Data-driven discovery of Green's functions

Nicolas Boullé

University College

University of Oxford

A thesis submitted for the degree of

Doctor of Philosophy

Trinity 2022

Abstract

Discovering hidden partial differential equations (PDEs) and operators from data is an important topic at the frontier between machine learning and numerical analysis. Theoretical results and deep learning algorithms are introduced to learn Green's functions associated with linear partial differential equations and rigorously justify PDE learning techniques.

A theoretically rigorous algorithm is derived to obtain a learning rate, which characterizes the amount of training data needed to approximately learn Green's functions associated with elliptic PDEs. The construction connects the fields of PDE learning and numerical linear algebra by extending the randomized singular value decomposition to non-standard Gaussian vectors and Hilbert–Schmidt operators, and exploiting the low-rank hierarchical structure of Green's functions using hierarchical matrices.

Rational neural networks (NNs) are introduced and consist of neural networks with trainable rational activation functions. The highly compositional structure of these networks, combined with rational approximation theory, implies that rational functions have higher approximation power than standard activation functions. In addition, rational NNs may have poles and take arbitrarily large values, which is ideal for approximating functions with singularities such as Green's functions.

Finally, theoretical results on Green's functions and rational NNs are combined to design a human-understandable deep learning method for discovering Green's functions from data. This approach complements state-of-the-art PDE learning techniques, as a wide range of physics can be captured from the learned Green's functions such as dominant modes, symmetries, and singularity locations.

Acknowledgements

I would first like to thank my supervisors Patrick Farrell, Marie Rognes, and Alex Townsend for their guidance and suggestions. Their passion and excitement for the field of numerical analysis, as well as their high academic standards, have been a constant source of inspiration and motivation during my DPhil.

I would also like to thank my confirmation examiners, Christoph Reisinger and Justin Sirignano, for their comments and suggestions, as well as Andrew Stuart and Jared Tanner for accepting to be my thesis examiners.

This thesis benefited from discussions with great collaborators, including Efstathios Charalampidis, Vassilios Dallas, Christopher Earls, Ada Ellingsrud, Panayotis Kevrekidis, Seick Kim, Yuji Nakatsukasa, Alberto Paganini, Debasmita Samaddar, Tianyi Shi, and Jonasz Słomka.

I am grateful to Simula Research Laboratory for co-funding my DPhil along with University College, the Oxford-Radcliffe scholarship, and the InFoMM CDT.

I thank my friends and colleagues from Oxford and Cornell, Boris Andrews, Francis Aznaran, Pablo Brubeck, Dan Fortunato, Marc Gilles, Gonzalo Gonzalez de Diego, Andrew Horning, Fabian Laakmann, Maike Meier, John Papadopoulos, Alex Puiu, Tianyi Shi, and Heather Wilber, who made this DPhil always enjoyable and fun with great academic and non-academic conversations.

Finally, I am grateful to my family and Tina for their continuous support and encouragements throughout the years.

Contents

Chapter 1

Introduction

This thesis aims at understanding whether partial differential equations (PDEs) can be discovered from data by connecting standard mathematical fields, such as numerical linear algebra, probability, and PDE analysis, with modern deep learning techniques. We focus on learning Green's functions associated with linear PDEs from pairs of forcing functions and solutions. Theoretical bounds exploiting the regularity of the problem are derived and a practical deep learning algorithm is proposed.

Chapter 2 derives a theoretically-rigorous scheme for learning Green's functions associated with elliptic PDEs in three dimensions, given input-output pairs. A learning rate is obtained, giving a bound on the number of training pairs needed to learn a Green's function to within a prescribed accuracy with high probability. Along the way, the randomized singular value decomposition (SVD) is extended from matrices to Hilbert–Schmidt (HS) operators, and a quantity is introduced to measure the quality of the training forcing terms to learn Green's functions. The randomized SVD is a popular and effective algorithm for computing a near-best rank k approximation of a matrix using matrix-vector products with standard Gaussian vectors.

Chapter 3 extends the randomized SVD to multivariate Gaussian vectors, allowing one to incorporate prior knowledge of the matrix into the algorithm. This enables us to explore the continuous analogue of the randomized SVD for HS operators using operator-function products with functions drawn from a Gaussian process (GP). A new covariance kernel for GPs, based on weighted Jacobi polynomials, is constructed to rapidly sample the GP and control the smoothness of the randomly generated functions. Numerical examples on matrices and HS operators demonstrate the applicability of the algorithm.

Chapter 4 considers neural networks with rational activation functions. The choice of the nonlinear activation function in deep learning architectures is crucial and heavily impacts the performance of a neural network. We establish optimal bounds in

terms of network complexity and prove that rational neural networks approximate smooth functions more efficiently than networks with Rectified Linear Unit (ReLU) activation functions with exponentially smaller depth. The flexibility and smoothness of rational activation functions make them an attractive alternative to ReLU, as demonstrated by numerical experiments.

Chapter 5 develops a data-driven approach for learning Green's functions using deep learning. By collecting physical system responses under excitations drawn from a Gaussian process, we train rational neural networks to learn Green's functions of hidden linear PDEs. These functions reveal human-understandable properties and features, such as linear conservation laws and symmetries, along with shock and singularity locations, boundary effects, and dominant modes. The technique is illustrated on several examples and allows us to capture a range of physics, including advection-diffusion, viscous shocks, and Stokes flow in a lid-driven cavity.

1.1 Deep learning

Deep learning has become an important topic across many domains of science due to its recent successes in image recognition, speech recognition, and drug discovery [89, 114, 118, 138]. Deep learning techniques are based on objects called artificial neural networks (NNs), which apply a succession of mathematical transformations on an input variable x to output a variable y, where $\mathcal{N}(x) = y$ and $\mathcal{N}$ denotes the neural network. An example of simple data fitting task is to assign labels 0 or 1 to points in $\mathbb{R}^2$, where, in this case, $x \in \mathbb{R}^2$ and $y \in \{0, 1\}$ [88]. A large number of NN architectures, characterized by the type of mathematical operations used, have been proposed over the past decades for performing different tasks, such as convolutional NNs for classifying images [114, 119], recurrent and long short-term memory neural networks for speech recognition [78, 90, 196], and generative adversarial network to generate realistic images [76, 103].

We consider one of the most standard types of deep learning model called feed-forward neural networks or multilayer perceptrons [75, Chapt. 6]. Let $L \geq 1$ be an integer and n_1, n_L be the respective dimension of the input and output data. A feed-forward network $\mathcal{N} : \mathbb{R}^{n_1} \to \mathbb{R}^{n_L}$, mapping from $\mathbb{R}^{n_1}$ to $\mathbb{R}^{n_L}$, with L layers consists of a composition of $L - 1$ functions $f_1, \ldots, f_{L-1}$ of the form

$$\mathcal{N}(x) = f_{L-1} \circ \cdots \circ f_1(x), \quad x \in \mathbb{R}^{n_1}.$$

At a given layer $1 \leq i \leq L-1$, the nonlinear transformation $f_i : \mathbb{R}^{n_i} \to \mathbb{R}^{n_{i+1}}$ determines the output of the neural network. The layers $2 \leq i \leq L-1$ are called the hidden layers of the network and their dimensionality determines the width of the network, while the number of layers is referred to as the depth [75, Chapt. 6]. For $1 \leq i \leq L-1$, we choose the function f_i to be of the form

$$f_i : x \mapsto \sigma(W_i x + b_i), \quad x \in \mathbb{R}^{n_i},$$

where $W_i \in \mathbb{R}^{n_i \times n_{i+1}}$ is a matrix called the weight matrix, $b_i \in \mathbb{R}^{n_{i+1}}$ is a bias vector, and σ is a nonlinear function called the activation function (also called activation unit). The weight matrices and bias vectors are trainable parameters of the network, and their coefficients are usually obtained using a gradient-based optimization algorithm, such as stochastic gradient descent, applied to a training dataset containing examples of inputs and expected associated outputs of the network [75, Chapt. 6.2]. In this thesis, we will measure the network complexity using its total number of parameters (*i.e.,* size) and number of layers (depth), which are standard measures in theoretical deep learning [8].

In Chapter 4, we will consider NNs with rational activation functions and derive theoretical results that quantify the size needed to approximate smooth functions within a prescribed accuracy. We will establish a connection between standard approximation theory for rational functions and the highly compositional structure of neural networks to show that rational neural networks require fewer parameters than ReLU networks to approximate smooth functions. We expect that the smoothness of rational neural networks away from their poles, together with their potential singularities, make them an interesting alternative to standard activation functions for physics-informed machine learning applications.

1.2 Physics-informed machine learning

Over the past decades, there has been spectacular progress in numerical techniques for solving PDEs, such as finite element methods, finite differences, and spectral methods [102]. However, solving inverse problems to identify parameters of a model or learn a physical model from real-world data remains highly challenging due to missing and noisy data [11, 209]. Hence, such problems are often ill-posed and require a data-driven approach. Recently, the fields of numerical analysis and machine learning have successfully converged towards physics-informed machine learning, which integrates partial data and prior knowledge on governing physical laws to solve inverse problems

using neural networks [102]. The flexibility of the networks, due to the large potential choices of architectures, along with their generalization ability in the presence of big data, either generated by numerical simulations or acquired via experiments, makes them ideal for such tasks. On the other hand, the selection of a specific architecture is a challenging task and is difficult to justify mathematically due to the complexity of the models.

One example of problems that can be tackled by deep learning is to solve a PDE by training a NN on initial and boundary training data. Two popular approaches are physics-informed neural networks (PINNs) [184] and the deep Galerkin method [203], which, in their original formulation, aim to solve PDEs of the form

$$\frac{\partial u}{\partial t} + \mathcal{L}(u) = 0, \quad x \in D \subset \mathbb{R}^d, \quad t \in [0, T], \tag{1.1}$$

where the partial differential operator $\mathcal{L}$ is potentially nonlinear. The left-hand side of Equation (1.1) is denoted by $f(x, t)$, *i.e.*, $f \coloneqq u_t + \mathcal{L}(u)$. These techniques are attractive because they are mesh-free as they do not require a spatial discretization of the domain and can be applied in high dimensions. The PINN approach consists of approximating the solution u to Equation (1.1) by a neural network. This results in a physics-informed neural network f, which can be evaluated using chain rule and automatic differentiation [14, 184]. The loss function is expressed as a sum of a supervised loss of data measurements at the boundary and an unsupervised loss of PDE [102, 184]:

$$\text{Loss} = w_{\text{data}}\mathcal{L}_{\text{data}} + w_{\text{PDE}}\mathcal{L}_{\text{PDE}},$$

where w_{data} and w_{PDE} are weights balancing the two terms and $\mathcal{L}_{\text{data}}$, $\mathcal{L}_{\text{PDE}}$ are defined as

$$\mathcal{L}_{\text{data}} = \frac{1}{N_{\text{data}}} \sum_{i=1}^{N_{\text{data}}} |u(x_i^{bdr}, t_i^{bdr}) - u_i^{bdr}|^2, \quad \mathcal{L}_{\text{PDE}} = \frac{1}{N_{\text{PDE}}} \sum_{j=1}^{N_{\text{PDE}}} |f(x_j^{dom}, t_j^{dom})|^2.$$

Here, $\{(x_i^{bdr}, t_i^{bdr})\}$ are points sampled at the initial and boundary locations, while the points $\{(x_j^{dom}, t_j^{dom})\}$ are sampled on the entire domain, and $\mathcal{L}_{\text{PDE}}$ is the average of the squared residual of the PDE evaluated at $\{(x_j^{dom}, t_j^{dom})\}$. These methods have been generalized since their introductions to tackle a wide range of PDEs such as integro-differential equations [136], fractional PDEs [169], and stochastic PDEs [244], and have been applied to problems in fluid mechanics [185], geophysics [124], and materials science [202].

This thesis focuses on another aspect of physics-informed machine learning called PDE learning, whose aim is to discover, or learn, a mathematical model from data. We consider stationary PDEs of the form:

$$\mathcal{L}(u) = f,$$

where $\mathcal{L}$ is a partial differential operator, f is called the forcing term, and u the associated solution of the PDE. The approaches that dominate the PDE learning literature focus on the "forward" problem and aim to discover properties of the differential operator $\mathcal{L}$. As an example, sparsity-promoting techniques [36, 195, 246] consist of building a library of states u and its spatio-temporal derivatives $u_t, u_x, u_{xx}, u_y, \ldots$ to identify parameters (or coefficients) and discover the main contributing terms in $\mathcal{L}$. Another method aims to find a symbolic expression for $\mathcal{L}$ and identify its dominant coefficients by solving a regression problem [224, 225]. Finally, one can also project the operator $\mathcal{L}$ onto a low-dimensional subspace to build a reduced-order model and to significantly speed up standard numerical solvers [177, 178].

An alternative approach, which we will consider, is to study the "inverse" problem and directly approximate the PDE solution operator, $\mathcal{L}^{-1} : f \mapsto u$, by an artificial neural network $\mathcal{N}$ from training pairs of forcing terms and solutions $\{f_j, u_j\}_{j=1}^{N}$ [71, 112, 126, 127, 128, 135, 233]. The network $\mathcal{N}$ takes a forcing term f evaluated at a finite number of sensors $\{y_i\}_{i=1}^{N_f}$ and a point x in the domain of $\mathcal{L}^{-1}(f)$ and outputs a real number approximating the solution u to the PDE $\mathcal{L}(u) = f$ evaluated at x:

$$\mathcal{N}\left(\begin{bmatrix} f(y_1) & \cdots & f(y_{N_f}) \end{bmatrix}^\top, x\right) \approx u(x).$$

The NN is then trained by minimizing the following loss function using stochastic gradient descent algorithms:

$$\text{Loss} = \frac{1}{NN_uN_f} \sum_{k=1}^{N} \sum_{i=1}^{N_u} \sum_{j=1}^{N_f} \left| \mathcal{N}\left(\begin{bmatrix} f_k(y_1) & \cdots & f_k(y_{N_f}) \end{bmatrix}^\top, x_i\right) - u_k(x_i) \right|^2,$$

where $\{x_i\}_{i=1}^{N_u}$ are spatial points at which the solutions are measured. Unlike coefficient discovery techniques, this approach provides a fast solver for PDEs, which may outperform state-of-the-art numerical solvers [127]. However, the physical interpretation of the learned solution operator remains highly challenging due to the mathematical complexity of the neural network that approximates it. Several black-box deep learning techniques are proposed to approximate the solution operator, which maps forcing terms f to observations of the associated system's responses u such

that $\mathcal{L}(u) = f$. These methods are based on the concept of neural operators [112], which generalize neural networks to learn maps between infinite-dimensional function spaces, and mainly differ in their choice of the neural network architecture that is used to approximate the solution map. For example, Fourier neural operator [127] uses a Fourier transform at each layer, while DeepONet [135] contains a concatenation of 'trunk' and 'branch' networks to enforce additional structure.

On the theoretical side, most of the research has focused on the approximation theory of infinite-dimensional operators by NNs, such as the generalization of the universal approximation theorem [46] to shallow and deep NNs [38, 135] as well as error estimates for Fourier neural operators and DeepONets with respect to the network width and depth [111, 112, 116]. Other approaches aim to approximate the matrix of the discretized Green's functions associated with elliptic PDEs from matrix-vector multiplications by exploiting sparsity patterns or hierarchical structure of the matrix [130, 198]. In addition, [49] derived convergence rates for learning linear self-adjoint operators based on the assumption that the target operator is diagonal in the basis of the Gaussian prior.

In this thesis, we focus on learning linear partial differential operators $\mathcal{L}$ for which the solution operator can be written as an integral operator,

$$\mathcal{L}^{-1}(f)(x) = \int_D G(x, y) f(y) \, \mathrm{d}y = u(x),$$

whose kernel G is known as the Green's function. Our approach contrasts with prior works because we aim to approximate the Green's function instead of the integral operator. As we will see in Chapters 2 and 5, imposing a prior structure on the solution operator offers theoretical and practical advantages over recent PDE learning techniques. First, standard mathematical techniques from elliptic PDE theory and numerical analysis can be exploited to derive rigorous results that quantify the amount of training data needed to learn the solution operator to within a prescribed accuracy. These types of results are notoriously challenging to obtain for deep learning algorithms due to the high nonlinearity of neural network architectures and the complexity of the optimization procedure. Secondly, unlike black-box deep learning techniques, it is possible to extract physical features of the original PDE from the associated Green's function, which is a well-understood mathematical object.

1.3 Green's functions

Throughout this thesis, we consider linear boundary value problems defined on a bounded domain $D \subset \mathbb{R}^d$, with $d \geq 1$, of the form:

$$\begin{aligned} \mathcal{L}u &= f, \quad \text{in } D, \\ u &= 0, \quad \text{on } \partial D, \end{aligned}$$

where $\mathcal{L}$ is a linear partial differential operator and $f : D \to \mathbb{R}$ is a given forcing function. A typical example of such problems is the Poisson equation in one dimension:

$$-\frac{d^2u}{dx^2} = f, \quad x \in (0,1), \quad u(0) = u(1) = 0. \tag{1.3}$$

Equation (1.3) can be solved for any forcing function f by introducing a kernel $G : [0,1] \times [0,1] \to \mathbb{R}$ so that the solution u can be expressed as the following integral [64, 79, 190],

$$u(x) = \int_0^1 G(x,y) f(y) \, \mathrm{d}y, \quad x \in [0,1]. \tag{1.4}$$

The function G is called the Green's function and is a solution to the equation $\mathcal{L}G(x,y) = \delta(x-y)$, where δ is the Dirac delta function and $x, y \in [0,1]$.

Green's functions are useful because they are independent of the forcing terms and only characterize the partial differential operators and boundary conditions. Once the Green's function has been determined, then the solution to Equation (1.3) with any forcing term can be obtained by computing the integral in Equation (1.4), which is numerically easier than solving the original PDE and imposing the appropriate boundary conditions [190]. Additionally, several properties of the PDE can be recovered from the Green's function, such as symmetries or eigenvalues.

Traditional methods for finding Green's functions can be summarized as deriving analytical formulas, computing eigenvalue expansions, or numerically solving a singular PDE [64, 190]. This is difficult when the geometry of the domain is complex or when the PDE has variable coefficients. Moreover, it requires knowledge of the partial differential operator, which may not be accessible in real applications [102]. Other works study properties of Green's functions and provide theoretical results such as decay bounds along the diagonal of the domain [42, 80, 91, 100] or low-rank structure on separable domains [16, 28, 63].

In this thesis, we aim to approximate Green's functions from pairs of forcing terms and system's responses $\{(f_j, u_j)\}_{j=1}^N$ by exploiting their low-rank structure on well-separated domains [16], and combining it with randomized numerical linear algebra [86].

1.4 Low-rank approximation

Let $\mathbf{A}$ be an $m \times n$ real matrix with $m \geq n$ and $k \leq n$ be an integer. The best rank k approximation to $\mathbf{A}$ in the Frobenius norm is the $m \times n$ real matrix $\mathbf{A}_k$, which is solution to the following minimization problem:

$$\min_{\mathbf{A}_k \in \mathbb{R}^{m \times n}} \|\mathbf{A} - \mathbf{A}_k\|_{\mathrm{F}} \quad \text{subject to} \quad \operatorname{rank}(\mathbf{A}_k) \leq k, \tag{1.5}$$

where $\|\cdot\|_{\mathrm{F}}$ denotes the Frobenius norm defined as $\|\mathbf{A}\|_{\mathrm{F}} = \sqrt{\operatorname{Tr}(\mathbf{A}\mathbf{A}^*)}$. The Eckart–Young theorem [58] states that (1.5) has a unique solution given by the truncation of the singular value decomposition of $\mathbf{A}$ to the kth term. The SVD of an $m \times n$ real matrix $\mathbf{A}$, with $m \geq n$, is a factorization of the form $\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^*$, where $\mathbf{U}$ is an $m \times m$ orthogonal matrix of left singular vectors, $\mathbf{\Sigma}$ is an $m \times n$ diagonal matrix with entries $\sigma_1(\mathbf{A}) \geq \cdots \geq \sigma_n(\mathbf{A}) \geq 0$, and $\mathbf{V}$ is an $n \times n$ orthogonal matrix of right singular vectors [74]. Then,

$$\min_{\substack{\mathbf{A}_k \in \mathbb{R}^{m \times n} \\ \operatorname{rank}(\mathbf{A_k}) \leq k}} \|\mathbf{A} - \mathbf{A}_k\|_{\mathrm{F}} = \left(\sum_{j=k+1}^{n} \sigma_j(\mathbf{A})^2 \right)^{1/2},$$

where

$$\mathbf{A}_k = \sum_{j=1}^{k} \sigma_j(\mathbf{A}) u_j v_j^*.$$

Here, u_j and v_j denote the jth column of $\mathbf{U}$ and $\mathbf{V}$, respectively.

