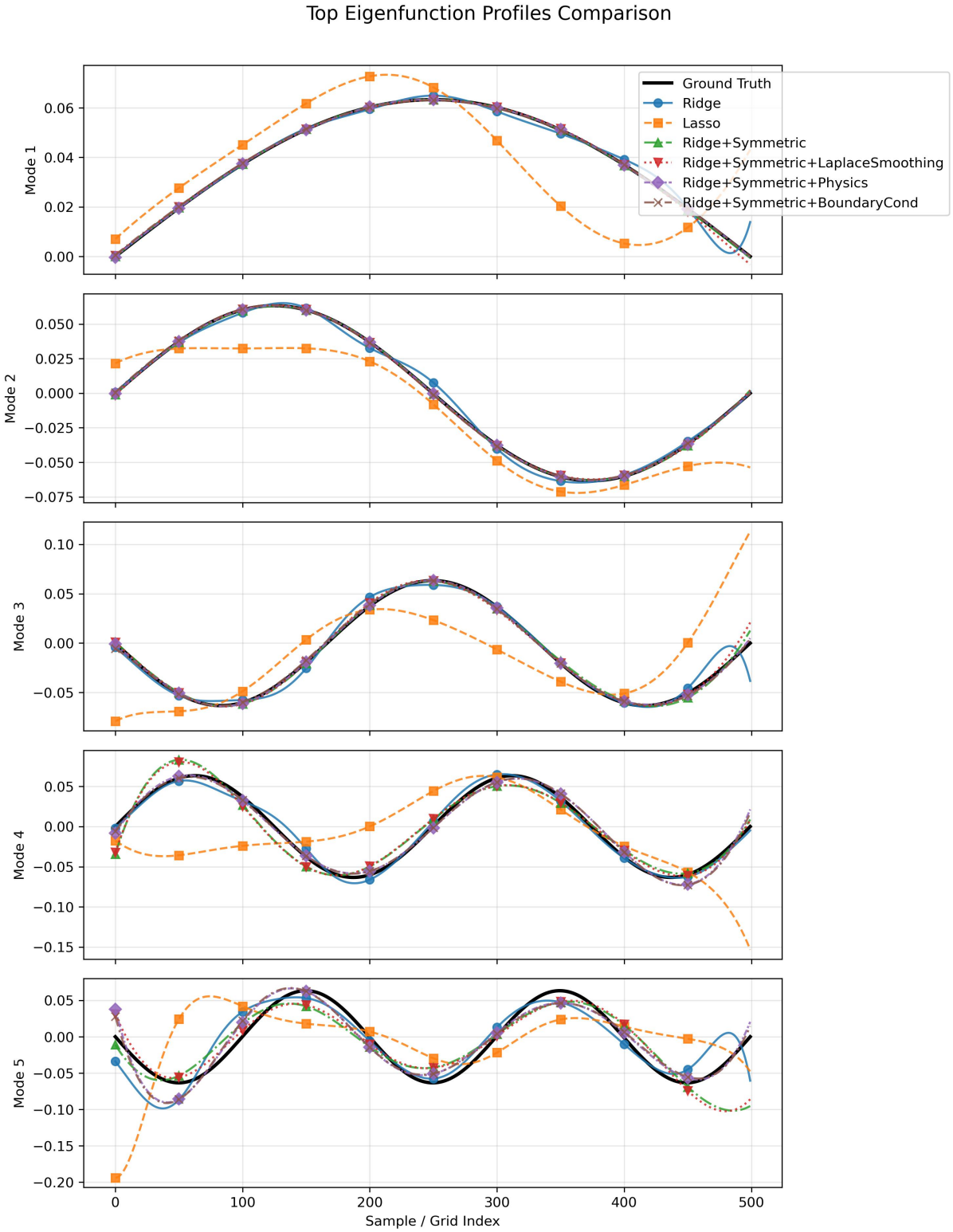

Figure 6.13: Spatial profiles of the first five estimated eigenfunctions $\hat{\phi}_1, \ldots, \hat{\phi}_5$ compared with the true eigenfunctions of the integral operator.