This result can be generalized to functions [200, 216] and, in particular, Green's functions of the form $G : D_1 \times D_2 \to \mathbb{R}$, where $D_1, D_2 \subset \mathbb{R}^d$. As an example, if $D_1 = [a, b]$ and $D_2 = [c, d]$ are two real intervals, and G is square-integrable, then it can be written as the following infinite series, which converges in the $L^2(D_1 \times D_2)$ sense to G,

$$G(x, y) = \sum_{\substack{j=1 \\ \sigma_j > 0}}^{\infty} \sigma_j u_j(x) v_j(y), \quad x \in D_1,\, y \in D_2,$$

where $\{u_j\}_{j \geq 1}$ and $\{v_j\}_{j \geq 1}$ form an orthonormal basis of $L^2(D_1)$ and $L^2(D_2)$, and $\sigma_1 \geq \sigma_2 \geq \cdots \geq 0$ are called the singular values of G. This series is referred to as the SVD of G. Similar to matrices, the best rank k approximant to G is obtained by truncating its SVD after k terms to obtain a separable approximation

$$G_k(x, y) = \sum_{j=1}^{k} \sigma_j u_j(x) v_j(y), \quad x \in D_1,\, y \in D_2.$$

By the Eckart–Young theorem, G_k is solution to the following minimization problem:

$$\min_{\substack{f_j \in L^2(D_1) \\ g_j \in L^2(D_2)}} \|G - \sum_{j=1}^{k} f_j g_j\|_{L^2(D_1 \times D_2)} = \|G - G_k\|_{L^2(D_1 \times D_2)} = \left(\sum_{j=k+1}^{\infty} \sigma_j^2 \right)^{1/2}.$$

Let $0 < \epsilon < 1$. If there exists an integer $k > 0$ and a separable expression satisfying

$$\|G - \sum_{j=1}^{k} f_j g_j\|_{L^2(D_1 \times D_2)} \leq \epsilon \|G\|_{L^2(D_1 \times D_2)}, \quad f_j \in L^2(D_1),\, g_j \in L^2(D_2),$$

then we say that G has numerical rank smaller than k. We remark that one can easily obtain a bound on the tail of the singular values of G by applying the Eckart–Young theorem as follows,

$$\left(\sum_{j=k+1}^{\infty} \sigma_j^2 \right)^{1/2} = \min_{\substack{f_j \in L^2(D_1) \\ g_j \in L^2(D_2)}} \|G - \sum_{j=1}^{k} f_j g_j\|_{L^2(D_1 \times D_2)} \leq \epsilon \|G\|_{L^2(D_1 \times D_2)}.$$

When $k = \mathcal{O}(\log^{\delta}(1/\epsilon))$ for some small $\delta \in \mathbb{N}$ as $\epsilon \to 0$, then we say that G has exponentially decaying singular values on $D_1 \times D_2$.

1.5 Randomized singular value decomposition

Computing the SVD of a matrix is a fundamental linear algebra task in machine learning [173], statistics [240], and signal processing [7, 227]. As we saw in Section 1.4, the SVD plays a central role in numerical linear algebra because truncating it after k terms provides the best rank k approximation to $\mathbf{A}$ in the spectral and Frobenius norms [58, 151]. Since computing the SVD of a large matrix can be computationally infeasible, there are various principal component analysis (PCA) [2, 92, 174] algorithms that perform dimensionality reduction by computing near-best rank k matrix approximations from matrix-vector products [86, 145, 159, 164, 238]. The randomized SVD uses matrix-vector products with random test vectors and is one of the most popular algorithms for constructing a low-rank approximation to $\mathbf{A}$ [86, 145]. While the error analysis performed in [86] for the randomized SVD uses standard Gaussian random vectors, other random embedding techniques have been considered such as random permutations [5], sparse sign matrices [44, 147, 162, 226], and subsampled randomized trigonometric transforms (SRTTs) [4, 5, 172, 241] to mitigate the computational cost of Gaussian vectors in practical applications. Throughout this thesis, we will focus on Gaussian vectors because they yield a more precise error analysis (cf. [145, Sec. 8.3]).

First, one performs the matrix-vector products $y_1 = \mathbf{A}x_1, \ldots, y_{k+p} = \mathbf{A}x_{k+p}$, where $x_1, \ldots, x_{k+p}$ are standard Gaussian random vectors with identically and independently distributed entries and $p \geq 1$ is an oversampling parameter. Then, one computes the economized QR factorization $\begin{bmatrix} y_1 & \cdots & y_{k+p} \end{bmatrix} = \mathbf{QR}$, before forming the rank $\leq k + p$ approximant $\mathbf{QQ}^*\mathbf{A}$. Note that if $\mathbf{A}$ is symmetric, one can form $\mathbf{QQ}^*\mathbf{A}$ by computing $\mathbf{Q}(\mathbf{AQ})^*$ via matrix-vector products involving $\mathbf{A}$; otherwise it requires the adjoint $\mathbf{A}^*$. The quality of the rank $\leq k + p$ approximant $\mathbf{QQ}^*\mathbf{A}$ is characterized by the following bound for $u, t \geq 1$ [86, Thm. 10.7],

$$\|\mathbf{A} - \mathbf{QQ}^*\mathbf{A}\|_{\mathrm{F}} \leq \left(1 + t\sqrt{\frac{3k}{p+1}}\right) \sqrt{\sum_{j=k+1}^{n} \sigma_j^2(\mathbf{A})} + ut\frac{\sqrt{k+p}}{p+1}\sigma_{k+1}(\mathbf{A}), \tag{1.6}$$

with failure probability at most $2t^{-p} + e^{-u^2}$. The squared tail of the singular values of $\mathbf{A}$, *i.e.*, $\sqrt{\sum_{j=k+1}^{n} \sigma_j^2(\mathbf{A})}$, gives the best rank k approximation error to $\mathbf{A}$ in the Frobenius norm. This result shows that the randomized SVD can compute a near-best low-rank approximation to $\mathbf{A}$ with high probability. In Chapters 2 and 3, we will generalize this result to random vectors sampled from a multivariate normal distribution with any covariance matrix, and Hilbert–Schmidt operators.

1.6 Hilbert–Schmidt operators

Hilbert–Schmidt operators generalize the notion of matrices acting on vectors to infinite dimensions with linear operators acting on functions [93, Ch. 4]. First, let $D_1, D_2 \subset \mathbb{R}^d$ be two domains with $d \geq 1$. For $1 \leq p \leq \infty$, we denote by $L^p(D_1)$ the space of measurable functions defined on the domain D_1 with finite L^p norm, where

$$\begin{aligned} \|f\|_{L^p(D_1)} &= \left(\int_{D_1} |f(x)|^p \,\mathrm{d}x\right)^{1/p} \quad \text{if } p < \infty, \\ \|f\|_{L^\infty(D_1)} &= \inf\left\{C > 0,\, |f(x)| \leq C \text{ for almost every } x \in D_1\right\}. \end{aligned}$$

Since the space of square-integrable functions, $L^2(D_1)$, is a separable Hilbert space, it admits a complete orthonormal basis $\{e_j\}_{j=1}^{\infty}$.

A linear operator $\mathscr{F} : L^2(D_1) \to L^2(D_2)$ is an HS operator [93, Def. 4.4.2] if it has finite HS norm, $\|\mathscr{F}\|_{\mathrm{HS}}$, defined as

$$\|\mathscr{F}\|_{\mathrm{HS}} := \left(\sum_{j=1}^{\infty} \|\mathscr{F}e_j\|_{L^2(D_2)}^2\right)^{1/2} < \infty.$$

This norm does not depend on the choice of the basis [93, Thm. 4.4.1]. The archetypical example of an HS operator is an integral operator $\mathscr{F} : L^2(D_1) \to L^2(D_2)$ defined as

$$(\mathscr{F}f)(x) = \int_{D_1} G(x,y)f(y)\,\mathrm{d}y, \quad f \in L^2(D_1),\, x \in D_2,$$

where $G \in L^2(D_2 \times D_1)$ is the kernel of $\mathscr{F}$ and $\|\mathscr{F}\|_{\mathrm{HS}} = \|G\|_{L^2(D_2\times D_1)}$. The adjoint operator $\mathscr{F}^* : L^2(D_2) \to L^2(D_1)$ is defined as

$$(\mathscr{F}^*g)(y) = \int_{D_2} G(x,y)g(x)\,\mathrm{d}x, \quad g \in L^2(D_2),\, y \in D_1.$$

Since HS operators are compact operators, they have an SVD [93, Thm. 4.3.1]. That is, that for any $f \in L^2(D_1)$ we have

$$\mathscr{F}f = \sum_{j=1}^{\infty} \sigma_j \langle q_{1j}, f\rangle q_{2j}, \tag{1.7}$$

where the equality holds in the $L^2(D_2)$ sense. Here, $\sigma_1 \geq \sigma_2 \geq \cdots \geq 0$ denote the square roots of the eigenvalues of the self-adjoint operator $\mathscr{F}^*\mathscr{F}$, $\{q_{1j}\}$ are the orthonormal eigenvectors of $\mathscr{F}^*\mathscr{F}$, and $\{q_{2j}\}$ are the orthonormal eigenvectors of $\mathscr{F}\mathscr{F}^*$. We refer to $\{(\sigma_j, q_{1j}, q_{2j})\}_{j=1}^{\infty}$ as the singular system of $\mathscr{F}$. When the HS operator is an integral operator, we refer to its singular values as the singular values of the underlying kernel.

Moreover, one finds that $\|\mathscr{F}\|_{\mathrm{HS}}^2 = \sum_{j=1}^{\infty} \sigma_j^2$, which shows that the HS norm is an infinite dimensional analogue of the Frobenius matrix norm $\|\cdot\|_{\mathrm{F}}$. In the same way that truncating the SVD after k terms gives the best rank k matrix approximation, truncating Equation (1.7) gives the best rank k approximation in the HS norm. That is, [93, Thm. 4.4.7]

$$\min_{u_j \in L^2(D_1), v_j \in L^2(D_2)} \|\mathscr{F} - \sum_{j=1}^{k} \langle u_j, \cdot\rangle v_j\|_{\mathrm{HS}} = \|\mathscr{F} - \mathscr{F}_k\|_{\mathrm{HS}} = \left(\sum_{j=k+1}^{\infty} \sigma_j^2\right)^{1/2},$$

where the operator $\mathscr{F}_k$ is defined as

$$\mathscr{F}_k f = \sum_{j=1}^{k} \sigma_j \langle q_{1j}, f\rangle q_{2j}, \quad f \in L^2(D_1).$$

This result is known as the Eckart–Young–Mirsky theorem [58, 151]. We will exploit this theorem in Chapter 2 to extend the randomized SVD to HS operators and learn Green's functions.

1.7 Quasimatrices

Quasimatrices are an infinite dimensional analogue of tall-skinny matrices [218]. Let $D_1, D_2 \subseteq \mathbb{R}^d$ be two domains with $d \geq 1$, we say that $\boldsymbol{\Omega}$ is a $D_1 \times k$ quasimatrix, if $\boldsymbol{\Omega}$ is a matrix with k columns where each column is a function in $L^2(D_1)$. That is,

$$\boldsymbol{\Omega} = \begin{bmatrix} \omega_1 \,|\, & \cdots & |\, \omega_k \end{bmatrix}, \quad \omega_j \in L^2(D_1).$$

Quasimatrices are useful to define analogues of matrix operations for HS operators [48, 208, 218, 221]. For example, if $\mathscr{F} : L^2(D_1) \to L^2(D_2)$ is an HS operator, then we write $\mathscr{F}\boldsymbol{\Omega}$ to denote the quasimatrix obtained by applying $\mathscr{F}$ to each column of $\boldsymbol{\Omega}$. Moreover, we write $\boldsymbol{\Omega}^*\boldsymbol{\Omega}$ and $\boldsymbol{\Omega}\boldsymbol{\Omega}^*$ to mean the following:

$$\boldsymbol{\Omega}^*\boldsymbol{\Omega} = \begin{bmatrix} \langle \omega_1, \omega_1 \rangle & \cdots & \langle \omega_1, \omega_k \rangle \\ \vdots & \ddots & \vdots \\ \langle \omega_k, \omega_1 \rangle & \cdots & \langle \omega_k, \omega_k \rangle \end{bmatrix}, \quad \boldsymbol{\Omega}\boldsymbol{\Omega}^* = \sum_{j=1}^{k} \omega_j(x)\omega_j(y),$$

where $\langle \cdot, \cdot \rangle$ is the $L^2(D_1)$ inner-product. Many operations for rectangular matrices in linear algebra can be generalized to quasimatrices such as the SVD, QR, LU, and Cholesky factorizations [218].

Throughout this thesis, the HS operator denoted by $\boldsymbol{\Omega}\boldsymbol{\Omega}^*\mathscr{F} : L^2(D_1) \to L^2(D_2)$ is given by $\boldsymbol{\Omega}\boldsymbol{\Omega}^*\mathscr{F}f = \sum_{j=1}^{k} \langle \omega_j, \mathscr{F}f \rangle \omega_j$. Moreover, if $\boldsymbol{\Omega}$ has full column rank then $\mathbf{P}_{\boldsymbol{\Omega}}\mathscr{F} := \boldsymbol{\Omega}(\boldsymbol{\Omega}^*\boldsymbol{\Omega})^\dagger\boldsymbol{\Omega}^*\mathscr{F}$ is the orthogonal projection of the range of $\mathscr{F}$ onto the column space of $\boldsymbol{\Omega}$. Here, $(\boldsymbol{\Omega}^*\boldsymbol{\Omega})^\dagger$ is the pseudo-inverse of $\boldsymbol{\Omega}^*\boldsymbol{\Omega}$. This notation is convenient to state the generalization of the randomized SVD in infinite dimensions.

1.8 Gaussian processes

A Gaussian process is an infinite dimensional analogue of a multivariate Gaussian distribution and a function drawn from a GP is analogous to a randomly generated vector. If $K : D \times D \to \mathbb{R}$ is a continuous symmetric positive semi-definite kernel, where $D \subseteq \mathbb{R}^d$ is a domain, then a GP is a stochastic process $\{X_t,\, t \in D\}$ such that for every finite set of indices $t_1, \ldots, t_n \in D$ the vector of random variables $(X_{t_1}, \ldots, X_{t_n})$ is a multivariate Gaussian distribution with mean $(0, \ldots, 0)$ and covariance $K_{ij} = K(t_i, t_j)$ for $1 \leq i, j \leq n$. We denote a GP with mean $(0, \ldots, 0)$ and covariance kernel K by $\mathcal{GP}(0, K)$.

Since K is a continuous symmetric positive semi-definite kernel, it has nonnegative eigenvalues $\lambda_1 \geq \lambda_2 \geq \cdots \geq 0$ and there is an orthonormal basis of eigenfunctions

$\{\psi_j\}_{j=1}^{\infty}$ of $L^2(D)$ such that [93, Thm. 4.6.5]:

$$K(x,y) = \sum_{j=1}^{\infty} \lambda_j \psi_j(x)\psi_j(y), \quad \int_D K(x,y)\psi_j(y)\,\mathrm{d}y = \lambda_j \psi_j(x), \quad x, y \in D, \tag{1.8}$$

where the infinite sum is absolutely and uniformly convergent [148]. Note that the eigenvalues of K are the ones of the integral operator with kernel K. In addition, we define the trace of the covariance kernel K by $\mathrm{Tr}(K) \coloneqq \sum_{j=1}^{\infty} \lambda_j < \infty$. The eigendecomposition of K gives an algorithm for sampling functions from $\mathcal{GP}(0, K)$. In particular, if

$$\omega = \sum_{j=1}^{\infty} \sqrt{\lambda_j} c_j \psi_j,$$

where the coefficients $\{c_j\}_{j=1}^{\infty}$ are independent and identically distributed (i.i.d.) standard Gaussian random variables and the series converges in mean-square and uniformly, then $\omega \sim \mathcal{GP}(0, K)$. This is known as the Karhunen–Loève theorem [101, 133]. We also have [93, Thm. 7.2.5]

$$\mathbb{E}\Big[\|\omega\|_{L^2(D)}^2\Big] = \sum_{j=1}^{\infty} \lambda_j \mathbb{E}\big[c_j^2\big]\, \|\psi_j\|_{L^2(D)}^2 = \sum_{j=1}^{\infty} \lambda_j = \int_D K(y,y)\,\mathrm{d}y < \infty,$$

where the last equality is analogous to the fact that the trace of a matrix is equal to the sum of its eigenvalues. In this thesis, we restrict our attention to GPs with positive definite covariance kernels so that the eigenvalues of K are strictly positive.

1.9 Contribution

The material of Chapter 2 to Chapter 5 is based on the following four papers with collaborators:

- **Learning elliptic PDEs with randomized linear algebra**
  Nicolas Boullé and Alex Townsend
  Foundations of Computational Mathematics, 2022

- **A generalization of the randomized singular value decomposition**
  Nicolas Boullé and Alex Townsend
  International Conference on Learning Representations, 2022

- **Rational neural networks**
  Nicolas Boullé, Yuji Nakatsukasa, and Alex Townsend
  Neural Information Processing Systems, 2020

- **Data-driven discovery of Green's functions with human-understandable deep learning**
  Nicolas Boullé, Christopher J. Earls, and Alex Townsend
  Scientific Reports, 2022

My co-authors had advisory roles; I proved the main theoretical results, performed the numerical experiments, and was the lead author in writing the papers.

Chapter 2

Learning elliptic PDEs with randomized linear algebra*

Can one learn a differential operator from pairs of solutions and righthand sides? If so, how many pairs are required? These two questions have received significant research attention [65, 127, 134, 170]. From data, one hopes to eventually learn physical laws of nature or conservation laws that elude scientists in the biological sciences [243], computational fluid dynamics [185], and computational physics [180]. The literature contains many highly successful practical schemes based on deep learning techniques [146, 184]. However, the challenge remains to understand when and why deep learning is effective theoretically. This chapter describes the first theoretically-justified scheme for discovering scalar-valued elliptic partial differential equations (PDEs) in three variables from input-output data and provides a rigorous learning rate. While our novelties are mainly theoretical, we hope to motivate future practical choices in PDE learning.

Let $D \subset \mathbb{R}^3$ be a bounded domain with Lipschitz smooth boundary, $L^2(D)$ be the space of square-integrable functions defined on D, $\mathcal{H}^k(D)$ be the space of k times weakly differentiable functions in the L^2-sense, and $\mathcal{H}_0^1(D)$ be the closure of $\mathcal{C}_c^\infty(D)$ in $\mathcal{H}^1(D)$. Here, $\mathcal{C}_c^\infty(D)$ is the space of infinitely differentiable compactly supported functions on D. Roughly speaking, $\mathcal{H}_0^1(D)$ are the functions in $\mathcal{H}^1(D)$ that are zero on the boundary of D. We suppose that there is an unknown second-order uniformly elliptic linear PDE operator $\mathcal{L} : \mathcal{H}^2(D) \cap \mathcal{H}_0^1(D) \to L^2(D)$ [64], which takes the form

$$\mathcal{L}u(x) = -\nabla \cdot (A(x)\nabla u) + c(x) \cdot \nabla u + d(x)u, \quad x \in D, \quad u|_{\partial D} = 0. \tag{2.1}$$

*This chapter is based on a paper with Alex Townsend [32], published in Foundations of Computational Mathematics. Townsend had an advisory role; I proved the theoretical results and was the lead author in writing the paper.

Here, for every $x \in D$, we have that $A(x) \in \mathbb{R}^{3\times 3}$ is a symmetric positive definite matrix with bounded coefficient functions so that $A_{ij} \in L^\infty(D)$, $c \in L^r(D)$ with $r \geq 3$, $d \in L^s(D)$ for $s \geq 3/2$, and $d(x) \geq 0$ [106]. We emphasize that the regularity requirements on the variable coefficients are quite weak.

The goal of PDE learning is to discover the operator $\mathcal{L}$ from $N \geq 1$ input-output pairs, *i.e.*, $\{(f_j, u_j)\}_{j=1}^N$, where $\mathcal{L}u_j = f_j$ and $u_j|_{\partial D} = 0$ for $1 \leq j \leq N$. There are two main types of PDE learning tasks: (1) Experimentally-determined input-output pairs, where one must do the best one can with the predetermined information and (2) Algorithmically-determined input-output pairs, where the data-driven learning algorithm can select $f_1, \ldots, f_N$ for itself. In this chapter, we focus on the PDE learning task where we have algorithmically-determined input-output pairs and aim to provide an upper bound on the sample complexity of the Green's function G associated with $\mathcal{L}$, *i.e.* characterize the number of pairs needed to learn G within a prescribed accuracy. In particular, we suppose that the functions $f_1, \ldots, f_N$ are generated at random and are drawn from a Gaussian process (GP) (see Section 1.8). Note that alternative strategies analogue to a power scheme in randomized numerical linear algebra [74, 86, 191, 192] to generate forcing terms iteratively might lead to better approximation errors. To keep our theoretical statements manageable, we restrict our attention to PDEs of the form:

$$\mathcal{L}u = -\nabla \cdot (A(x)\nabla u), \quad x \in D, \quad u|_{\partial D} = 0. \tag{2.2}$$

Lower-order terms in Equation (2.1) should cause few theoretical problems [16], though our algorithm and our bounds get far more complicated.