6.4 Results from DeepONets

Regime	**OOS MSE** $(\times 10^{-5})$	**OOS Rel** L_2	**Kernel MSE** $(\times 10^{-2})$	**Kernel Rel** L_2
Overdetermined	$2.34\ [1.94, 2.78]$	$0.0268\ [0.0242, 0.0295]$	3.504	0.2485
Well-Determined	$2.06\ [1.76, 2.44]$	$0.0248\ [0.0219, 0.0277]$	3.150	0.2357
+10% Determined	$2.14\ [1.71, 2.70]$	$0.0256\ [0.0226, 0.0302]$	2.961	0.2285

Table 6.31: Out-of-sample prediction and kernel estimation metrics across different system determination regimes (Mean $\pm$ 95% CI).

Model: Ridge on Trunk, Ridge on Branch and Tanh activation functions

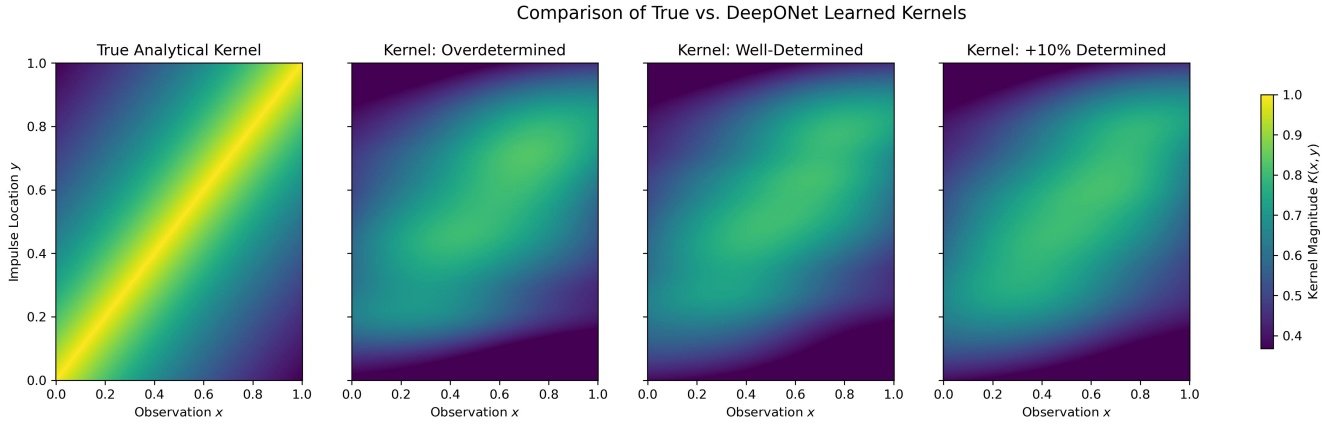

Figure 6.14: Visual output and kernel reconstruction of the

Model: Ridge on Trunk, Ridge on Branch and Tanh activation functions

Regime	**OOS MSE** $(\times 10^{-5})$	**OOS Rel** L_2	**Kernel MSE** $(\times 10^{-2})$	**Kernel Rel** L_2
Overdetermined	$1.94\ [1.58, 2.52]$	$0.0238\ [0.0208, 0.0261]$	2.668	0.2169
Well-Determined	$1.56\ [1.06, 2.10]$	$0.0177\ [0.0160, 0.0197]$	2.794	0.2219
+10% Determined	$2.86\ [2.14, 3.68]$	$0.0232\ [0.0214, 0.0258]$	2.391	0.2053

Table 6.32: Out-of-sample prediction and kernel estimation metrics across different system determination regimes (Mean $\pm$ 95% CI).

Model: Ridge on Trunk, Lasso Branch and Tanh activation functions.

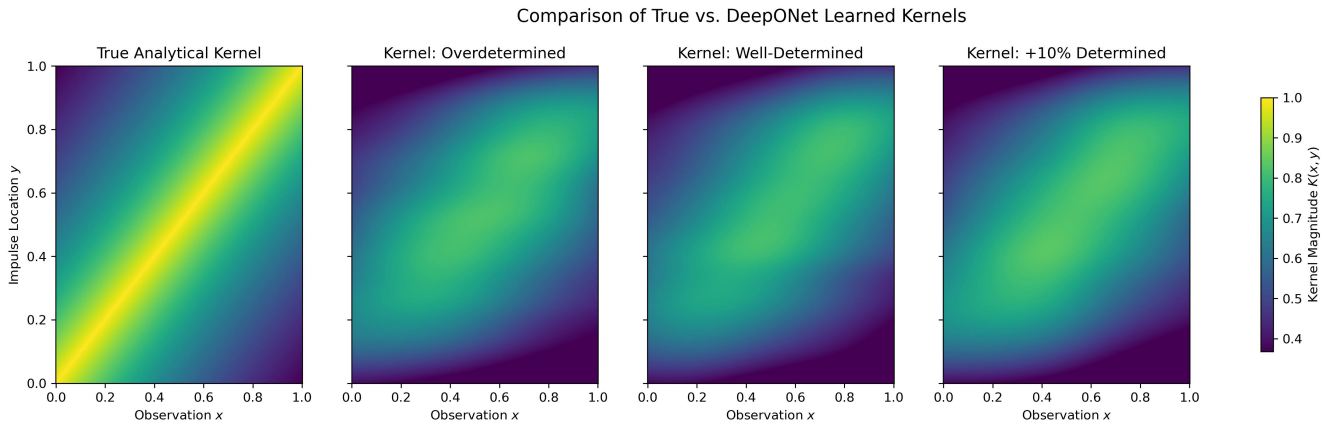

Figure 6.15: Visual output and kernel reconstruction of the

Model: Ridge on Trunk, Lasso on Branch and Tanh activation functions

Regime	**OOS MSE** $(\times 10^{-5})$	**OOS Rel** L_2	**Kernel MSE** $(\times 10^{-2})$	**Kernel Rel** L_2
Overdetermined	$2.08\ [1.68, 2.56]$	$0.0244\ [0.0213, 0.0284]$	3.169	0.2364
Well-Determined	$2.03\ [1.66, 2.45]$	$0.0237\ [0.0209, 0.0272]$	2.471	0.2087
+10% Determined	$1.78\ [1.53, 2.17]$	$0.0225\ [0.0201, 0.0247]$	2.163	0.1953

Table 6.33: Out-of-sample prediction and kernel estimation metrics across different system determination regimes (Mean $\pm$ 95% CI).

Model: Ridge on Trunk, Lasso on Branch and Sine activation functions.