The approach that dominates the PDE learning literature is to directly learn $\mathcal{L}$ by either (1) learning parameters in the PDE [24, 247], (2) using neural networks (NNs) to approximate the action of the PDE on functions [180, 182, 183, 184, 185], or (3) deriving a model from a library of operators via sparsity considerations [36, 141, 195, 197, 231, 234]. Instead of trying to learn the unbounded, closed operator $\mathcal{L}$ directly, we follow [27, 65, 71] and discover the Green's function associated with $\mathcal{L}$. That is, we attempt to learn the function $G : D \times D \to \mathbb{R}^+ \cup \{\infty\}$ such that [64]

$$u_j(x) = \int_D G(x, y) f_j(y)\, \mathrm{d}y, \quad x \in D, \quad 1 \leq j \leq N. \tag{2.3}$$

Seeking G, as opposed to $\mathcal{L}$, has several theoretical benefits:

1. The integral operator in Equation (2.3) is compact [60], while $\mathcal{L}$ is only closed [59]. This allows G to be rigorously learned by input-output pairs $\{(f_j, u_j)\}_{j=1}^N$, as its range can be approximated by finite-dimensional spaces (see Theorem 2.3).

2. It is known that G has a hierarchical low-rank structure [16, Thm. 2.8]: for $0 < \epsilon < 1$, there exists a function $G_k(x,y) = \sum_{j=1}^{k} g_j(x)h_j(y)$ with $k = \mathcal{O}(\log^4(1/\epsilon))$ such that [16, Thm. 2.8]

$$\|G - G_k\|_{L^2(X\times Y)} \leq \epsilon \|G\|_{L^2(X\times \hat{Y})},$$

where $X, Y \subseteq D$ are sufficiently separated domains, and $Y \subseteq \hat{Y} \subseteq D$ denotes a larger domain than Y (see Theorem 2.4 for the definition). The further apart X and Y, the faster the singular values of G decay. Moreover, G also has an off-diagonal decay property [80, 100]:

$$G(x,y) \leq \frac{c}{\|x-y\|_2}\|G\|_{L^2(D\times D)}, \quad x \neq y,\ x \in D,\ y \in D,$$

where c is a constant independent of x and y. Exploiting these structures of G leads to a rigorous algorithm for constructing a global approximant to G (see Section 2.2).
3. The function G is smooth away from its diagonal, allowing one to efficiently approximate it [80].

Once a global approximation $\tilde{G}$ has been constructed for G using input-output pairs, given a new righthand side f one can directly compute the integral in Equation (2.3) to obtain the corresponding solution u to Equation (2.1). Usually, numerically computing the integral in Equation (2.3) must be done with sufficient care as G possesses a singularity when $x = y$. However, our global approximation $\tilde{G}$ has a hierarchical structure and is constructed as 0 near the diagonal. Therefore, for each fixed $x \in D$, we simply recommend that $\int_D \tilde{G}(x,y)f_j(y)\,\mathrm{d}y$ is partitioned into the panels that corresponds to the hierarchical decomposition, and then discretized each panel with a quadrature rule.

There are two main contributions in this chapter: (1) the generalization of the randomized singular value decomposition (SVD) algorithm for learning matrices from matrix-vector products to Hilbert–Schmidt (HS) operators and (2) a theoretical learning rate for discovering Green's functions associated with PDEs of the form Equation (2.2). These contributions are summarized in Theorems 2.1 and 2.3.

Theorem 2.1 says that, with high probability, one can recover a near-best rank k HS operator using $k+p$ operator-function products, for a small integer p. In the bound of the theorem, a quantity, denoted by $0 < \gamma_k \leq 1$, measures the quality of the input-output training pairs (see Sections 2.1.1 and 2.1.4). We then combine Theorem 2.1

with the theory of Green's functions for elliptic PDEs to derive a theoretical learning rate for PDEs.

In Theorem 2.3, we show that Green's functions associated with uniformly elliptic PDEs in three dimensions can be recovered using $N = \mathcal{O}(\epsilon^{-6}\log^4(1/\epsilon))$ input-output pairs $(f_j, u_j)_{j=1}^N$ to within an accuracy of $\mathcal{O}(\Gamma_\epsilon^{-1/2}\log^3(1/\epsilon)\epsilon)$ with high probability, for $0 < \epsilon < 1$. Our learning rate associated with uniformly elliptic PDEs in three variables is therefore $\mathcal{O}(\epsilon^{-6}\log^4(1/\epsilon))$. The quantity $0 < \Gamma_\epsilon \leq 1$ (defined in Equation (2.23)) measures the quality of the GP used to generate the random functions $\{f_j\}_{j=1}^N$ for learning G. We emphasize that the number of training pairs is small only if the GP's quality is high. The probability bound in Theorem 2.3 implies that the constructed approximation is close to G with high probability and converges almost surely to the Green's function as $\epsilon \to 0$.

2.1 Low-rank approximation of Hilbert–Schmidt operators

In a landmark paper, Halko, Martinsson, and Tropp proved that one could learn the column space of a finite matrix—to high accuracy and with a high probability of success—by using matrix-vector products with standard Gaussian random vectors [86]. We now set out to generalize this from matrices to HS operators. Alternative randomized low-rank approximation techniques such as the generalized Nyström method [159] might also be generalized in a similar manner. Since the proof is relatively long, we state our final generalization now.

Theorem 2.1. *Let $D_1, D_2 \subseteq \mathbb{R}^d$ be domains with $d \geq 1$ and $\mathscr{F} : L^2(D_1) \to L^2(D_2)$ be an HS operator. Select a target rank $k \geq 1$, an oversampling parameter $p \geq 2$, and a $D_1 \times (k+p)$ quasimatrix $\boldsymbol{\Omega}$ such that each column is i.i.d. and drawn from $\mathcal{GP}(0, K)$, where $K : D_1 \times D_1 \to \mathbb{R}$ is a continuous symmetric positive definite kernel with eigenvalues $\lambda_1 \geq \lambda_2 \geq \cdots > 0$. If $\mathbf{Y} = \mathscr{F}\boldsymbol{\Omega}$, then*

$$\mathbb{E}[\|\mathscr{F} - \mathbf{P}_{\mathbf{Y}}\mathscr{F}\|_{\mathrm{HS}}] \leq \left(1 + \sqrt{\frac{1}{\gamma_k}\frac{k(k+p)}{p-1}}\right)\left(\sum_{j=k+1}^{\infty}\sigma_j^2\right)^{1/2}, \tag{2.4}$$

where $\gamma_k = k/(\lambda_1 \operatorname{Tr}(\mathbf{C}^{-1}))$ with $\mathbf{C}_{ij} = \int_{D_1\times D_1} v_i(x)K(x,y)v_j(y)\,\mathrm{d}x\,\mathrm{d}y$ for $1 \leq i, j \leq k$. Here, $\mathbf{P}_{\mathbf{Y}}$ is the orthogonal projection onto the vector space spanned by the columns of $\mathbf{Y}$, σ_j is the jth singular value of $\mathscr{F}$, and v_j is the jth right singular vector of $\mathscr{F}$.

Assume further that $p \geq 4$, then for any $s, t \geq 1$, we have

$$\|\mathscr{F} - \mathbf{P}_{\mathbf{Y}}\mathscr{F}\|_{\text{HS}} \leq \sqrt{1 + t^2 s^2 \frac{3}{\gamma_k} \frac{k(k+p)}{p+1} \sum_{j=1}^{\infty} \frac{\lambda_j}{\lambda_1}} \left(\sum_{j=k+1}^{\infty} \sigma_j^2 \right)^{1/2}, \tag{2.5}$$

with probability $\geq 1 - t^{-p} - \left[se^{-(s^2-1)/2}\right]^{k+p}$.

We remark that the term $\left[se^{-(s^2-1)/2}\right]^{k+p}$ in the statement of Theorem 2.1 is bounded by e^{-s^2} for $s \geq 2$ and $k + p \geq 5$. The term $0 \leq \gamma_k \leq 1$ is discussed in Section 2.1.4 and is bounded by the inverse of the harmonic mean of k eigenvalues of the covariance kernel under some conditions on the kernel eigenvectors and the right singular vectors of the Hilbert–Schmidt operator $\mathscr{F}$ (see Lemma 2.2). In the rest of the section, we prove this theorem.

2.1.1 Three caveats that make the generalization non-trivial

One might imagine that the generalization of the randomized SVD algorithm from matrices to HS operators is trivial, but this is not the case due to three caveats.

First, the randomized SVD on finite matrices always uses matrix-vector products with standard Gaussian random vectors [86]. However, for GPs, one must always have a continuous kernel K in $\mathcal{GP}(0, K)$, which discretizes to a non-standard multivariate Gaussian distribution. Therefore, we must extend [86, Thm. 10.5] to allow for non-standard multivariate Gaussian distributions. The discrete version of our extension is the following:

Corollary 2.1. *Let $\mathbf{A}$ be a real $n_2 \times n_1$ matrix with singular values $\sigma_1 \geq \cdots \geq \sigma_{\min\{n_1,n_2\}}$. Choose a target rank $k \geq 1$ and an oversampling parameter $p \geq 2$. Draw an $n_1 \times (k+p)$ Gaussian matrix, $\mathbf{\Omega}$, with independent columns where each column is i.i.d. from a multivariate Gaussian distribution with mean $(0, \ldots, 0)^\top$ and positive definite covariance matrix $\mathbf{K}$. If $\mathbf{Y} = \mathbf{A}\mathbf{\Omega}$, then the expected approximation error is bounded by*

$$\mathbb{E}\left[\|\mathbf{A} - \mathbf{P}_{\mathbf{Y}}\mathbf{A}\|_{\text{F}}\right] \leq \left(1 + \sqrt{\frac{k+p}{p-1} \sum_{j=n_1-k+1}^{n_1} \frac{\lambda_1}{\lambda_j}}\right) \left(\sum_{j=k+1}^{\infty} \sigma_j^2 \right)^{1/2}, \tag{2.6}$$

where $\lambda_1 \geq \cdots \geq \lambda_{n_1} > 0$ are the eigenvalues of $\mathbf{K}$ and $\mathbf{P}_{\mathbf{Y}}$ is the orthogonal projection onto the vector space spanned by the columns of $\mathbf{Y}$. Assume further that $p \geq 4$, then for any $s, t \geq 1$, we have

$$\|\mathbf{A} - \mathbf{P}_{\mathbf{Y}}\mathbf{A}\|_{\text{F}} \leq \left(1 + ts \cdot \sqrt{\frac{3(k+p)}{p+1} \left(\sum_{j=1}^{n_1} \lambda_j \right) \sum_{j=n_1-k+1}^{n_1} \frac{1}{\lambda_j}}\right) \left(\sum_{j=k+1}^{\infty} \sigma_j^2 \right)^{1/2},$$

with probability $\geq 1 - t^{-p} - [se^{-(s^2-1)/2}]^{k+p}$.

Choosing a covariance matrix $\mathbf{K}$ with eigenvalue decay so that $\lim_{n_1\to\infty} \sum_{j=1}^{n_1} \lambda_j < \infty$ allows $\mathbb{E}[\|\mathbf{\Omega}\|_{\mathrm{F}}^2]$ to remain bounded as $n_1 \to \infty$. This is of interest when applying the randomized SVD algorithm to extremely large matrices and is critical for HS operators. A stronger statement of this result (see Theorem 3.1) shows that prior information on $\mathbf{A}$ can be incorporated into the covariance matrix to achieve lower approximation error than the randomized SVD with standard Gaussian vectors.

Secondly, we need an additional essential assumption. The kernel in $\mathcal{GP}(0, K)$ is "reasonable" for learning $\mathscr{F}$, where reasonableness is measured by the quantity γ_k in Theorem 2.1. If the first k right singular functions of the HS operator $v_1, \ldots, v_k$ are spanned by the first $k+m$ eigenfunctions of K $\psi_1, \ldots, \psi_{k+m}$, for some $m \in \mathbb{N}$, then (see Equation (2.9) and Lemma 2.2)

$$\frac{1}{k}\sum_{j=1}^{k}\frac{\lambda_1}{\lambda_j} \leq \frac{1}{\gamma_k} \leq \frac{1}{k}\sum_{j=m+1}^{k+m}\frac{\lambda_1}{\lambda_j}.$$

In the matrix setting, this assumption always holds with $m = n_1 - k$ (see Corollary 2.1) and one can have $\gamma_k = 1$ when $\lambda_1 = \cdots = \lambda_{n_1}$ [86, Thm. 10.5].

Finally, probabilistic error bounds for the randomized SVD in [86] are derived using tail bounds for functions of standard Gaussian matrices [121, Sec. 5.1]. Unfortunately, we are not aware of tail bounds for non-standard Gaussian quasimatrices. This results in a weaker bounds by a factor of $\sqrt{k+p}$ in Corollary 2.1 compared to [86, Thm. 10.7].

2.1.2 Deterministic error bound

Apart from the three caveats, the proof of Theorem 2.1 follows the outline of the argument in [86, Thm. 10.5]. We define two quasimatrices $\mathbf{U}$ and $\mathbf{V}$ containing the left and right singular functions of $\mathscr{F}$ so that the jth column of $\mathbf{V}$ is v_j. We also denote by $\mathbf{\Sigma}$ the infinite diagonal matrix with the singular values of $\mathscr{F}$, *i.e.*, $\sigma_1 \geq \sigma_2 \geq \cdots \geq 0$, on the diagonal. Finally, for a fixed $k \geq 1$, we define the $D_1 \times k$ quasimatrix as the truncation of $\mathbf{V}$ after the first k columns and $\mathbf{V}_2$ as the remainder. Similarly, we split $\mathbf{\Sigma}$ into two parts:

$$\mathbf{\Sigma} = \begin{array}{c} \begin{array}{cc} k & \infty \end{array} \\ \left(\begin{array}{cc} \mathbf{\Sigma}_1 & 0 \\ 0 & \mathbf{\Sigma}_2 \end{array}\right) \end{array} \begin{array}{c} k \\ \infty \end{array} .$$

We are ready to prove an infinite dimensional analogue of [86, Thm. 9.1] for HS operators.

Theorem 2.2 (Deterministic error bound)**.** *Let $\mathscr{F} : L^2(D_1) \to L^2(D_2)$ be an HS operator with SVD given in Equation* (1.7)*. Let $\mathbf{\Omega}$ be a $D_1 \times k$ quasimatrix and $\mathbf{Y} = \mathscr{F}\mathbf{\Omega}$. If $\mathbf{\Omega}_1 = \mathbf{V}_1^*\mathbf{\Omega}$ and $\mathbf{\Omega}_2 = \mathbf{V}_2^*\mathbf{\Omega}$, then assuming $\mathbf{\Omega}_1$ has full rank, we have*

$$\|\mathscr{F} - \mathbf{P}_\mathbf{Y}\mathscr{F}\|_{\text{HS}}^2 \leq \|\mathbf{\Sigma}_2\|_{\text{HS}}^2 + \|\mathbf{\Sigma}_2\mathbf{\Omega}_2\mathbf{\Omega}_1^\dagger\|_{\text{HS}}^2,$$

where $\mathbf{P}_\mathbf{Y} = \mathbf{Y}(\mathbf{Y}^*\mathbf{Y})^\dagger\mathbf{Y}^*$ is the orthogonal projection onto the space spanned by the columns of $\mathbf{Y}$ and $\mathbf{\Omega}_1^\dagger = (\mathbf{\Omega}_1^*\mathbf{\Omega}_1)^{-1}\mathbf{\Omega}_1^*$.

Proof. First, note that because $\mathbf{U}\mathbf{U}^*$ is the orthonormal projection onto the range of $\mathscr{F}$ and $\mathbf{U}$ is a basis for the range, we have

$$\|\mathscr{F} - \mathbf{P}_\mathbf{Y}\mathscr{F}\|_{\text{HS}} = \|\mathbf{U}\mathbf{U}^*\mathscr{F} - \mathbf{P}_\mathbf{Y}\mathbf{U}\mathbf{U}^*\mathscr{F}\|_{\text{HS}}.$$

By Parseval's theorem [194, Thm. 4.18], we have

$$\|\mathbf{U}\mathbf{U}^*\mathscr{F} - \mathbf{P}_\mathbf{Y}\mathbf{U}\mathbf{U}^*\mathscr{F}\|_{\text{HS}} = \|\mathbf{U}^*\mathbf{U}\mathbf{U}^*\mathscr{F} - \mathbf{U}^*\mathbf{P}_\mathbf{Y}\mathbf{U}\mathbf{U}^*\mathscr{F}\mathbf{V}\|_{\text{HS}}.$$

Moreover, we have the equality $\|\mathscr{F} - \mathbf{P}_\mathbf{Y}\mathscr{F}\|_{\text{HS}} = \|(\mathbf{I} - \mathbf{P}_{\mathbf{U}^*\mathbf{Y}})\mathbf{U}^*\mathscr{F}\mathbf{V}\|_{\text{HS}}$ because the inner product $\langle \sum_{j=1}^\infty \alpha_j u_j, \sum_{j=1}^\infty \beta u_j \rangle = 0$ if and only if $\sum_{j=1}^\infty \alpha_j\beta_j = 0$. We now take $\mathbf{A} = \mathbf{U}^*\mathscr{F}\mathbf{V}$, which is a bounded infinite matrix such that $\|\mathbf{A}\|_{\text{F}} = \|\mathscr{F}\|_{\text{HS}} < \infty$. The statement of the theorem immediately follows from the proof of [86, Thm. 9.1]. □

This theorem shows that the bound on the approximation error $\|\mathscr{F} - \mathbf{P}_\mathbf{Y}\mathscr{F}\|_{\text{HS}}$ depends on the singular values of the HS operator and the test matrix $\mathbf{\Omega}$.

2.1.3 Probability distribution of $\mathbf{\Omega}_1$

If the columns of $\mathbf{\Omega}$ are independent and identically distributed as $\mathcal{GP}(0, K)$, then the matrix $\mathbf{\Omega}_1$ in Theorem 2.2 is of size $k \times \ell$ with entries that follow a Gaussian distribution. To see this, note that

$$\mathbf{\Omega}_1 = \mathbf{V}_1^*\mathbf{\Omega} = \begin{pmatrix} \langle v_1, \omega_1 \rangle & \cdots & \langle v_1, \omega_\ell \rangle \\ \vdots & \ddots & \vdots \\ \langle v_k, \omega_1 \rangle & \cdots & \langle v_k, \omega_\ell \rangle \end{pmatrix}, \quad \omega_j \sim \mathcal{GP}(0, K).$$

If $\omega \sim \mathcal{GP}(0, K)$ with K given in Equation (1.8), then we find that

$$\langle v, \omega \rangle \sim \mathcal{N}\left(0, \sum_{j=1}^\infty \lambda_j \langle v, \psi_j \rangle^2\right)$$

so we conclude that $\mathbf{\Omega}_1$ has Gaussian entries with zero mean. Finding the covariances between the entries is more involved.

Lemma 2.1. *With the same setup as Theorem 2.2, suppose that the columns of* $\mathbf{\Omega}$ *are independent and identically distributed as* $\mathcal{GP}(0, K)$*. Then, the matrix* $\mathbf{\Omega}_1 = \mathbf{V}_1^*\mathbf{\Omega}$ *in Theorem 2.2 has independent columns and each column is identically distributed as a multivariate Gaussian with positive definite covariance matrix* $\mathbf{C}$ *given by*

$$\mathbf{C}_{ij} = \int_{D_1 \times D_1} v_i(x) K(x, y) v_j(y) \, \mathrm{d}x \, \mathrm{d}y, \quad 1 \le i, j \le k, \tag{2.7}$$

where v_i *is the* i*th column of* $\mathbf{V}_1$*.*

Proof. We already know that the entries are Gaussian with mean 0. Moreover, the columns are independent because $\omega_1, \ldots, \omega_\ell$ are independent. Therefore, we focus on the covariance matrix. Let $1 \le i, i' \le k$, $1 \le j, j' \le \ell$, then since $\mathbb{E}[\langle v_i, \omega_j \rangle] = 0$ we have

$$\operatorname{cov}(\langle v_i, \omega_j \rangle, \langle v_{i'}, \omega_{j'} \rangle) = \mathbb{E}\left[\langle v_i, \omega_j \rangle \langle v_{i'}, \omega_{j'} \rangle\right] = \mathbb{E}\left[X_{ij} X_{i'j'}\right],$$

where $X_{ij} = \langle v_i, \omega_j \rangle$. Since $\langle v_i, \omega_j \rangle \sim \sum_{n=1}^{\infty} \sqrt{\lambda_n} c_n^{(j)} \langle v_i, \psi_n \rangle$, where $c_n^{(j)} \sim \mathcal{N}(0, 1)$, we have

$$\operatorname{cov}(\langle v_i, \omega_j \rangle, \langle v_{i'}, \omega_{j'} \rangle) = \mathbb{E}\left[\lim_{m_1, m_2 \to \infty} X_{ij}^{m_1} X_{i'j'}^{m_2}\right], \quad X_{ij}^{m_1} \coloneqq \sum_{n=1}^{m_1} \sqrt{\lambda_n} c_n^{(j)} \langle v_i, \psi_n \rangle.$$

We first show that $\lim_{m_1, m_2 \to \infty} \left| \mathbb{E}\big[X_{ij}^{m_1} X_{i'j'}^{m_2}\big] - \mathbb{E}[X_{ij} X_{i'j'}] \right| = 0$. For any $m_1, m_2 \ge 1$, we have by the triangle inequality,

$$\begin{aligned}
\left| \mathbb{E}\big[X_{ij}^{m_1} X_{i'j'}^{m_2}\big] - \mathbb{E}[X_{ij} X_{i'j'}] \right| &\le \mathbb{E}\left[\left| X_{ij}^{m_1} X_{i'j'}^{m_2} - X_{ij} X_{i'j'} \right|\right] \\
&\le \mathbb{E}\left[\left| (X_{ij}^{m_1} - X_{ij}) X_{i'j'}^{m_2} \right|\right] + \mathbb{E}\left[\left| X_{ij} (X_{i'j'}^{m_2} - X_{i'j'}) \right|\right] \\
&\le \mathbb{E}\left[\left| X_{ij}^{m_1} - X_{ij} \right|^2\right]^{\frac{1}{2}} \mathbb{E}\left[\left| X_{i'j'}^{m_2} \right|^2\right]^{\frac{1}{2}} + \mathbb{E}\left[\left| X_{i'j'} - X_{i'j'}^{m_2} \right|^2\right]^{\frac{1}{2}} \mathbb{E}\left[\left| X_{ij} \right|^2\right]^{\frac{1}{2}},
\end{aligned}$$

where the last inequality follows from the Cauchy–Schwarz inequality. We now set out to show that both terms in the last inequality converge to zero as $m_1, m_2 \to \infty$. The terms $\mathbb{E}[|X_{i'j'}^{m_2}|^2]$ and $\mathbb{E}[|X_{ij}|^2]$ are bounded by $\sum_{n=1}^{\infty} \lambda_n < \infty$, using the Cauchy–Schwarz inequality. Moreover, we have

$$\mathbb{E}\left[\left| X_{ij}^{m_1} - X_{ij} \right|^2\right] = \mathbb{E}\left[\left| \sum_{n=m_1+1}^{\infty} \sqrt{\lambda_n} c_n^{(j)} \langle v_i, \psi_n \rangle \right|^2\right] \le \sum_{n=m_1+1}^{\infty} \lambda_n \xrightarrow[m_1 \to \infty]{} 0,$$

because $X_{ij} - X_{ij}^{m_1} \sim \mathcal{N}(0, \sum_{n=m_1+1}^{\infty} \lambda_n \langle v_i, \psi_n \rangle^2)$. Then, we find that $\operatorname{cov}(X_{ij}, X_{i'j'}) = \lim_{m_1, m_2 \to \infty} \mathbb{E}[X_{ij}^{m_1} X_{i'j'}^{m_2}]$ and we obtain

$$\begin{aligned}
\operatorname{cov}(X_{ij}, X_{i'j'}) &= \lim_{m_1, m_2 \to \infty} \mathbb{E}\left[\sum_{n=1}^{m_1} \sum_{n'=1}^{m_2} \sqrt{\lambda_n \lambda_{n'}} c_n^{(j)} c_{n'}^{(j')} \langle v_i, \psi_n \rangle \langle v_{i'}, \psi_{n'} \rangle\right] \\
&= \lim_{m_1, m_2 \to \infty} \sum_{n=1}^{m_1} \sum_{n'=1}^{m_2} \sqrt{\lambda_n \lambda_{n'}} \mathbb{E}[c_n^{(j)} c_{n'}^{(j')}] \langle v_i, \psi_n \rangle \langle v_{i'}, \psi_{n'} \rangle.
\end{aligned}$$

The latter expression is zero if $n \neq n'$ or $j \neq j'$ because then $c_n^{(j)}$ and $c_{n'}^{(j')}$ are independent random variables with mean 0. Since $\mathbb{E}[(c_n^{(j)})^2] = 1$, we have

$$\operatorname{cov}(X_{ij}, X_{i'j'}) = \begin{cases} \sum_{n=1}^{\infty} \lambda_n \langle v_i, \psi_n \rangle \langle v_{i'}, \psi_n \rangle, & j = j', \\ 0, & \text{otherwise.} \end{cases}$$

The result follows as the infinite sum is equal to the integral in Equation (2.7). To see that $\mathbf{C}$ is positive definite, let $a \in \mathbb{R}^k$, then $a^*\mathbf{C}a = \mathbb{E}[Z_a^2] \geq 0$, where $Z_a \sim \mathcal{N}(0, \sum_{n=1}^{\infty} \lambda_n \langle a_1 v_1 + \cdots + a_k v_k, \psi_n \rangle^2)$. Moreover, $a^*\mathbf{C}a = 0$ implies that $a = 0$ because $v_1, \ldots, v_k$ are orthonormal and $\{\psi_n\}$ is an orthonormal basis of $L^2(D_1)$. $\square$

Lemma 2.1 gives the distribution of the matrix $\mathbf{\Omega}_1$, which is essential to prove Theorem 2.1 in Section 2.1.6. In particular, $\mathbf{\Omega}_1$ has independent columns that are each distributed as a multivariate Gaussian with covariance matrix given in Equation (2.7).

2.1.4 Quality of the covariance kernel

To investigate the quality of the kernel, we introduce the Wishart distribution, which is a family of probability distributions over symmetric and nonnegative-definite matrices that often appear in the context of covariance matrices [239]. If $\mathbf{\Omega}_1$ is a $k \times \ell$ random matrix with independent columns, where each column is a multivariate Gaussian distribution with mean $(0, \ldots, 0)^\top$ and covariance $\mathbf{C}$, then $\mathbf{A} = \mathbf{\Omega}_1 \mathbf{\Omega}_1^*$ has a Wishart distribution [239]. We write $\mathbf{A} \sim W_k(\ell, \mathbf{C})$. We note that $\|\mathbf{\Omega}_1^\dagger\|_{\mathrm{F}}^2 = \operatorname{Tr}[(\mathbf{\Omega}_1^\dagger)^* \mathbf{\Omega}_1^\dagger] = \operatorname{Tr}(\mathbf{A}^{-1})$, where the second equality holds with probability one because the matrix $\mathbf{A} = \mathbf{\Omega}_1 \mathbf{\Omega}_1^*$ is invertible with probability one (see [156, Thm. 3.1.4]). By [156, Thm. 3.2.12] for $\ell - k \geq 2$, we have $\mathbb{E}[\mathbf{A}^{-1}] = \frac{1}{\ell - k - 1} \mathbf{C}^{-1}$, $\mathbb{E}[\operatorname{Tr}(\mathbf{A}^{-1})] = \operatorname{Tr}(\mathbf{C}^{-1})/(\ell - k - 1)$, and conclude that

$$\mathbb{E}\left[\|\mathbf{\Omega}_1^\dagger\|_{\mathrm{F}}^2\right] = \frac{1}{\gamma_k \lambda_1} \frac{k}{\ell - k - 1}, \quad \gamma_k \coloneqq \frac{k}{\lambda_1 \operatorname{Tr}(\mathbf{C}^{-1})}. \tag{2.8}$$

The quantity γ_k can be viewed as measuring the quality of the covariance kernel K for learning the HS operator $\mathscr{F}$ (see Theorem 2.1). First, $1 \leq \gamma_k < \infty$ as $\mathbf{C}$ is symmetric positive definite. Moreover, for $1 \leq j \leq k$, the jth largest eigenvalue of $\mathbf{C}$ is bounded by the jth largest eigenvalue of K as $\mathbf{C}$ is a principal submatrix of $\mathbf{V}^* K \mathbf{V}$ [104, Sec. III.5]. Therefore, the following inequality holds,

$$\frac{1}{k} \sum_{j=1}^{k} \frac{\lambda_1}{\lambda_j} \leq \frac{1}{\gamma_k} < \infty, \tag{2.9}$$

and the harmonic mean of the first k scaled eigenvalues of K is a lower bound for $1/\gamma_k$. In the ideal situation, the eigenfunctions of K are the right singular functions of $\mathscr{F}$, *i.e.*, $\psi_n = v_n$, $\mathbf{C}$ is a diagonal matrix with entries $\lambda_1, \ldots, \lambda_k$, and $\gamma_k = k/(\sum_{j=1}^{k} \lambda_1/\lambda_j)$ is as small as possible.

We now provide a useful upper bound on γ_k in a more general setting.

Lemma 2.2. *Let $\mathbf{V}_1$ be a $D_1 \times k$ quasimatrix with orthonormal columns and assume that there exists $m \in \mathbb{N}$ such that the columns of $\mathbf{V}_1$ are spanned by the first $k+m$ eigenvectors of the continuous positive definite kernel $K : D_1 \times D_1 \to \mathbb{R}$. Then*

$$\frac{1}{\gamma_k} \leq \frac{1}{k} \sum_{j=m+1}^{k+m} \frac{\lambda_1}{\lambda_j},$$

where $\lambda_1 \geq \lambda_2 \geq \cdots > 0$ are the eigenvalues of K. This bound is tight in the sense that the inequality can be attained as an equality.

Proof. Let $\mathbf{Q} = [v_1 \,|\, \cdots \,|\, v_k \,|\, q_{k+1} \,|\, \cdots \,|\, q_{k+m}]$ be a quasimatrix with orthonormal columns whose columns form an orthonormal basis for $\operatorname{Span}(\psi_1, \ldots, \psi_{k+m})$. Then, $\mathbf{Q}$ is an invariant space of K and $\mathbf{C}$ is a principal submatrix of $\mathbf{Q}^* K \mathbf{Q}$, which has eigenvalues $\lambda_1 \geq \cdots \geq \lambda_{k+m}$. By [104, Thm. 6.46] the k eigenvalues of $\mathbf{C}$, denoted by $\mu_1, \ldots, \mu_k$, are greater than the first $k+m$ eigenvalues of K: $\mu_j \geq \lambda_{m+j}$ for $1 \leq j \leq k$, and the result follows as the trace of a matrix is the sum of its eigenvalues. □

2.1.5 Probabilistic error bounds

As discussed in Section 2.1.1, we need to extend the probability bounds of the randomized SVD to allow for non-standard Gaussian random vectors. The following lemma is a generalization of [86, Thm. A.7].

Lemma 2.3. *Let $k, \ell \geq 1$ such that $\ell - k \geq 4$ and $\mathbf{\Omega}_1$ be a $k \times \ell$ random matrix with independent columns such that each column has mean $(0, \ldots, 0)^\top$ and positive definite covariance $\mathbf{C}$. For all $t \geq 1$, we have*

$$\mathbb{P}\left\{ \|\mathbf{\Omega}_1^\dagger\|_{\mathrm{F}}^2 > \frac{3 \operatorname{Tr}(\mathbf{C}^{-1})}{\ell - k + 1} \cdot t^2 \right\} \leq t^{-(\ell-k)}.$$

Proof. Since $\mathbf{\Omega}_1 \mathbf{\Omega}_1^* \sim W_k(\ell, \mathbf{C})$, the reciprocals of its diagonal elements follow a scaled chi-square distribution [156, Thm. 3.2.12], *i.e.*,

$$\frac{((\mathbf{\Omega}_1 \mathbf{\Omega}_1^*)^{-1})_{jj}}{(\mathbf{C}^{-1})_{jj}} \sim X_j^{-1}, \quad X_j \sim \chi^2_{\ell-k+1}, \quad 1 \leq j \leq k.$$

Let $Z = \|\mathbf{\Omega}_1^\dagger\|_{\mathrm{F}}^2 = \mathrm{Tr}[(\mathbf{\Omega}_1\mathbf{\Omega}_1^*)^{-1}]$ and $q = (\ell - k)/2$. Following the proof of [86, Thm. A.7], we have the inequality

$$\mathbb{P}\left\{|Z| \geq \frac{3\,\mathrm{Tr}(\mathbf{C}^{-1})}{\ell - k + 1} \cdot t^2\right\} \leq \left[\frac{3\,\mathrm{Tr}(\mathbf{C}^{-1})}{\ell - k + 1} \cdot t^2\right]^{-q} \mathbb{E}\left[|Z|^q\right], \quad t \geq 1.$$

Moreover, by the Minkowski inequality, we have

$$\left(\mathbb{E}\left[|Z^q|\right]\right)^{1/q} = \left(\mathbb{E}\left[\left|\sum_{j=1}^{k} [\mathbf{C}^{-1}]_{jj} X_j^{-1}\right|^q\right]\right)^{1/q} \leq \sum_{j=1}^{k} [\mathbf{C}^{-1}]_{jj} \mathbb{E}\left[|X_j^{-1}|^q\right]^{1/q} \leq \frac{3\,\mathrm{Tr}(\mathbf{C}^{-1})}{\ell - k + 1},$$

where the last inequality is from [86, Lem. A.9]. The result follows from the argument in the proof of [86, Thm. A.7]. □

Under the assumption of Lemma 2.2, we find that Lemma 2.3 gives the following bound:

$$\mathbb{P}\left\{\|\mathbf{\Omega}_1^\dagger\|_{\mathrm{F}} > t \cdot \sqrt{\frac{3}{\ell - k + 1} \sum_{j=m+1}^{k+m} \lambda_j^{-1}}\right\} \leq t^{-(\ell-k)}.$$

In particular, in the finite dimensional case when $\lambda_1 = \cdots = \lambda_n = 1$, we recover the probabilistic bound found in [86, Thm. A.7].

To obtain the probability statement found in Equation (2.11) we require control of the tail of the distribution of a Gaussian quasimatrix with non-standard covariance kernel (see Section 2.1.6). In the theory of the randomized SVD, one relies on the concentration of measure results [86, Prop. 10.3]. However, we need to employ a different strategy and instead directly bound the HS norm of $\mathbf{\Omega}_2$. One difficulty is that the norm of this matrix must be controlled for large dimensions n, which leads to a weaker probability bound than [86]. While it is possible to apply Markov's inequality to obtain deviation bounds, we highlight that Lemma 2.4 provides a Chernoff-type bound, *i.e.*, exponential decay of the tail distribution of $\|\mathbf{\Omega}_2\|_{\mathrm{HS}}$, which is crucial to approximate Green's functions (see Section 2.2.4.3).

Lemma 2.4. *With the same notation as in Theorem 2.2, let $\ell \geq k \geq 1$. For all $s \geq 1$ we have*

$$\mathbb{P}\left\{\|\mathbf{\Omega}_2\|_{\mathrm{HS}}^2 > \ell s^2\,\mathrm{Tr}(K)\right\} \leq \left[s e^{-(s^2-1)/2}\right]^{\ell}.$$

Proof. We first remark that

$$\|\mathbf{\Omega}_2\|_{\mathrm{HS}}^2 \leq \|\mathbf{\Omega}\|_{\mathrm{HS}}^2 = \sum_{j=1}^{\ell} Z_j, \quad Z_j \coloneqq \|\omega_j\|_{L^2(D_1)}^2, \tag{2.10}$$

where the Z_j are i.i.d. because $\omega_j \sim \mathcal{GP}(0, K)$ are i.i.d. For $1 \leq j \leq \ell$, we have (c.f. Section 1.8),

$$\omega_j = \sum_{m=1}^{\infty} c_m^{(j)} \sqrt{\lambda_m} \psi_m,$$

where $c_m^{(j)} \sim \mathcal{N}(0,1)$ are i.i.d. for $m \geq 1$ and $1 \leq j \leq \ell$. First, since the series in Equation (2.10) converges absolutely, we have

$$Z_j = \sum_{m=1}^{\infty} (c_m^{(j)})^2 \lambda_m = \lim_{N \to \infty} \sum_{m=1}^{N} X_m, \quad X_m = (c_m^{(j)})^2 \lambda_m,$$

where the X_m are independent random variables and $X_m \sim \lambda_m \chi^2$ for $1 \leq m \leq N$. Here, χ^2 denotes the chi-squared distribution [155, Chapt. 4.3].

Let $N \geq 1$ and $0 < \theta < 1/(2\operatorname{Tr}(K))$, we can bound the moment generating function of $\sum_{m=1}^{N} X_m$ as

$$\begin{aligned} \mathbb{E}\left[e^{\theta \sum_{m=1}^{N} X_m}\right] = \prod_{m=1}^{N} \mathbb{E}\left[e^{\theta X_m}\right] = \prod_{m=1}^{N} (1 - 2\theta\lambda_m)^{-1/2} &\leq \left(1 - 2\theta \sum_{m=1}^{N} \lambda_m\right)^{-1/2} \\ &\leq (1 - 2\theta \operatorname{Tr}(K))^{-1/2}, \end{aligned}$$

because X_m/λ_m are independent random variables that follow a chi-squared distribution. Using the monotone convergence theorem, we have

$$\mathbb{E}\left[e^{\theta Z_j}\right] \leq (1 - 2\theta \operatorname{Tr}(K))^{-1/2}.$$

Let $\tilde{s} \geq 0$ and $0 < \theta < 1/(2\operatorname{Tr}(K))$. By the Chernoff bound [41, Thm. 1], we obtain

$$\begin{aligned} \mathbb{P}\left\{\|\boldsymbol{\Omega}_2\|_{\mathrm{HS}}^2 > \ell(1+\tilde{s})\operatorname{Tr}(K)\right\} &\leq e^{-(1+\tilde{s})\operatorname{Tr}(K)\ell\theta} \mathbb{E}\left[e^{\theta Z_j}\right]^{\ell} \\ &= e^{-(1+\tilde{s})\operatorname{Tr}(K)\ell\theta} (1 - 2\theta \operatorname{Tr}(K))^{-\ell/2}. \end{aligned}$$

We can minimize this upper bound over $0 < \theta < 1/(2\operatorname{Tr}(K))$ by choosing $\theta = \tilde{s}/(2(1+\tilde{s})\operatorname{Tr}(K))$, which gives

$$\mathbb{P}\left\{\|\boldsymbol{\Omega}_2\|_{\mathrm{HS}}^2 > \ell(1+\tilde{s})\operatorname{Tr}(K)\right\} \leq (1+\tilde{s})^{\ell/2} e^{-\ell\tilde{s}/2}.$$

Choosing $s = \sqrt{1+\tilde{s}} \geq 1$ concludes the proof. □

Lemma 2.4 can be refined further to take into account the interaction between the Hilbert–Schmidt operator $\mathscr{F}$ and the covariance kernel K (see Lemma 3.1).

2.1.6 Randomized SVD algorithm for HS operators

We first prove an intermediary result, which generalizes [86, Prop. 10.1] to HS operators. Note that one may obtain sharper bounds using a suitably chosen covariance kernels that yields a lower approximation error (see Chapter 3).

Lemma 2.5. *Let $\mathbf{\Sigma}_2$, $\mathbf{V}_2$, and $\mathbf{\Omega}$ be defined as in Theorem 2.2, and $\mathbf{T}$ be an $\ell \times k$ matrix, where $\ell \geq k \geq 1$. Then,*

$$\mathbb{E}\left[\|\mathbf{\Sigma}_2\mathbf{V}_2^*\mathbf{\Omega}\mathbf{T}\|_{\mathrm{HS}}^2\right] \leq \lambda_1\|\mathbf{\Sigma}_2\|_{\mathrm{HS}}^2\|\mathbf{T}\|_{\mathrm{F}}^2,$$

where λ_1 is the first eigenvalue of K.