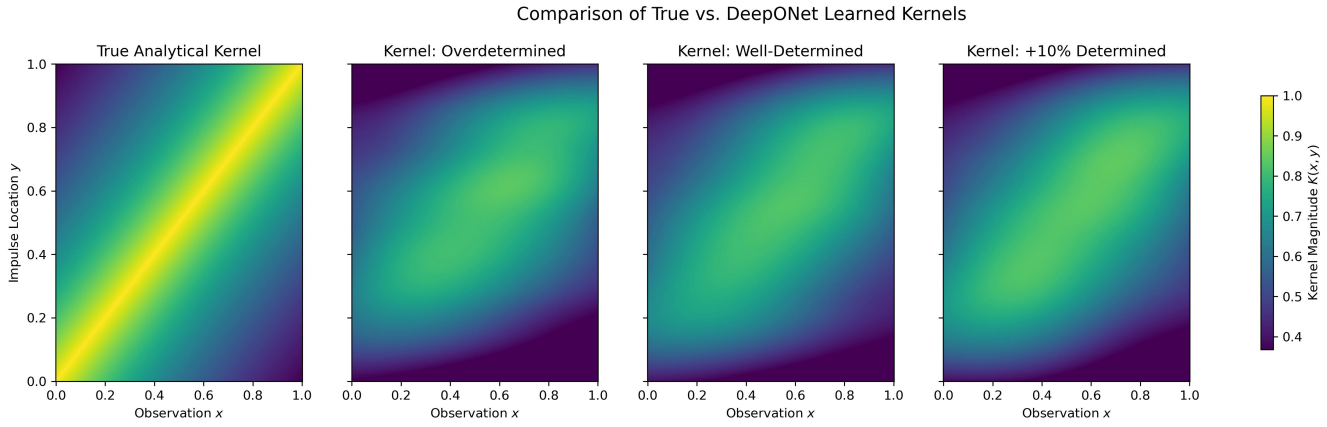

Figure 6.16: Visual output and kernel reconstruction of the

Model: Ridge on Trunk, Lasso on Branch and Sine activation functions

Regime	**OOS MSE** $(\times 10^{-5})$	**OOS Rel** L_2	**Kernel MSE** $(\times 10^{-2})$	**Kernel Rel** L_2
Overdetermined	$1.94\ [1.40, 2.58]$	$0.0229\ [0.0192, 0.0279]$	2.794	0.2219
Well-Determined	$1.82\ [1.39, 2.47]$	$0.0208\ [0.0191, 0.0228]$	3.029	0.2311
+10% Determined	$1.90\ [1.65, 2.21]$	$0.0228\ [0.0204, 0.0249]$	2.106	0.1927

Table 6.34: Out-of-sample prediction and kernel estimation metrics across different system determination regimes (Mean $\pm$ 95% CI).

Model: Ridge on Trunk, Lasso on Branch, Sine activasio functions, Soft Boundary conditions

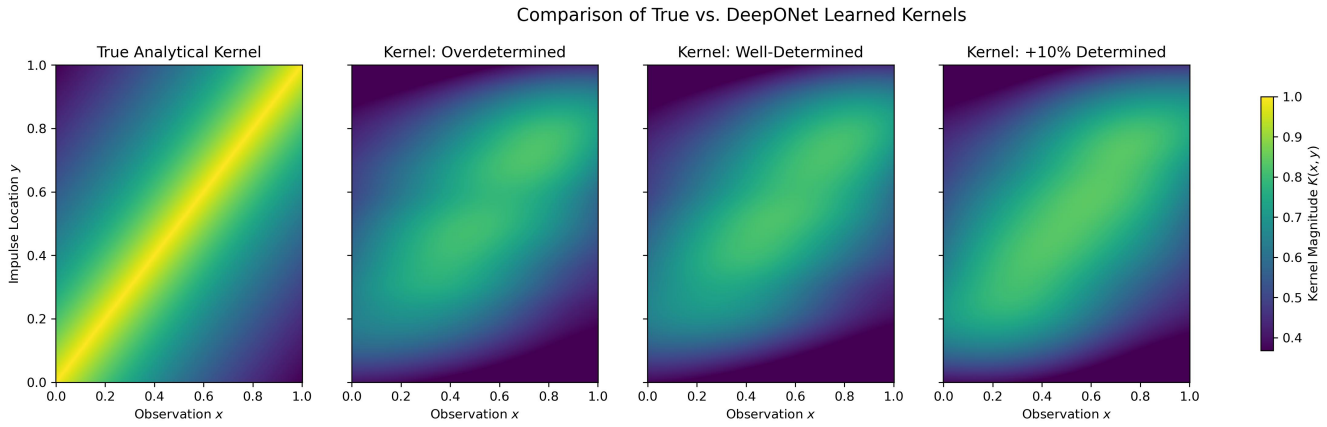

Figure 6.17: Visual output and kernel reconstruction of the

Model: Ridge on Trunk, Lasso on Branch, Sine activasio functions, Soft Boundary conditions

Bibliography

[1] Erwin Kreyszig. *Introductory Functional Analysis with Applications*. John Wiley & Sons, 1978.

[2] Arch W. Naylor and George R. Sell. *Linear Operator Theory in Engineering and Science*. Springer-Verlag New York, 1982.

[3] Jan van Neerven. *Functional Analysis*. Cambridge University Press, 2022.

[4] Alen Alexanderian. *On Compact Operators*. Lecture Notes.

[5] Haim Brezis. *Functional Analysis, Sobolev Spaces and Partial Differential Equations*. Springer New York, 2010.

[6] Guang-Bin Huang, Qin-Yu Zhu, and Chee-Kheong Siew. \Extreme learning machine: Theory and applications". In: *Neurocomputing* (2006).

[7] Lu Lu et al. \Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators". In: *Nature Machine Intelligence* (2021).

[8] Tianping Chen and Hong Chen. \Universal approximation to nonlinear operators by neural networks with arbitrary activation functions and its application to dynamical systems". In: *IEEE Transactions on Neural Networks* (1995).

[9] Gianluca Fabiani et al. \RandONets: Shallow networks with random projections for learning linear and nonlinear operators". In: *Journal of Computational Physics* (2025).