Proof. Let $\mathbf{T} = \mathbf{U_T}\mathbf{D_T}\mathbf{V_T^*}$ be the SVD of $\mathbf{T}$. If $\{v_{\mathbf{T},i}\}_{i=1}^k$ are the columns of $\mathbf{V_T}$, then

$$\mathbb{E}\left[\|\mathbf{\Sigma}_2\mathbf{V}_2^*\mathbf{\Omega}\mathbf{T}\|_{\mathrm{HS}}^2\right] = \sum_{i=1}^{k}\mathbb{E}\left[\|\mathbf{\Sigma}_2\mathbf{\Omega}_2\mathbf{U_T}\mathbf{D_T}\mathbf{V_T^*}v_{\mathbf{T},i}\|_2^2\right],$$

where $\mathbf{\Omega}_2 = \mathbf{V}_2^*\mathbf{\Omega}$. Therefore, we have

$$\mathbb{E}\left[\|\mathbf{\Sigma}_2\mathbf{\Omega}_2\mathbf{T}\|_{\mathrm{HS}}^2\right] = \sum_{i=1}^{k}((\mathbf{D_T})_{ii})^2\mathbb{E}\left[\|\mathbf{\Sigma}_2\mathbf{\Omega}_2\mathbf{U_T}(:,i)\|_2^2\right].$$

Moreover, using the monotone convergence theorem for non-negative random variables, we have

$$\begin{aligned}\mathbb{E}\left[\|\mathbf{\Sigma}_2\mathbf{\Omega}_2\mathbf{U_T}(:,i)\|_2^2\right] &= \mathbb{E}\left[\sum_{n=1}^{\infty}\sum_{j=1}^{\ell}\sigma_{k+n}^2\left|\mathbf{\Omega}_2(n,j)\right|^2\mathbf{U_T}(j,i)^2\right] \\ &= \sum_{n=1}^{\infty}\sum_{j=1}^{\ell}\sigma_{k+n}^2\mathbf{U_T}(j,i)^2\mathbb{E}\left[\left|\mathbf{\Omega}_2(n,j)\right|^2\right],\end{aligned}$$

where $\sigma_{k+1}, \sigma_{k+2}, \ldots$ are the diagonal elements of $\mathbf{\Sigma}_2$. Then, the quasimatrix $\mathbf{\Omega}_2$ has independent columns and, using Lemma 2.1, we have

$$\mathbb{E}\left[\left|\mathbf{\Omega}_2(n,j)\right|^2\right] = \int_{D_1\times D_1} v_{k+n}(x)K(x,y)v_{k+n}(y)\,\mathrm{d}x\,\mathrm{d}y,$$

where v_{k+n} is the nth column of $\mathbf{V}_2$. Then, $\mathbb{E}\left[\left|\mathbf{\Omega}_2(n,j)\right|^2\right] \leq \lambda_1$, as $\mathbb{E}\left[\left|\mathbf{\Omega}_2(n,j)\right|^2\right]$ is written as a Rayleigh quotient. Finally, we have

$$\mathbb{E}\left[\|\mathbf{\Sigma}_2\mathbf{V}_2^*\mathbf{\Omega}\mathbf{T}\|_{\mathrm{HS}}^2\right] \leq \lambda_1\sum_{i=1}^{k}((\mathbf{D_T})_{ii})^2\sum_{j=1}^{\ell}\mathbf{U_T}(j,i)^2\sum_{n=1}^{\infty}\sigma_{k+n}^2 = \lambda_1\|\mathbf{T}\|_{\mathrm{F}}^2\|\mathbf{\Sigma}_2\|_{\mathrm{HS}}^2,$$

by orthonormality of the columns on $\mathbf{U_T}$. □

We are now ready to prove Theorem 2.1, which shows that the randomized SVD can be generalized to HS operators.

Proof of Theorem 2.1. Let $\mathbf{\Omega}_1, \mathbf{\Omega}_2$ be the quasimatrices defined in Theorem 2.2. The $k \times (k+p)$ matrix $\mathbf{\Omega}_1$ has full rank with probability one and by Theorem 2.2, we have

$$\mathbb{E}\left[\|(\mathbf{I} - \mathbf{P}_\mathbf{Y})\mathscr{F}\|_{\mathrm{HS}}\right] \leq \mathbb{E}\left[\left(\|\mathbf{\Sigma}_2\|_{\mathrm{HS}}^2 + \|\mathbf{\Sigma}_2\mathbf{\Omega}_2\mathbf{\Omega}_1^\dagger\|_{\mathrm{HS}}^2\right)^{1/2}\right] \leq \|\mathbf{\Sigma}_2\|_{\mathrm{HS}} + \mathbb{E}\|\mathbf{\Sigma}_2\mathbf{\Omega}_2\mathbf{\Omega}_1^\dagger\|_{\mathrm{HS}}$$
$$\leq \|\mathbf{\Sigma}_2\|_{\mathrm{HS}} + \mathbb{E}\left[\|\mathbf{\Sigma}_2\mathbf{\Omega}_2\|_{\mathrm{HS}}^2\right]^{1/2} \mathbb{E}\left[\|\mathbf{\Omega}_1^\dagger\|_{\mathrm{F}}^2\right]^{1/2},$$

where the last inequality follows from Cauchy–Schwarz inequality. Then, combining Lemma 2.5 and Equation (2.8), we have

$$\mathbb{E}\left[\|\mathbf{\Sigma}_2\mathbf{\Omega}_2\|_{\mathrm{HS}}^2\right] \leq \lambda_1(k+p)\|\mathbf{\Sigma}_2\|_{\mathrm{HS}}^2 \quad \text{and} \quad \mathbb{E}\left[\|\mathbf{\Omega}_1\|_{\mathrm{F}}^2\right] \leq \frac{1}{\gamma_k \lambda_1}\frac{k}{p-1},$$

where γ_k is defined in Section 2.1.4. The observation that $\|\mathbf{\Sigma}_2\|_{\mathrm{HS}}^2 = \sum_{j=k+1}^{\infty} \sigma_j^2$ concludes the proof of Equation (2.4).

For the probabilistic bound in Equation (2.5), we note that by Theorem 2.2 we have,

$$\|\mathscr{F} - \mathbf{P}_\mathbf{Y}\mathscr{F}\|_{\mathrm{HS}}^2 \leq \|\mathbf{\Sigma}_2\|_{\mathrm{HS}}^2 + \|\mathbf{\Sigma}_2\mathbf{\Omega}_2\mathbf{\Omega}_1^\dagger\|_{\mathrm{HS}}^2 \leq (1 + \|\mathbf{\Omega}_2\|_{\mathrm{HS}}^2\|\mathbf{\Omega}_1^\dagger\|_{\mathrm{F}}^2)\|\mathbf{\Sigma}_2\|_{\mathrm{HS}}^2,$$

where the second inequality uses the submultiplicativity of the HS norm. The bound follows from bounding $\|\mathbf{\Omega}_1^\dagger\|_{\mathrm{F}}^2$ and $\|\mathbf{\Omega}_2\|_{\mathrm{HS}}^2$ using Lemmas 2.3 and 2.4, respectively. □

Remark 2.1. *The expectation bound* (2.4) *in Theorem 2.1 does not control the square of the HS norm and therefore cannot be used to obtain an expectation bound for the randomized scheme for learning Green's functions described in Section 2.2.*

The following proposition provides an expectation bound for the randomized SVD of the HS norm squared.

Proposition 2.1. *With the notations of Theorem 2.1, we have*

$$\mathbb{E}\left[\|\mathscr{F} - \mathbf{P}_\mathbf{Y}\mathscr{F}\|_{\mathrm{HS}}^2\right] \leq \left(1 + \frac{3\sqrt{2}}{\gamma_k}\frac{k(k+p)}{p+1}\sum_{j=1}^{\infty}\frac{\lambda_j}{\lambda_1}\right)\sum_{j=k+1}^{\infty}\sigma_j^2.$$

Proof. Let $\mathbf{\Omega}_1, \mathbf{\Omega}_2$ be the quasimatrices defined in Theorem 2.2. We combine Theorem 2.2 with the submultiplicativity of the HS norm and Cauchy–Schwarz inequality to obtain

$$\mathbb{E}[\|\mathscr{F} - \mathbf{P}_\mathbf{Y}\mathscr{F}\|_{\mathrm{HS}}^2] \leq \|\mathbf{\Sigma}_2\|_{\mathrm{HS}}^2 + \mathbb{E}[\|\mathbf{\Sigma}_2\mathbf{\Omega}_2\mathbf{\Omega}_1^\dagger\|_{\mathrm{HS}}^2] \leq (1 + \mathbb{E}[\|\mathbf{\Omega}_2\|_{\mathrm{HS}}^4]^{\frac{1}{2}}\mathbb{E}[\|\mathbf{\Omega}_1^\dagger\|_{\mathrm{F}}^4]^{\frac{1}{2}})\|\mathbf{\Sigma}_2\|_{\mathrm{HS}}^2.$$

We can then control both terms $\mathbb{E}[\|\mathbf{\Omega}_2\|_{\mathrm{HS}}^4]^{1/2}$ and $\mathbb{E}[\|\mathbf{\Omega}_1^\dagger\|_{\mathrm{F}}^4]^{1/2}$ independently.

First, since $\mathbf{\Omega}_1\mathbf{\Omega}_1^* \sim W_k(k+p, \mathbf{C})$, where $\mathbf{C}$ is defined in Lemma 2.1, we have (cf. the proof of Lemma 2.3)

$$\frac{((\mathbf{\Omega}_1\mathbf{\Omega}_1^*)^{-1})_{jj}}{(\mathbf{C}^{-1})_{jj}} \sim X_j^{-1}, \quad X_j \sim \chi^2_{p+1}, \quad 1 \le j \le k.$$

Therefore,

$$\begin{aligned}\mathbb{E}[\|\mathbf{\Omega}_1^\dagger\|^4]^{1/2} = \mathbb{E}[\mathrm{Tr}((\mathbf{\Omega}_1\mathbf{\Omega}_1^*)^{-1})^2]^{1/2} &= \mathbb{E}\left[\left(\sum_{j=1}^k (\mathbf{C}^{-1})_{jj} X_j^{-1}\right)^2\right]^{1/2} \\ &\le \mathrm{Tr}(\mathbf{C}^{-1})\mathbb{E}^2[X_1^{-1}],\end{aligned}$$

by the triangle inequality for the norm defined as $\mathbb{E}^2(Z) \coloneqq \mathbb{E}[|Z|^2]^{1/2}$ (see [86, Sec. A.3.1]). Finally, using [86, Lem. A.9], we have $\mathbb{E}^2[X_1^{-1}] = 3/(p+1)$, which gives

$$\mathbb{E}[\|\mathbf{\Omega}_1^\dagger\|^4]^{1/2} \le \frac{3}{p+1}\,\mathrm{Tr}(\mathbf{C}^{-1}).$$

The second term can be bounded as

$$\|\mathbf{\Omega}_2\|_{\mathrm{HS}}^2 \le \|\mathbf{\Omega}\|_{\mathrm{HS}}^2 = \sum_{j=1}^{k+p} Z_j, \quad Z_j = \|\omega_j\|_{L^2(D_1)}^2,$$

where the Z_j are i.i.d.. Therefore, by the triangle inequality for the $\mathbb{E}^2$-norm applied to the random variable $\|\mathbf{\Omega}\|_{\mathrm{HS}}^2$, we have

$$\mathbb{E}(\|\mathbf{\Omega}_2\|_{\mathrm{HS}}^4)^{1/2} \le (k+p)\mathbb{E}[Z_1^2]^{1/2}.$$

We then characterize the distribution of Z_1 following the proof of Lemma 2.4 as

$$Z_1 = \sum_{m=1}^\infty \lambda_m Y_m, \quad Y_m \sim \chi^2.$$

Applying triangle inequality to $\mathbb{E}^2(Z_1)$ yields

$$\mathbb{E}[Z_1^2]^{1/2} \le \sum_{m=1}^\infty \lambda_m \mathbb{E}[Y_m^2]^{1/2} = \mathrm{Tr}(K)\mathbb{E}[Y_1^2]^{1/2} = \sqrt{2}\,\mathrm{Tr}(K),$$

which concludes the proof. □

2.2 Recovering the Green's function from input-output pairs

It is known that the Green's function associated with Equation (2.2) always exists, is unique, and is a nonnegative function $G : D \times D \to \mathbb{R}^+ \cup \{\infty\}$ such that

$$u(x) = \int_D G(x,y) f(y)\,\mathrm{d}y, \quad f \in \mathcal{C}_c^\infty(D).$$

For each $y \in \Omega$ and any $r > 0$, we have $G(\cdot, y) \in \mathcal{H}^1(D \setminus B_r(y)) \cap \mathcal{W}_0^{1,1}(D)$ [80]. Here, $B_r(y) = \{z \in \mathbb{R}^3 : \|z - y\|_2 < r\}$, $\mathcal{W}^{1,1}(D)$ is the space of weakly differentiable functions in the L^1-sense, and $\mathcal{W}_0^{1,1}(D)$ is the closure of $\mathcal{C}_c^\infty(D)$ in $\mathcal{W}^{1,1}(D)$. Since the PDE in Equation (2.2) is self-adjoint, we also know that for almost every $x, y \in D$, we have $G(x,y) = G(y,x)$ [80].

We now state Theorem 2.3, which shows that if $N = \mathcal{O}(\epsilon^{-6}\log^4(1/\epsilon))$ and one has N input-output pairs $\{(f_j, u_j)\}_{j=1}^N$ with algorithmically-selected f_j, then the Green's function associated with $\mathcal{L}$ in Equation (2.2) can be recovered to within an accuracy of $\mathcal{O}(\Gamma_\epsilon^{-1/2}\log^3(1/\epsilon)\epsilon)$ with high probability. Here, the quantity $0 < \Gamma_\epsilon \leq 1$ measures the quality of the random input functions $\{f_j\}_{j=1}^N$ (see Section 2.2.4.2).

Theorem 2.3. *Let $0 < \epsilon < 1$, $D \subset \mathbb{R}^3$ be a bounded Lipschitz domain, and $\mathcal{L}$ given in Equation (2.2). If G is the Green's function associated with $\mathcal{L}$, then there is a randomized algorithm that constructs an approximation $\tilde{G}$ of G using $\mathcal{O}(\epsilon^{-6}\log^4(1/\epsilon))$ input-output pairs such that, as $\epsilon \to 0$, we have*

$$\|G - \tilde{G}\|_{L^2(D\times D)} = \mathcal{O}\left(\Gamma_\epsilon^{-1/2}\log^3(1/\epsilon)\epsilon\right)\|G\|_{L^2(D\times D)}, \tag{2.11}$$

with probability $\geq 1 - \mathcal{O}(\epsilon^{\log(1/\epsilon)-6})$. The term Γ_ϵ is defined by Equation (2.23).

For simplicity, we have not reported the dependence of the bound in Equation (2.11) with respect to the spectral condition number, $\kappa_C = \lambda_{\max}/\lambda_{\min}$[1], of the coefficient matrix $A(x)$ in Equation (2.2).

Our algorithm that leads to the proof of Theorem 2.3 relies on the extension of the randomized SVD to HS operators (see Section 2.1) and a hierarchical partition of the domain of G into "well-separated" domains. The scheme described in this section is summarized by Algorithm 1.

[1]Here, $\lambda_{\max}$ is defined as $\sup_{x\in D}\lambda_{\max}(A(x))$ and $\lambda_{\min} = \inf_{x\in D}\lambda_{\min}(A(x)) > 0$.

Algorithm 1 Approximation of the Green's function

Input: Action of the integral operator with kernel G

Output: Approximation $\tilde{G}$ of G

1: Construct a hierarchical partition of the domain $D \times D$
2: Approximate the Green's function on the admissible domains with the randomized SVD
3: Neglect G on the non-admissible domains using a decay bound for the Green's function near the diagonal

2.2.1 Recovering the Green's function on admissible domains

Roughly speaking, as $\|x-y\|_2$ increases G becomes smoother about (x, y), which can be made precise using so-called admissible domains [13, 15, 84]. For $X, Y \subset \mathbb{R}^3$, let $\operatorname{diam} X \coloneqq \sup_{x,y\in X} \|x-y\|_2$ be the diameter of X, and $\operatorname{dist}(X, Y) \coloneqq \inf_{x\in X, y\in Y} \|x-y\|_2$ be the shortest distance between X and Y. Admissible domains are defined as follows.

Definition 2.1. *For a fixed parameter $\rho > 0$, we say that two bounded and non-empty domains $X, Y \subset \mathbb{R}^3$ are admissible if*

$$\operatorname{dist}(X, Y) \geq \rho \max\{\operatorname{diam} X, \operatorname{diam} Y\}.$$

Otherwise, we say that $X \times Y$ is non-admissible.

There exists a weaker definition of admissible domains, which only requires that $\operatorname{dist}(X, Y) \geq \rho \min\{\operatorname{diam} X, \operatorname{diam} Y\}$ [84, p. 59], but we do not consider it.

2.2.1.1 Approximation theory on admissible domains

It turns out that the Green's function associated with Equation (2.2) has exponentially decaying singular values when restricted to admissible domains. Roughly speaking, if $X, Y \subset D$ are such that $X \times Y$ is an admissible domain, then G is well-approximated by a function of the form [16]

$$G_k(x, y) = \sum_{j=1}^{k} g_j(x) h_j(y), \quad (x, y) \in X \times Y, \tag{2.12}$$

for some functions $g_1, \ldots, g_k \in L^2(X)$ and $h_1, \ldots, h_k \in L^2(Y)$. This is summarized in Theorem 2.4, which is a corollary of [16, Thm. 2.8].

Theorem 2.4. *Let G be the Green's function associated with Equation (2.2) and $\rho > 0$. Let $X, Y \subset D$ such that $\operatorname{dist}(X, Y) \geq \rho \max\{\operatorname{diam} X, \operatorname{diam} Y\}$. Then, for any*

$0 < \epsilon < 1$, *there exists* $k \leq k_\epsilon \coloneqq \lceil c(\rho, \operatorname{diam} D, \kappa_C)\rceil \lceil \log(1/\epsilon)\rceil^4 + \lceil \log(1/\epsilon)\rceil$ *and an approximant,* G_k*, of* G *in the form given in Equation* (2.12) *such that*

$$\|G - G_k\|_{L^2(X\times Y)} \leq \epsilon \|G\|_{L^2(X\times \hat{Y})}, \quad \hat{Y} \coloneqq \{y \in D,\, \operatorname{dist}(y, Y) \leq \frac{\rho}{2} \operatorname{diam} Y\},$$

where $\kappa_C = \lambda_{\max}/\lambda_{\min}$ *is the spectral condition number of the coefficient matrix* $A(x)$ *in Equation* (2.2) *and* c *is a constant that only depends on* ρ, $\operatorname{diam} D$, κ_C.

Proof. In [16, Thm. 2.8], it is shown that if $Y = \tilde{Y} \cap D$ and $\tilde{Y}$ is convex, then there exists $k \leq c_{\rho/2}^3 \lceil \log(1/\epsilon)\rceil^4 + \lceil \log(1/\epsilon)\rceil$ and an approximant, G_k, of G such that

$$\|G(x,\cdot) - G_k(x,\cdot)\|_{L^2(Y)} \leq \epsilon \|G(x,\cdot)\|_{L^2(\hat{Y})}, \quad x \in X, \tag{2.13}$$

where $\hat{Y} \coloneqq \{y \in D,\, \operatorname{dist}(y, Y) \leq \frac{\rho}{2} \operatorname{diam} Y\}$ and $c_{\rho/2}$ is a constant that only depends on ρ, $\operatorname{diam} Y$, and κ_C. As remarked by [16], $\tilde{Y}$ can be included in a convex of diameter $\operatorname{diam} D$ that includes D to obtain the constant $c(\rho, \operatorname{diam} D, \kappa_C)$. The statement follows by integrating the error bound in Equation (2.13) over X. □

Since the truncated SVD of G on $X \times Y$ gives the best rank $k_\epsilon \geq k$ approximation to G, Theorem 2.4 also gives bounds on singular values:

$$\left(\sum\nolimits_{j=k_\epsilon+1}^{\infty} \sigma_{j,X\times Y}^2\right)^{1/2} \leq \|G - G_k\|_{L^2(X\times Y)} \leq \epsilon \|G\|_{L^2(X\times \hat{Y})}, \tag{2.14}$$

where $\sigma_{j,X\times Y}$ is the jth singular value of G restricted to $X \times Y$. Since $k_\epsilon = \mathcal{O}(\log^4(1/\epsilon))$, we conclude that the singular values of G restricted to admissible domains $X \times Y$ rapidly decay to zero.

2.2.1.2 Randomized SVD for admissible domains

Since G has exponentially decaying singular values on admissible domains $X \times Y$, we use the randomized SVD for HS operators to learn G on $X \times Y$ with high probability (see Section 2.1).

We start by defining a GP on the domain Y. Let $\mathcal{R}_{Y\times Y} K$ be the restriction[2] of the covariance kernel K to the domain $Y \times Y$, which is a continuous symmetric positive definite kernel so that $\mathcal{GP}(0, \mathcal{R}_{Y\times Y} K)$ defines a GP on Y. We choose a target rank $k \geq 1$, an oversampling parameter $p \geq 2$, and form a quasimatrix $\boldsymbol{\Omega} = \left[f_1 \,|\, \cdots \,|\, f_{k+p}\right]$ such that $f_j \in L^2(Y)$ and $f_j \sim \mathcal{GP}(0, \mathcal{R}_{Y\times Y} K)$ are identically distributed and independent. We then extend by zero each column of $\boldsymbol{\Omega}$ from $L^2(Y)$ to

[2] We denote the restriction operator by $\mathcal{R}_{Y\times Y} : L^2(D \times D) \to L^2(Y \times Y)$.

$L^2(D)$ by $\mathcal{R}_Y^*\mathbf{\Omega} = \left[\mathcal{R}_Y^* f_1 \,|\, \cdots \,|\, \mathcal{R}_Y^* f_{k+p}\right]$, where $\mathcal{R}_Y^* f_j \sim \mathcal{GP}(0, \mathcal{R}_{Y\times Y}^* \mathcal{R}_{Y\times Y} K)$. The zero extension operator $\mathcal{R}_Y^* : L^2(Y) \to L^2(D)$ is the adjoint of $\mathcal{R}_Y : L^2(D) \to L^2(Y)$.

Given the training data, $\mathbf{Y} = \left[u_1 \,|\, \cdots \,|\, u_{k+p}\right]$ such that $\mathcal{L}u_j = \mathcal{R}_Y^* f_j$ and $u_j|_{\partial D} = 0$, we now construct an approximation to G on $X \times Y$ using the randomized SVD (see Section 2.1). Following Theorem 2.1, we have the following approximation error for $t \geq 1$ and $s \geq 2$:

$$\|G - \tilde{G}_{X\times Y}\|_{L^2(X\times Y)}^2 \leq \left(1 + t^2 s^2 \frac{3}{\gamma_{k,X\times Y}} \frac{k(k+p)}{p+1} \sum_{j=1}^{\infty} \frac{\lambda_j}{\lambda_1}\right) \left(\sum\nolimits_{j=k+1}^{\infty} \sigma_{j,X\times Y}^2\right)^{1/2}, \tag{2.15}$$

with probability greater than $1 - t^{-p} - e^{-s^2(k+p)}$. Here, $\lambda_1 \geq \lambda_2 \geq \cdots > 0$ are the eigenvalues of K, $\tilde{G}_{X\times Y} = \mathbf{P}_{\mathcal{R}_X\mathbf{Y}} \mathcal{R}_X \mathscr{F} \mathcal{R}_Y^*$ and $\mathbf{P}_{\mathcal{R}_X\mathbf{Y}} = \mathcal{R}_X\mathbf{Y}((\mathcal{R}_X\mathbf{Y})^* \mathcal{R}_X\mathbf{Y})^\dagger (\mathcal{R}_X\mathbf{Y})^*$ is the orthogonal projection onto the space spanned by the columns of $\mathcal{R}_X\mathbf{Y}$. Moreover, $\gamma_{k,X\times Y}$ is a measure of the quality of the covariance kernel of $\mathcal{GP}(0, \mathcal{R}_{Y\times Y}^* \mathcal{R}_{Y\times Y} K)$ (see Section 2.1.4) and, for $1 \leq i, j \leq k$, defined as $\gamma_{k,X\times Y} = k/(\lambda_1 \operatorname{Tr}(\mathbf{C}_{X\times Y}^{-1}))$, where

$$[\mathbf{C}_{X\times Y}]_{ij} = \int_{D\times D} \mathcal{R}_Y^* v_{i,X\times Y}(x) K(x,y) \mathcal{R}_Y^* v_{j,X\times Y}(y) \,\mathrm{d}x \,\mathrm{d}y,$$

and $v_{1,X\times Y}, \ldots, v_{k,X\times Y} \in L^2(Y)$ are the first k right singular functions of G restricted to $X \times Y$.

Unfortunately, there is a big problem with the formula $\tilde{G}_{X\times Y} = \mathbf{P}_{\mathcal{R}_X\mathbf{Y}} \mathcal{R}_X \mathscr{F} \mathcal{R}_Y^*$. It cannot be formed because we only have access to input-output data, so we have no mechanism for composing $\mathbf{P}_{\mathcal{R}_X\mathbf{Y}}$ on the left of $\mathcal{R}_X \mathscr{F} \mathcal{R}_Y^*$. Instead, we note that since the partial differential operator in Equation (2.2) is self-adjoint, $\mathscr{F}$ is self-adjoint, and G is itself symmetric. That means we can use this to write down a formula for $\tilde{G}_{Y\times X}$ instead. That is,

$$\tilde{G}_{Y\times X} = \tilde{G}_{X\times Y}^* = \mathcal{R}_Y \mathscr{F} \mathcal{R}_X^* \mathbf{P}_{\mathcal{R}_X\mathbf{Y}},$$

where we used the fact that $\mathbf{P}_{\mathcal{R}_X\mathbf{Y}}$ is also self-adjoint. This means we can construct $\tilde{G}_{Y\times X}$ by asking for more input-output data to assess the quasimatrix $\mathscr{F}(\mathcal{R}_X^* \mathcal{R}_X \mathbf{Y})$. Of course, to compute $\tilde{G}_{X\times Y}$, we can swap the roles of X and Y in the above argument.

With a target rank of $k = k_\epsilon = \lceil c(\rho, \operatorname{diam} D, \kappa_C) \rceil \lceil \log(1/\epsilon) \rceil^4 + \lceil \log(1/\epsilon) \rceil$ and an oversampling parameter of $p = k_\epsilon$, we can combine Theorem 2.4 and Equations (2.14) and (2.15) to obtain the bound

$$\|G - \tilde{G}_{X\times Y}\|_{L^2(X\times Y)}^2 \leq \left(1 + t^2 s^2 \frac{6k_\epsilon}{\gamma_{k_\epsilon,X\times Y}} \sum_{j=1}^{\infty} \frac{\lambda_j}{\lambda_1}\right) \epsilon^2 \|G\|_{L^2(X\times \hat{Y})}^2,$$

with probability greater than $1-t^{-k_\epsilon}-e^{-2s^2k_\epsilon}$. A similar approximation error holds for $\tilde{G}_{Y\times X}$ without additional evaluations of $\mathscr{F}$. We conclude that our algorithm requires $N_{\epsilon,X\times Y} = 2(k_\epsilon + p) = \mathcal{O}\big(\log^4(1/\epsilon)\big)$ input-output pairs to learn an approximant to G on $X \times Y$ and $Y \times X$.

2.2.2 Ignoring the Green's function on non-admissible domains

When the Green's function is restricted to non-admissible domains, its singular values may not decay. Instead, to learn G we take advantage of the off-diagonal decay property of G. It is known that for almost every $x \neq y \in D$ then

$$G(x,y) \leq \frac{c_{\kappa_C}}{\|x-y\|_2}\|G\|_{L^2(D\times D)}, \tag{2.16}$$

where c_{κ_C} is an implicit constant that only depends on κ_C (see [80, Thm. 1.1]). Note that we have normalized [80, Eq. 1.8] to highlight the dependence on $\|G\|_{L^2(D\times D)}$.

If $X \times Y$ is a non-admissible domain, then for any $(x,y) \in X \times Y$, we find that

$$\|x-y\|_2 \leq \operatorname{dist}(X,Y) + \operatorname{diam}(X) + \operatorname{diam}(Y) < (2+\rho)\max\{\operatorname{diam} X, \operatorname{diam} Y\},$$

because $\operatorname{dist}(X,Y) < \rho \max\{\operatorname{diam} X, \operatorname{diam} Y\}$. This means that $x \in B_r(y) \cap D$, where $r = (2+\rho)\max\{\operatorname{diam} X, \operatorname{diam} Y\}$. Using Equation (2.16), we have

$$\begin{aligned}\int_X G(x,y)^2 dx &\leq \int_{B_r(y)\cap D} G(x,y)^2 \,\mathrm{d}x \leq c_{\kappa_C}^2 \|G\|_{L^2(D\times D)}^2 \int_{B_r(y)} \|x-y\|_2^{-2}\,\mathrm{d}x \\ &\leq 4\pi c_{\kappa_C}^2 r \|G\|_{L^2(D\times D)}^2.\end{aligned}$$

Noting that $\operatorname{diam}(Y) \leq r/(2+\rho)$ and $\int_Y 1\,\mathrm{d}y \leq 4\pi(\operatorname{diam}(Y)/2)^3/3$, we have the following inequality for non-admissible domains $X \times Y$:

$$\|G\|_{L^2(X\times Y)}^2 \leq \frac{2\pi^2}{3(2+\rho)^3} c_{\kappa_C}^2 r^4 \|G\|_{L^2(D\times D)}^2, \tag{2.17}$$

where $r = (2+\rho)\max\{\operatorname{diam} X, \operatorname{diam} Y\}$. We conclude that the Green's function restricted to a non-admissible domain has a relatively small norm when the domain itself is small. Therefore, in our approximant $\tilde{G}$ for G, we ignore G on non-admissible domains by setting $\tilde{G}$ to be zero.

2.2.3 Hierarchical admissible partition of domain

We now describe a hierarchical partitioning of $D \times D$ so that many subdomains are admissible domains, and the non-admissible domains are all small. For ease of notation, we may assume—without loss of generality—that $\operatorname{diam} D = 1$ and $D \subset [0,1]^3$; otherwise, one should shift and scale D. Moreover, partitioning $[0,1]^3$ and restricting the partition to D is easier than partitioning D directly. For the definition of admissible domains, we find it convenient to select $\rho = 1/\sqrt{3}$.

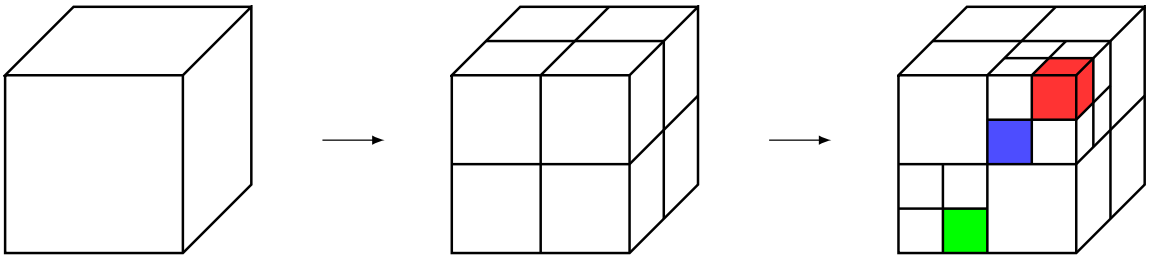


Figure 2.1: Two levels of hierarchical partitioning of $[0,1]^3$. The blue and green domains are admissible, while the blue and red domains are non-admissible.

Let $I = [0,1]^3$. The hierarchical partitioning for n levels is defined recursively as:

- $I_{1\times1\times1} \coloneqq I_1 \times I_1 \times I_1 = [0,1]^3$ is the root for level $L = 0$.
- At a given level $0 \le L \le n-1$, if $I_{j_1\times j_2\times j_3} \coloneqq I_{j_1} \times I_{j_2} \times I_{j_3}$ is a node of the tree, then it has 8 children defined as

$$\{I_{2j_1+n_j(1)} \times I_{2j_2+n_j(2)} \times I_{2j_3+n_j(3)} \mid n_j \in \{0,1\}^3\}.$$

  Here, if $I_j = [a,b]$, $0 \le a < b \le 1$, then $I_{2j} = \left[a, \frac{a+b}{2}\right]$ and $I_{2j+1} = \left[\frac{a+b}{2}, b\right]$.

The set of non-admissible domains can be given by an unwieldy expression

$$P_{\text{non-adm}} = \bigcup_{\substack{\bigwedge_{i=1}^3 |j_i - \tilde{j}_i| \le 1 \\ 2^n \le j_1, j_2, j_3 \le 2^{n+1}-1 \\ 2^n \le \tilde{j}_1, \tilde{j}_2, \tilde{j}_3 \le 2^{n+1}-1}} I_{j_1\times j_2\times j_3} \times I_{\tilde{j}_1\times\tilde{j}_2\times\tilde{j}_3}, \tag{2.18}$$

where $\wedge$ is the logical "and" operator. The set of admissible domains is given by

$$P_{\text{adm}} = \bigcup_{L=1}^{n} \Lambda(P_{\text{non-adm}}(L-1)) \backslash P_{\text{non-adm}}(L)), \tag{2.19}$$

where $P_{\text{non-adm}}(L)$ is the set of non-admissible domain for a hierarchical level of L and

$$\Lambda(P_{\text{non-adm}}(L-1)) = \bigcup_{\substack{I_{j_1\times j_2\times j_3}\times I_{\tilde{j}_1\times\tilde{j}_2\times\tilde{j}_3} \\ \in P_{\text{non-adm}}(L-1)}} \bigcup_{n_j, n_{\tilde{j}}\in\{0,1\}^3} I_{\times_{i=1}^3 2j_i+n_j(i)} \times I_{\times_{i=1}^3 2\tilde{j}_i+n_{\tilde{j}}(i)}.$$

Using Equation (2.18)-Equation (2.19), the number of admissible and non-admissible domains are precisely $|P_{\text{non-adm}}| = (3\times 2^n - 2)^3$ and $|P_{\text{adm}}| = \sum_{\ell=1}^n 2^6(3\times 2^{L-1}-2)^3 - (3\times 2^L - 2)^3$. In particular, the size of the partition at the hierarchical level $0 \le L \le n$ is equal to 8^L and the tree has a total of $(8^{n+1}-1)/7$ nodes (see Figure 2.2).

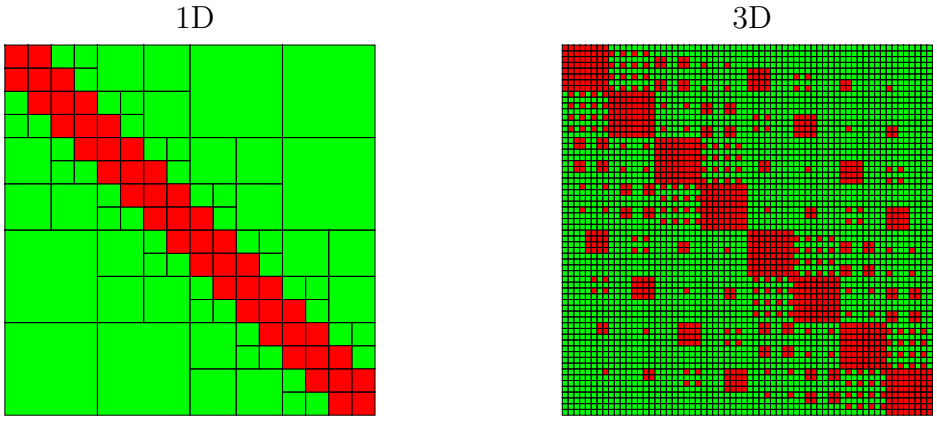


Figure 2.2: For illustration purposes, we include the hierarchical structure of the Green's functions in 1D after 4 levels (left) and in 3D after 2 levels (right). The hierarchical structure in 3D is complicated as this is physically a 6-dimensional tensor that has been rearranged so it can be visualized.

Finally, the hierarchical partition of $D \times D$ can be defined via the partition $P = P_{\text{adm}} \cup P_{\text{non-adm}}$ of $[0,1]^3$ by doing the following:

$$D \times D = \bigcup_{\tau\times\sigma\in P} (\tau \cap D) \times (\sigma \cap D).$$

The sets of admissible and non-admissible domains of $D \times D$ are denoted by P_{adm} and $P_{\text{non-adm}}$ in the next sections.

2.2.4 Recovering the Green's function on the entire domain

We now show that we can recover G on the entire domain $D \times D$.

2.2.4.1 Global approximation on the non-admissible set

Let n_ϵ be the number of levels in the hierarchical partition $D \times D$ (see Section 2.2.3). We want to make sure that the norm of the Green's function on all non-admissible domains is small so that we can safely ignore that part of G (see Section 2.2.2). As one increases the hierarchical partitioning levels, the volume of the non-admissible domains get smaller (see Figure 2.3).

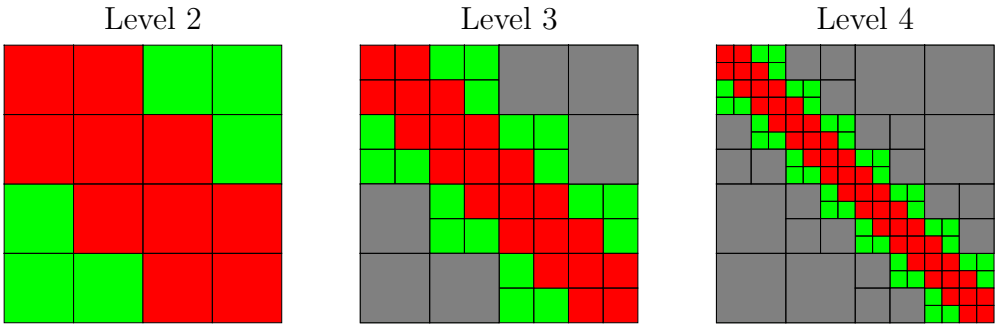


Figure 2.3: For illustration purposes, we include the hierarchical structure of the Green function in 1D. The green blocks are admissible domains at that level, the gray blocks are admissible at a higher level, and the red blocks are the non-admissible domains at that level. The area of the non-admissible domains decreases at deeper levels.

Let $X \times Y \in P_{\text{non-adm}}$ be a non-admissible domain, the two domains X and Y have diameter bounded by $\sqrt{3}/2^{n_\epsilon}$ because they are included in cubes of side length $1/2^{n_\epsilon}$ (see Section 2.2.3). Combining this with Equation (2.17) yields

$$\|G\|^2_{L^2(X\times Y)} \leq 2\pi^2(6+\sqrt{3})c^2_{\kappa_C} 2^{-4n_\epsilon} \|G\|^2_{L^2(D\times D)}.$$

Therefore, the L^2-norm of G on the non-admissible domain $P_{\text{non-adm}}$ satisfies

$$\|G\|^2_{L^2(P_{\text{non-adm}})} = \sum_{X\times Y \in P_{\text{non-adm}}} \|G\|^2_{L^2(X\times Y)} \leq 54\pi^2(6+\sqrt{3})c^2_{\kappa_C} 2^{-n_\epsilon} \|G\|^2_{L^2(D\times D)},$$

where we used $|P_{\text{non-adm}}| = (3 \times 2^{n_\epsilon} - 2)^3 \leq 27(2^{3n_\epsilon})$. This means that if we select n_ϵ to be

$$n_\epsilon = \left\lceil \log_2(54\pi^2(6+\sqrt{3})c^2_{\kappa_C}) + 2\log_2(1/\epsilon) \right\rceil \sim 2\log_2(1/\epsilon), \tag{2.20}$$

then we guarantee that $\|G\|_{L^2(P_{\text{non-adm}})} \leq \epsilon \|G\|_{L^2(D\times D)}$. We can safely ignore G on non-admissible domains—by taking the zero approximant—while approximating G to within ϵ.

2.2.4.2 Learning rate of the Green's function

Following Section 2.2.1.2, we can construct an approximant $\tilde{G}_{X\times Y}$ to the Green's function on an admissible domain $X \times Y$ of the hierarchical partitioning using the HS randomized SVD algorithm, which requires $N_{\epsilon,X\times Y} = \mathcal{O}(\log^4(1/\epsilon))$ input-output training pairs (see Section 2.2.1.2). Therefore, the number of training input-output pairs needed to construct an approximant to G on all admissible domains is given by

$$N_\epsilon = \sum_{X\times Y\in P_{\text{adm}}} N_{\epsilon,X\times Y} = \mathcal{O}\left(|P_{\text{adm}}|\log^4(1/\epsilon)\right),$$

where $|P_{\text{adm}}|$ denotes the total number of admissible domains at the hierarchical level n_ϵ, which is given by Equation (2.20). Then, we have (see Section 2.2.3):

$$|P_{\text{adm}}| = \sum_{\ell=1}^{n_\epsilon} 2^6(3\times 2^{\ell-1}-2)^3 - (3\times 2^\ell - 2)^3 \leq 6^3 2^{3n_\epsilon}, \tag{2.21}$$

and, using Equation (2.20), we obtain $|P_{\text{adm}}| = \mathcal{O}(1/\epsilon^6)$. This means that the total number of required input-output training pairs to learn G with high probability is bounded by $N_\epsilon = \mathcal{O}\left(\epsilon^{-6}\log^4(1/\epsilon)\right)$.

2.2.4.3 Global approximation error

We know that with $N_\epsilon = \mathcal{O}(\epsilon^{-6}\log^4(1/\epsilon))$ input-output training pairs, we can construct an accurate approximant to G on each admissible and non-admissible domain. Since the number of admissible and non-admissible domains depends on ϵ, we now check that this implies a globally accurate approximant that we denote by $\tilde{G}$.

Since $\tilde{G}$ is zero on non-admissible domains and $P_{\text{adm}} \cap P_{\text{non-adm}}$ has measure zero, we have

$$\|G-\tilde{G}\|^2_{L^2(D\times D)} \leq \epsilon^2\|G\|^2_{L^2(D\times D)} + \sum_{X\times Y\in P_{\text{adm}}} \|G-\tilde{G}\|^2_{L^2(X\times Y)}. \tag{2.22}$$

Following Section 2.2.4.2, if $X\times Y$ is admissible then the approximation error satisfies

$$\|G-\tilde{G}_{X\times Y}\|^2_{L^2(X\times Y)} \leq 12t^2s^2\frac{k_\epsilon}{\gamma_{k_\epsilon,X\times Y}}\sum_{j=1}^{\infty}\frac{\lambda_j}{\lambda_1}\epsilon^2\|G\|^2_{L^2(X\times\hat{Y})},$$

with probability greater than $1-t^{-k_\epsilon}-e^{-2s^2k_\epsilon}$. Here, $\hat{Y} = \{y\in D, \operatorname{dist}(y,Y)\leq \operatorname{diam} Y/2\sqrt{3}\}$ (see Theorem 2.4 with $\rho = 1/\sqrt{3}$). To measure the worst $\gamma_{k_\epsilon,X\times Y}$, we define

$$\Gamma_\epsilon = \min\{\gamma_{k_\epsilon,X\times Y} : X\times Y\in P_{\text{adm}}\}. \tag{2.23}$$

From Equation (2.9), we know that $0 < \Gamma_\epsilon \leq 1$ and that $1/\Gamma_\epsilon$ is greater than the harmonic mean of the first k_ϵ scaled eigenvalues of the covariance kernel K, *i.e.*,

$$\frac{1}{\Gamma_\epsilon} \geq \frac{1}{k_\epsilon} \sum_{j=1}^{k_\epsilon} \frac{\lambda_1}{\lambda_j}, \tag{2.24}$$

Now, one can see that $X \times \hat{Y}$ is included in at most $5^3 = 125$ neighbours including itself. Assuming that all the probability bounds hold on the admissible domains, this implies that

$$\begin{aligned}
\sum_{X\times Y \in P_{\text{adm}}} \|G - \tilde{G}\|^2_{L^2(X\times Y)} &\leq \sum_{X\times Y \in P_{\text{adm}}} \|G - \tilde{G}\|^2_{L^2(X\times Y)} \\
&\leq 12t^2 s^2 \frac{k_\epsilon}{\lambda_1 \Gamma_\epsilon} \operatorname{Tr}(K)\epsilon^2 \sum_{X\times Y \in P_{\text{adm}}} \|G\|^2_{L^2(X\times \hat{Y})} \\
&\leq 1500 t^2 s^2 \frac{k_\epsilon}{\lambda_1 \Gamma_\epsilon} \operatorname{Tr}(K)\epsilon^2 \|G\|^2_{L^2(D\times D)}.
\end{aligned}$$

We then choose $t = e$ and $s = k_\epsilon^{1/4}$ so that the approximation bound on each admissible domain holds with probability of failure less than $2e^{-\sqrt{k_\epsilon}}$. Finally, using Equation (2.22) we conclude that as $\epsilon \to 0$, the approximation error on $D \times D$ satisfies

$$\|G - \tilde{G}\|_{L^2(D\times D)} = \mathcal{O}\left(\Gamma_\epsilon^{-1/2} \log^3(1/\epsilon)\epsilon\right) \|G\|_{L^2(D\times D)},$$

with probability $\geq (1 - 2e^{-\sqrt{k_\epsilon}})^{6^3 2^{3n_\epsilon}} = 1 - \mathcal{O}(\epsilon^{\log(1/\epsilon)-6})$, where n_ϵ is given by Equation (2.20). We conclude that the approximant $\tilde{G}$ is a good approximation to G with very high probability.

2.3 Discussion

There are several possible extensions of the results presented in this chapter related to the recovery of hierarchical matrices, the study of other partial differential operators, and practical deep learning applications, which we discuss further in this section.

2.3.1 Fast and stable reconstruction of hierarchical matrices

We described an algorithm for reconstructing Green's function on admissible domains of a hierarchical partition of $D \times D$ that requires performing the HS randomized SVD $\mathcal{O}(\epsilon^{-6})$ times. We want to reduce it to a factor that is $\mathcal{O}(\text{polylog}(1/\epsilon))$. A polylogarithmic function in x is any polynomial in $\log(x)$ and is denoted by $\text{polylog}(x)$.

For $n \times n$ hierarchical matrices, there are several existing algorithms for recovering the matrix based on matrix-vector products [25, 130, 143, 144]. There are two main approaches: (1) the "bottom-up" approach: one begins at the lowest level of the hierarchy and moves up and (2) the "top-down" approach: one updates the approximant by peeling off the off-diagonal blocks and going down the hierarchy. The bottom-up approach requires $\mathcal{O}(n)$ applications of the randomized SVD algorithm [143]. There are lower complexity alternatives that only require $\mathcal{O}(\log(n))$ matrix-vector products with random vectors [130]. However, the algorithm in [130] is not yet proven to be theoretically stable as errors from low-rank approximations potentially accumulate exponentially, though this is not observed in practice. For symmetric positive semi-definite matrices, it may be possible to employ a sparse Cholesky factorization [198, 199]. This leads us to formulate the following challenge:

Algorithmic challenge: Design a provably stable algorithm that can recover an $n \times n$ hierarchical matrix using $\mathcal{O}(\log(n))$ matrix-vector products with high probability.

If one can design such an algorithm and it can be extended to HS operators, then the $\mathcal{O}(\epsilon^{-6}\log^4(1/\epsilon))$ term in Theorem 2.3 may improve to $\mathcal{O}(\text{polylog}(1/\epsilon))$. This means that the learning rate of partial differential operators of the form of Equation (2.2) will be a polynomial in $\log(1/\epsilon)$ and grow sublinearly with respect to $1/\epsilon$.

2.3.2 Extension to other partial differential operators

Our learning rate for elliptic partial differential operators (PDOs) in three variables (see Section 2.2) depends on the decay of the singular values of the Green's function on admissible domains [16]. We expect that one can also find the learning rate for other PDOs.

It is known that the Green's functions associated to elliptic PDOs in two dimensions exist and satisfy the following pointwise estimate [53]:

$$|G(x,y)| \leq C\left(\frac{1}{\gamma R^2} + \log\left(\frac{R}{\|x-y\|_2}\right)\right), \quad \|x-y\|_2 \leq R \coloneqq \frac{1}{2}\max(d_x, d_y), \quad (2.25)$$

where $d_x = \text{dist}(x, \partial D)$, γ is a constant depending on the size of the domain D, and C is an implicit constant. One can conclude that $G(x,\cdot)$ is locally integrable for all $x \in D$ with $\|G(x,\cdot)\|_{L^p(B_r(x)\cap D)} < \infty$ for $r > 0$ and $1 \leq p < \infty$. We believe that the pointwise estimate in Equation (2.25) implies the off-diagonal low-rank structure of G here, as suggested in [16]. Therefore, we expect that the results in this chapter can

be extended to elliptic PDOs in two variables. It should also be possible to characterize the learning rate for elliptic PDOs with lower order terms (under reasonable conditions) [54, 94, 106] as the associated Green's functions have similar regularity and pointwise estimates. The main task is to extend [16, Thm. 2.8] to construct separable approximations of the Green's functions on admissible domains.

PDOs in four or more variables are far more challenging since we rely on the following bound on the Green's function on non-admissible domains [80]:

$$G(x,y) \leq \frac{c(d,\kappa_C)}{\lambda_{\min}} \|x-y\|_2^{2-d}, \quad x \neq y \in D,$$

where $D \subset \mathbb{R}^d$, $d \geq 3$ is the dimension, and c is a constant depending only on d and κ_C. This inequality implies that the L^p-norm of G on non-admissible domains is finite when $0 \leq p < d/(d-2)$. However, for a dimension $d \geq 4$, we have $p < 2$ and one cannot ensure that the L^2 norm of G is finite. Therefore, the Green's function may not be compatible with the HS randomized SVD.

The low-rank theory of Bebendorf and Hackbush has been recently extended from elliptic to parabolic operators [16] and combined with pointwise estimates for Green's functions [107] to obtain a learning rate parabolic PDEs, expressed in the L^1-norm [28]. In contrast, we believe that deriving a theoretical learning rate for hyperbolic PDOs remains a significant research challenge for many reasons. The first roadblock is that the Green's function associated with hyperbolic PDOs do not necessarily lie in $L^2(D \times D)$. For example, the Green's function associated with the wave equation in three variables, *i.e.*, $\mathcal{L} = \partial_t^2 - \nabla^2$, is not square-integrable as

$$G(x,t,y,s) = \frac{\delta(t-s-\|x-y\|_2)}{4\pi\|x-y\|_2}, \quad (x,t),(y,s) \in \mathbb{R}^3 \times [0,\infty),$$

where $\delta(\cdot)$ is the Dirac delta function.

Finally, while the extension to nonlinear dynamical systems seem out of reach of the technique presented in this chapter, characterizing the sample complexity of such systems and learn finite-dimensional approximations of the dynamics using Koopman operator theory [6, 34, 110] would be an interesting future research direction.

2.3.3 Connection with neural networks

As a concluding remark, we emphasize that the algorithm described in this chapter to learn Green's functions is not meant to be applied in practice. The proof of Theorem 2.3 relies on the construction of a hierarchical partition of the domain $D \times D$ and the HS randomized SVD algorithm applied on each admissible domain. While this

gives an algorithm for approximating Green's functions with high probability, it would be prohibitively computationally expensive to employ the hierarchical scheme and the generalization of the randomized SVD to large-scale three-dimensional problems.

However, there are more practical approaches based on deep learning that currently do not yet have theoretical guarantees [65, 71]. As we will see in Chapter 5, deep learning techniques may be more competitive due to their ability to learn non self-adjoint problems and the fast optimization algorithms, based on stochastic gradient descent, for training neural networks. There are many possible connections between the work presented in this chapter and neural networks from practical and theoretical viewpoints.

A promising opportunity that we will explore in Chapter 5 is to design a NN that can learn and approximate Green's functions using input-output training pairs $\{(f_j, u_j)\}_{j=1}^N$. Once a neural network $\mathcal{N}$ has been trained such that $\|\mathcal{N} - G\|_{L^2} \leq \epsilon \|G\|_{L^2}$, the solution to $\mathcal{L}u = f$ can be obtained by computing the integral $u(x) = \int_D \mathcal{N}(x, y) f(y) \, \mathrm{d}y$. Therefore, this may give an efficient computational approach for discovering operators since a NN is only trained once. Incorporating a priori knowledge of the Green's function into the network architecture design could be particularly beneficial. As an example, one might exploit the low-rank structure by performing dimensionality reduction with an autoencoder [71, 75]. One could also wrap the selection of the kernel in the GP for generating random functions and training data into a Bayesian framework. We expect that the theory developed in this chapter could guide deep learning experiments regarding the choice of training data and the type neural network architectures used to take advantage of the hierarchical structure of Green's functions and their singularity along the diagonal.

Finally, we wonder how many parameters in a NN are needed to approximate a Green's function associated with elliptic PDOs within a tolerance of $0 < \epsilon < 1$. Can one exploit the off-diagonal low-rank structure of Green's functions to reduce the number of parameters? We expect the recent work on the characterization of ReLU NNs' approximation power is useful [82, 175, 242]. The use of NNs with high approximation power such as rational NNs might also be of interest to approximate the singularities of the Green's function near the diagonal, as we shall see in Chapters 4 and 5.

Chapter 3

A generalization of the randomized singular value decomposition*

The theory behind the randomized SVD has been extended in Chapter 2 to non-standard covariance matrices and HS operators. However, the probability bounds, generalizing [86, Thm. 10.7], are not sharp enough to emphasize the improved performance of covariance matrices with prior information over the standard randomized SVD. In this chapter, we improve the bounds obtained in Chapter 2 when the matrix-vector products are with multivariate Gaussian random vectors. Our theory allows for multivariate Gaussian random input vectors that have a general symmetric positive semi-definite covariance matrix. A key novelty of this work is that prior knowledge of the matrix $\mathbf{A}$ can be exploited to design covariance matrices that achieve lower approximation errors than the randomized SVD with standard Gaussian vectors. We then design a practical algorithm for learning Hilbert–Schmidt (HS) operators using random input functions, sampled from a Gaussian process (GP). Examples of applications include learning integral kernels such as Green's functions associated with linear partial differential equations, as discussed in the previous chapter.

The choice of the covariance kernel in the GP is crucial and impacts both the theoretical bounds and numerical results of the randomized SVD. This leads us to introduce a new covariance kernel based on weighted Jacobi polynomials for learning HS operators. One of the main advantages of this kernel is that it is directly expressed as a Karhunen–Loève expansion [101, 133] so that it is faster to sample functions from the associated GP than using a standard squared-exponential kernel. In addition, we

*This chapter is based on a paper with Alex Townsend [31], published in ICLR 2022. Townsend had an advisory role; I proved the theoretical results, performed the numerical experiments, and was the lead author in writing the paper.

show that the smoothness of the functions sampled from a GP with the Jacobi kernel can be controlled as it is related to the decay rate of the kernel's eigenvalues.

3.1 Theoretical bounds for non-standard covariance matrices

In this section we provide new probability bounds for GPs with nonstandard covariance matrices. Let $m \geq n \geq 1$ and $\mathbf{A}$ be an $m \times n$ real matrix with singular value decomposition $\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^*$, where $\mathbf{U}$ and $\mathbf{V}$ are orthonormal matrices, and $\mathbf{\Sigma}$ be an $m \times n$ diagonal matrix with entries $\sigma_1(\mathbf{A}) \geq \cdots \geq \sigma_n(\mathbf{A}) \geq 0$. For a fixed target rank $k \geq 1$, we define $\mathbf{\Sigma}_1$ and $\mathbf{\Sigma}_2$ to be the $k \times k$ and $(n-k) \times (n-k)$ diagonal matrices, which respectively contain the first k singular values of $\mathbf{A}$: $\sigma_1(\mathbf{A}) \geq \cdots \geq \sigma_k(\mathbf{A})$, and the remaining singular values. Let $\mathbf{V}_1$ be the $n \times k$ matrix obtained by truncating $\mathbf{V}_1$ after k columns and $\mathbf{V}_2$ the remainder. In this section, $\mathbf{K}$ denotes a symmetric positive semi-definite $n \times n$ matrix with kth largest eigenvalue $\lambda_k > 0$ and $\mathbf{\Omega} \in \mathbb{R}^{n \times \ell}$ a Gaussian random matrix with $\ell \geq k$ independent columns sampled from a multivariate normal distribution with covariance matrix $\mathbf{K}$. Finally, we define $\mathbf{\Omega}_1 \coloneqq \mathbf{V}_1^*\mathbf{\Omega}$ and $\mathbf{\Omega}_2 \coloneqq \mathbf{V}_2^*\mathbf{\Omega}$. The following theorem is a refinement of Theorem 2.1. While it is formulated with matrices, the same result holds for HS operators in infinite dimensions.

Theorem 3.1. *Let $\mathbf{A}$ be an $m \times n$ matrix, $k \geq 1$ an integer, and choose an oversampling parameter $p \geq 4$. If $\mathbf{\Omega} \in \mathbb{R}^{n \times (k+p)}$ is a Gaussian random matrix, where each column is i.i.d. from a multivariate Gaussian distribution with covariance matrix $\mathbf{K} \in \mathbb{R}^{n \times n}$, and $\mathbf{QR} = \mathbf{A\Omega}$ is the economized QR decomposition of $\mathbf{A\Omega}$, then for all $u, t \geq 1$,*

$$\|\mathbf{A} - \mathbf{QQ}^*\mathbf{A}\|_{\mathrm{F}} \leq \left(1 + ut\sqrt{(k+p)\frac{3k}{p+1}\frac{\beta_k}{\gamma_k}}\right)\sqrt{\sum_{j=k+1}^{n}\sigma_j^2(\mathbf{A})}, \tag{3.1}$$

with failure probability at most $t^{-p} + [ue^{-(u^2-1)/2}]^{k+p}$. Here, the covariance quality factors are denoted by $\gamma_k = k/(\lambda_1 \operatorname{Tr}((\mathbf{V}_1^*\mathbf{K}\mathbf{V}_1)^{-1})))$ and $\beta_k = \operatorname{Tr}(\mathbf{\Sigma}_2^2\mathbf{V}_2^*\mathbf{K}\mathbf{V}_2)/(\lambda_1\|\mathbf{\Sigma}_2\|_{\mathrm{F}}^2)$, where λ_1 is the largest eigenvalue of $\mathbf{K}$.

This result differs from Theorem 2.1 and [86, Thm. 10.5] due to the additional factors γ_k and β_k, which measure the quality of the covariance matrix to learn $\mathbf{A}$ in Theorem 3.1. They can be respectively bounded (Lemmas 2.2 and 3.2) using the

eigenvalues $\lambda_1 \geq \cdots \geq \lambda_n$ of the covariance matrix $\mathbf{K}$ and the singular values of $\mathbf{A}$ as:

$$\frac{1}{\gamma_k} \leq \frac{1}{k} \sum_{j=n-k+1}^{n} \frac{\lambda_1}{\lambda_j}, \qquad \beta_k \leq \sum_{j=k+1}^{n} \frac{\lambda_{j-k}}{\lambda_1} \sigma_j^2(\mathbf{A}) \Big/ \sum_{j=k+1}^{n} \sigma_j^2(\mathbf{A}). \tag{3.2}$$

This shows that the performance of the generalized randomized SVD depends on the decay rate of the sequence $\{\lambda_j\}$. The quantities γ_k and β_k depend on how much prior information of the $k+1, \ldots, n$ right singular vectors of $\mathbf{A}$ is encoded in $\mathbf{K}$. In the ideal situation where these singular vectors are known, then one can define $\mathbf{K}$ such that $\beta_k = 0$ for $\lambda_{k+1} = \cdots = \lambda_n = 0$. Unlike the weaker but more explicit bound proven in Section 2.1, this highlights that a suitably chosen covariance matrix can outperform the randomized SVD with standard Gaussian vectors (see Section 3.4.1 for a numerical example).

The proof of Theorem 3.1 will require bounding $\|\mathbf{\Omega}_1^\dagger\|_{\mathrm{F}}^2$, which we achieve using Lemma 2.3, as well as the term $\|\mathbf{\Sigma}_2\mathbf{\Omega}_2\|_{\mathrm{F}}^2$, which is done in the following lemma.

Lemma 3.1. *With the notations introduced at the beginning of the section, for all $s \geq 0$, we have*

$$\mathbb{P}\left\{\|\mathbf{\Sigma}_2\mathbf{\Omega}_2\|_{\mathrm{F}}^2 > \ell(1+s)\,\mathrm{Tr}(\mathbf{\Sigma}_2^2\mathbf{V}_2^*\mathbf{K}\mathbf{V}_2)\right\} \leq (1+s)^{\ell/2} e^{-s\ell/2}.$$

Proof. Let ω_j be the jth column of $\mathbf{\Omega}$ for $1 \leq j \leq \ell$ and $v_1, \ldots, v_n$ be the n columns of the orthonormal matrix $\mathbf{V}$. We first remark that

$$\|\mathbf{\Omega}_2\|_{\mathrm{F}}^2 = \sum_{j=1}^{\ell} Z_j, \quad Z_j \coloneqq \sum_{n_1=1}^{n-k} \sigma_{k+n_1}^2(\mathbf{A})(v_{k+n_1}^*\omega_j)^2,$$

where the Z_j are i.i.d. because $\omega_j \sim \mathcal{N}(0, \mathbf{K})$ are i.i.d. Let $\lambda_1 \geq \cdots \geq \lambda_n \geq 0$ be the eigenvalues of $\mathbf{K}$ with eigenvectors $\psi_1, \ldots, \psi_n \in \mathbb{R}^n$. For $1 \leq j \leq \ell$, we have,

$$\omega_j = \sum_{i=1}^{n} (c_i^{(j)})^2 \sqrt{\lambda_i}\psi_i,$$

where $c_i^{(j)} \sim \mathcal{N}(0,1)$ are i.i.d. for $1 \leq i \leq n$ and $1 \leq j \leq \ell$. Then,

$$Z_j = \sum_{i=1}^{n} (c_i^{(j)})^2 \lambda_i \sum_{n_1=1}^{n-k} \sigma_{k+n_1}^2(\mathbf{A})(v_{k+n_1}^*\psi_i)^2 = \sum_{i=1}^{n} X_i$$

where the X_i are independent. Let $\gamma_i = \lambda_i \sum_{n_1=1}^{n-k} \sigma_{k+n_1}^2(\mathbf{A})(v_{k+n_1}^*\psi_i)^2$, then $X_i \sim \gamma_i\chi^2$ for $1 \leq i \leq n$.

Let $0 < \theta < 1/(2\sum_{i=1}^n \gamma_i)$. We can bound the moment generating function of $\sum_{i=1}^n X_i$ as

$$\mathbb{E}\left[e^{\theta \sum_{i=1}^n X_i}\right] = \prod_{i=1}^n \mathbb{E}\left[e^{\theta X_i}\right] = \prod_{i=1}^n (1-2\theta\gamma_i)^{-1/2} \leq \left(1 - 2\theta \sum_{i=1}^n \gamma_i\right)^{-1/2}$$

because the X_i/γ_i are independent and follow a chi-squared distribution. The right inequality is obtained by showing by recurrence that, if $a_1, \ldots, a_n \geq 0$ are such that $\sum_{i=1}^n a_i \leq 1$, then $\prod_{i=1}^n (1-a_i) \geq 1 - \sum_{i=1}^n a_i$. For convenience, we define $C_1 \coloneqq \sum_{i=1}^n \gamma_n$, we have shown that

$$\mathbb{E}\left[e^{\theta Z_j}\right] \leq (1-2\theta C_1)^{-1/2}.$$

Moreover, we find that

$$\begin{aligned} C_1 &= \sum_{n_1=1}^{n-k} \sigma_{k+n_1}^2 v_{k+n_1}^* \left(\sum_{i=1}^n \psi_i^* \lambda_i \psi_i\right) v_{k+n_1} = \sum_{n_1=1}^{n-k} \sigma_{k+n_1}^2(\mathbf{A}) v_{k+n_1}^* \mathbf{K} v_{k+n_1} \\ &= \operatorname{Tr}(\mathbf{\Sigma}_2^2 \mathbf{V}_2^* \mathbf{K} \mathbf{V}_2). \end{aligned}$$

Let $s \geq 0$ and $0 < \theta < 1/(2C_1)$. By the Chernoff bound [41, Thm. 1], we obtain

$$\begin{aligned} \mathbb{P}\left\{\|\mathbf{\Sigma}_2 \mathbf{\Omega}_2\|_{\mathrm{F}}^2 > \ell(1+s)C_1\right\} &\leq e^{-(1+s)C_1 \ell\theta} \mathbb{E}\left[e^{\theta Z_j}\right]^\ell \\ &= e^{-(1+s)C_1\ell\theta}(1-2\theta C_1)^{-\ell/2}. \end{aligned}$$

We minimize the bound over $0 < \theta < 1/(2\operatorname{Tr}(K))$ by choosing $\theta = s/(2(1+s)C_1)$, which gives

$$\mathbb{P}\left\{\|\mathbf{\Sigma}_2 \mathbf{\Omega}_2\|_{\mathrm{F}}^2 > \ell(1+s)C_1\right\} \leq (1+s)^{\ell/2} e^{-\ell s/2}.$$

□

We now prove Theorem 3.1, which provides a refined probability bound for the performance of the generalized randomized SVD on matrices.

Proof of Theorem 3.1. Using Theorem 2.2 and the submultiplicativity of the Frobenius norm, we have

$$\|\mathbf{A} - \mathbf{Q}\mathbf{Q}^*\mathbf{A}\|_{\mathrm{F}}^2 \leq \|\mathbf{\Sigma}_2\|_{\mathrm{F}}^2 + \|\mathbf{\Sigma}_2 \mathbf{\Omega}_2\|_{\mathrm{F}}^2 \|\mathbf{\Omega}_1^\dagger\|_{\mathrm{F}}^2. \tag{3.3}$$

Let $\ell = k + p$ with $p \geq 4$. Combining Lemmas 2.3 and 3.1 to bound the terms $\|\mathbf{\Sigma}_2 \mathbf{\Omega}_2\|_{\mathrm{F}}^2$ and $\|\mathbf{\Omega}_1^\dagger\|_{\mathrm{F}}^2$ in Equation (3.3) yields the following probability estimate:

$$\begin{aligned} \|\mathbf{A} - \mathbf{Q}\mathbf{Q}^*\mathbf{A}\|_{\mathrm{F}}^2 &\leq \|\mathbf{\Sigma}_2\|_{\mathrm{F}}^2 + 3t^2(1+s)\frac{k+p}{p+1} \operatorname{Tr}((\mathbf{V}_1^* \mathbf{K} \mathbf{V}_1)^{-1}) \operatorname{Tr}(\mathbf{\Sigma}_2^2 \mathbf{V}_2^* \mathbf{K} \mathbf{V}_2) \\ &\leq \left(1 + 3t^2(1+s)\frac{(k+p)k}{p+1}\frac{\beta_k}{\gamma_k}\right) \sum_{j=k+1}^n \sigma_j^2(\mathbf{A}), \end{aligned}$$

with failure probability at most $t^{-p} + (1+s)^{(k+p)/2} e^{-s(k+p)/2}$. Note that we introduced $\gamma_k \coloneqq k/(\lambda_1 \operatorname{Tr}((\mathbf{V}_1^* \mathbf{K} \mathbf{V}_1)^{-1})))$ and $\beta_k \coloneqq \operatorname{Tr}(\mathbf{\Sigma}_2^2 \mathbf{V}_2^* \mathbf{K} \mathbf{V}_2)/(\lambda_1 \|\mathbf{\Sigma}_2\|_{\mathrm{F}}^2)$. We conclude the proof by defining $u = \sqrt{1+s} \geq 1$. □

The following Lemma provides an estimate of the quantity β_k introduced in the statement of Theorem 3.1.

Lemma 3.2. *Let $\beta_k = \operatorname{Tr}(\mathbf{\Sigma}_2^2 \mathbf{V}_2^* \mathbf{K} \mathbf{V}_2)/(\lambda_1 \|\mathbf{\Sigma}_2\|_{\mathrm{F}}^2)$, then the following inequality holds*

$$\beta_k \leq \sum_{j=k+1}^{n} \frac{\lambda_{j-k}}{\lambda_1} \sigma_j^2(\mathbf{A}) \Big/ \sum_{j=k+1}^{n} \sigma_j^2(\mathbf{A}).$$

Proof. Let $\mu_1 \geq \cdots \geq \mu_{n-k}$ be the eigenvalues of the matrix $\mathbf{V}_2^* \mathbf{K} \mathbf{V}_2$. Using von Neumann's trace inequality [152, 230], we have

$$\operatorname{Tr}(\mathbf{\Sigma}_2^2 \mathbf{V}_2^* \mathbf{K} \mathbf{V}_2) \leq \sum_{j=k+1}^{n} \mu_{j-k} \sigma_j^2(\mathbf{A}).$$

Then, the matrix $\mathbf{V}_2^* \mathbf{K} \mathbf{V}_2$ is a principal submatrix of $\mathbf{V}^* \mathbf{K} \mathbf{V}$, which has the same eigenvalues of K. Therefore, by [104, Thm. 6.46], the eigenvalues of $\mathbf{V}_2^* \mathbf{K} \mathbf{V}_2$ are individually bounded by the eigenvalues of $\mathbf{K}$, *i.e.*, $\mu_j \leq \lambda_j$ for $1 \leq j \leq n-k$, which concludes the proof. □

Finally, we highlight that the statement of Theorem 3.1 can be simplified by choosing $p = 5$, $t = 4$, and $u = 3$ to highlight the difference with the standard bounds for the randomized SVD.

Corollary 3.1 (Generalized randomized SVD)**.** *Let $\mathbf{A}$ be an $m \times n$ matrix and $k \geq 1$ an integer. If $\mathbf{\Omega} \in \mathbb{R}^{n \times (k+5)}$ is a Gaussian random matrix, where each column is i.i.d. from a multivariate Gaussian distribution with symmetric positive semi-definite covariance matrix $\mathbf{K} \in \mathbb{R}^{n \times n}$, and $\mathbf{QR} = \mathbf{A\Omega}$ is the economized QR decomposition of $\mathbf{A\Omega}$, then*

$$\mathbb{P}\left[\|\mathbf{A} - \mathbf{Q}\mathbf{Q}^* \mathbf{A}\|_{\mathrm{F}} \leq \left(1 + 9\sqrt{k(k+5)\frac{\beta_k}{\gamma_k}} \right) \sqrt{\sum_{j=k+1}^{n} \sigma_j^2(\mathbf{A})} \right] \geq 0.999.$$

In contrast, a simplification of the theorem for the randomized SVD [86, Thm. 10.7] by choosing $t = 6$ and $u = 4$ gives the following result.

Corollary 3.2 (Randomized SVD)**.** *Let* $\mathbf{A}$ *be an* $m \times n$ *matrix and* $k \geq 1$ *an integer. If* $\mathbf{\Omega} \in \mathbb{R}^{n \times (k+5)}$ *is a standard Gaussian random matrix and* $\mathbf{QR} = \mathbf{A\Omega}$ *is the economized QR decomposition of* $\mathbf{A\Omega}$*, then*

$$\mathbb{P}\left[\|\mathbf{A} - \mathbf{QQ}^*\mathbf{A}\|_{\mathrm{F}} \leq \left(1 + 16\sqrt{k+5}\right)\sqrt{\sum_{j=k+1}^{n} \sigma_j^2(\mathbf{A})}\right] \geq 0.999.$$

The following proposition bounds the expected approximation error of the randomized SVD with multivariate Gaussian inputs.

Proposition 3.1. *Let* $\mathbf{A}$ *be an* $m \times n$ *matrix,* $k \geq 1$ *an integer, and choose an oversampling parameter* $p \geq 2$*. If* $\mathbf{\Omega} \in \mathbb{R}^{n \times (k+p)}$ *is a Gaussian random matrix, where each column is sampled from a multivariate Gaussian distribution with covariance matrix* $\mathbf{K} \in \mathbb{R}^{n \times n}$*, and* $\mathbf{QR} = \mathbf{A\Omega}$ *is the economized QR decomposition of* $\mathbf{A\Omega}$*, then,*

$$\mathbb{E}\left[\|\mathbf{A} - \mathbf{QQ}^*\mathbf{A}\|_{\mathrm{F}}\right] \leq \left(1 + \sqrt{\frac{\beta_k}{\gamma_k}\frac{k(k+p)}{p-1}}\right)\sqrt{\sum_{j=k+1}^{n} \sigma_j^2(\mathbf{A})},$$

where $\gamma_k = k/(\lambda_1 \operatorname{Tr}((\mathbf{V}_1^*\mathbf{K}\mathbf{V}_1)^{-1})))$ *and* $\beta_k = \operatorname{Tr}(\mathbf{\Sigma}_2^2\mathbf{V}_2^*\mathbf{K}\mathbf{V}_2)/(\lambda_1\|\mathbf{\Sigma}_2\|_{\mathrm{F}}^2)$*.*

We remark that for standard Gaussian inputs, we have $\gamma_k = \beta_k = 1$ in Proposition 3.1, and we recover the average Frobenius error of the randomized SVD [86, Thm. 10.5] up to a factor of $(k+p)$ due to the non-independence of $\mathbf{\Omega}_1$ and $\mathbf{\Omega}_2$ in general. The proof of Proposition 3.1 consists of combining the proof of Theorem 2.1 with the following lemma, which is a refinement of Lemma 2.5.

Lemma 3.3. *Let* $\ell \geq 1$*,* $\mathbf{\Omega} \in \mathbb{R}^{n \times \ell}$ *be a Gaussian random matrix, where each column is sampled from a multivariate Gaussian distribution with covariance matrix* $\mathbf{K}$*, and* $\mathbf{T}$ *be an* $\ell \times k$ *matrix. Then,*

$$\mathbb{E}[\|\mathbf{\Sigma}_2\mathbf{V}_2^*\mathbf{\Omega}\mathbf{T}\|_{\mathrm{F}}^2] = \operatorname{Tr}(\mathbf{\Sigma}_2^2\mathbf{V}_2^*\mathbf{K}\mathbf{V}_2)\|\mathbf{T}\|_{\mathrm{F}}^2. \tag{3.4}$$

Proof. Let $\mathbf{K} = \mathbf{Q}_\mathbf{K}\mathbf{\Lambda}\mathbf{Q}_\mathbf{K}^*$ be the eigenvalue decomposition of $\mathbf{K}$, where $\mathbf{Q}_\mathbf{K}$ is orthonormal and $\mathbf{\Lambda}$ is a diagonal matrix containing the eigenvalues of $\mathbf{K}$ in decreasing order: $\lambda_1 \geq \cdots \geq \lambda_n \geq 0$. We note that $\mathbf{\Omega}$ can be expressed as $\mathbf{\Omega} = \mathbf{Q}_\mathbf{K}\mathbf{\Lambda}^{1/2}\mathbf{G}$, where $\mathbf{G}$ is a standard Gaussian matrix. Let $\mathbf{S} = \mathbf{\Sigma}_2\mathbf{V}_2^*\mathbf{Q}_\mathbf{K}\mathbf{\Lambda}^{1/2}$, the proof follows from [86, Prop. A.1], which shows that $\mathbb{E}\|\mathbf{SGT}\|_{\mathrm{F}}^2 = \|\mathbf{S}\|_{\mathrm{F}}^2\|\mathbf{T}\|_{\mathrm{F}}^2$. □

Note that one can bound the term $\operatorname{Tr}(\mathbf{\Sigma}_2^2\mathbf{V}_2^*\mathbf{K}\mathbf{V}_2)$ by $\lambda_1\|\mathbf{\Sigma}_2\|_{\mathrm{F}}^2$, where λ_1 is the largest eigenvalue of $\mathbf{K}$ (cf. Lemma 2.5). While this provides a simple upper bound, it does not demonstrate that the use of a covariance matrix containing prior information on the singular vectors of $\mathbf{A}$ can outperform the randomized SVD with standard Gaussian inputs.

3.2 Randomized SVD for Hilbert–Schmidt operators

We now describe the randomized SVD for learning HS operators (see Algorithm 2). The algorithm is implemented in the Chebfun software system [56], which is a MATLAB package for computing with functions. The Chebfun implementation of the randomized SVD for HS operators uses Chebfun's capabilities, which offer continuous analogues of several matrix operations like the QR decomposition and numerical integration. Indeed, the continuous analogue of a matrix-vector multiplication $\mathbf{A\Omega}$ for an HS integral operator $\mathscr{F}$ (see Section 1.6 for definitions and properties of HS operators), with kernel $G : D \times D \to \mathbb{R}$, is

$$(\mathscr{F} f)(x) = \int_D G(x, y) f(y) \, \mathrm{d}y, \quad x \in D, \, f \in L^2(D),$$

where $D \subset \mathbb{R}^d$ with $d \geq 1$.

Algorithm 2 Randomized SVD for HS operators

Input: HS integral operator $\mathscr{F}$ with kernel $G(x, y)$, number of samples $k > 0$
Output: Approximation G_k of G

1: Define a GP covariance kernel K
2: Sample the GP k times to generate a quasimatrix of random functions $\Omega = [f_1 \dots f_k]$
3: Evaluate the integral operator at Ω, $Y = [\mathscr{F}(f_1) \dots \mathscr{F}(f_k)]$
4: Orthonormalize the columns of Y, $Q = \mathrm{orth}(Y) = [q_1 \dots q_k]$
5: Compute an approximation to G by evaluating the adjoint of $\mathscr{F}$
6: Initialize $G_k(x, y)$ to 0
7: **for** $i = 1 : k$ **do**
8: $\quad G_k(x, y) \leftarrow G_k(x, y) + q_i(x) \int_D G(z, y) q_i(z) \, \mathrm{d}z$

The algorithm takes as input an integral operator that we aim to approximate. Note that we focus here on learning an integral operator, but other HS operators would work similarly. The first step of the randomized SVD for HS operators consists of generating a $D \times k$ quasimatrix Ω by sampling a GP k times, where k is the target rank (see Section 3.3). Therefore, each column of Ω is an object, consisting of a polynomial approximation of a smooth random function sampled from the GP in the Chebyshev basis. After evaluating the HS operator at Ω to obtain a quasimatrix Y, we use the QR algorithm [218] to obtain an orthonormal basis Q for the range of the columns of Y. Then, the randomized SVD for HS operators requires the

left-evaluation of the operator $\mathscr{F}$ or, equivalently, the evaluation of its adjoint $\mathscr{F}_t$ satisfying:

$$(\mathscr{F}_t f)(x) = \int_D G(y,x) f(y)\,\mathrm{d}y, \quad x \in D.$$

We evaluate the adjoint of $\mathscr{F}$ at each column vector of Q to construct an approximation G_k of G. Finally, the approximation error between the operator kernel G and the learned kernel G_k can be computed in the L^2-norm, corresponding to the HS norm of the integral operator.

3.3 Covariance kernels

To generate the random input functions $f_1, \ldots, f_k$ for the randomized SVD for HS operators, we draw them from a GP, denoted by $\mathcal{GP}(0, K)$, for a certain covariance kernel K. A widely employed covariance kernel is the squared-exponential function K_{SE} [187] given by

$$K_{\mathrm{SE}}(x,y) = \exp\left(-|x-y|^2/(2\ell^2)\right), \quad x, y \in D, \tag{3.5}$$

where $\ell > 0$ is a parameter controlling the length-scale of the GP. This kernel is isotropic as it only depends on $|x-y|$, is infinitely differentiable, and its eigenvalues decay supergeometrically to 0. Since the bound in Theorem 3.1 degrades as the ratio λ_1/λ_j increases for $j \geq k+1$ (cf. Equation (3.2)), the randomized SVD for learning HS operators prefers covariance kernels with slowly decaying eigenvalues. Our randomized SVD cannot hope to learn HS operators where the range of the operator has a rank greater than $\tilde{k}$, where $\tilde{k}$ is such that the $\tilde{k}$th eigenvalue of K_{SE} reaches machine precision. In Figure 3.1, we display the squared-exponential kernel with length-scale parameters $\ell = 1, 0.1, 0.01$ together with sampled functions from $\mathcal{GP}(0, K_{\mathrm{SE}})$. We observe that the functions become more oscillatory as the length-scale parameter ℓ decreases and hence the numerical rank of the kernel increases or, equivalently, the associated eigenvalues $\{\lambda_j\}$ decay more slowly to zero.

Other popular kernels for GPs include the Matérn kernel [131, 187] and Brownian bridge [161]. Prior information on the HS operator can also be enforced through the choice of the covariance kernel. For instance, one can impose the periodicity of the samples by using the following squared-exponential periodic kernel:

$$K_{\mathrm{Per}}(x,y) = \exp\left(-\frac{2}{\ell^2}\sin^2\left(\frac{x-y}{2}\right)\right), \quad x, y \in D,$$

where $\ell > 0$ is the length-scale parameter.

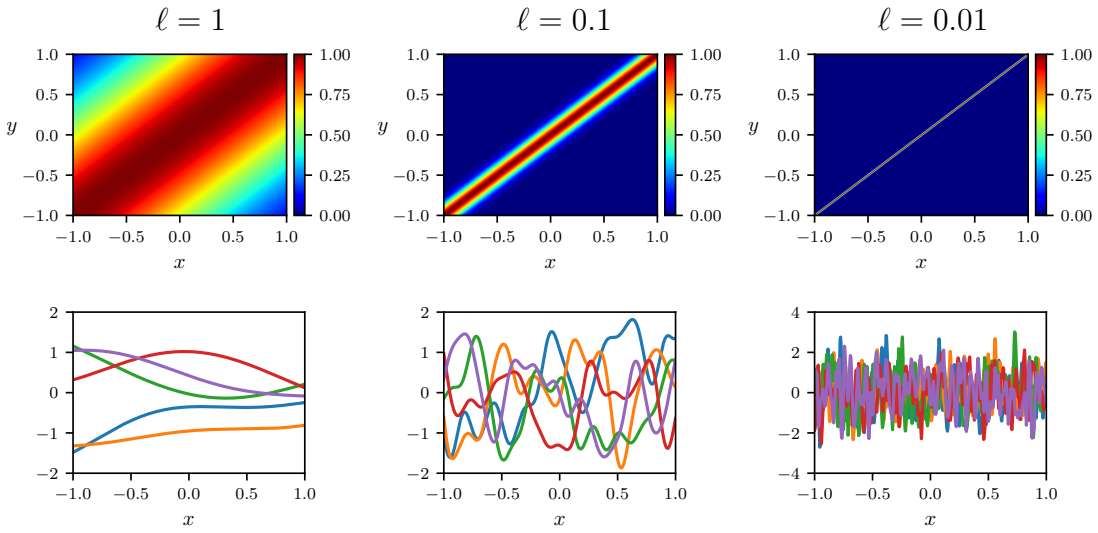


Figure 3.1: Squared-exponential covariance kernel K_{SE} with parameter $\ell = 1, 0.1, 0.01$ (top row) and five functions sampled from $\mathcal{GP}(0, K_{\text{SE}})$ (bottom row).

3.3.1 Sample random functions from a Gaussian process

In finite dimensions, a random vector $u \sim \mathcal{N}(0, \mathbf{K})$, where $\mathbf{K} \in \mathbb{R}^{n\times n}$ is a covariance matrix with Cholesky factorization $\mathbf{K} = \mathbf{L}\mathbf{L}^*$, can be generated from the matrix-vector product $u = \mathbf{L}c$. Here, $c \in \mathbb{R}^n$ is a vector whose entries follow the standard Gaussian distribution. We now detail how this process extends to infinite dimensions with a continuous covariance kernel. Let K be a continuous symmetric positive-definite covariance function defined on the domain $[a, b] \times [a, b] \subset \mathbb{R}^2$ with $-\infty < a < b < \infty$. We consider the continuous analogue of the Cholesky factorization to write K as [218]

$$K(x, y) = \sum_{j=1}^{\infty} r_j(x) r_j(y) = L_c(x) L_c^*(y), \quad x, y \in [a, b],$$

where r_j is the jth row of L_c, which —in Chebfun's terminology— is a lower-triangular quasimatrix. In practice, we truncate the series after n terms, either arbitrarily or when the nth largest kernel eigenvalue, λ_n, falls below machine precision. Then, if $c \in \mathbb{R}^n$ follows the standard Gaussian distribution, a function u can be sampled from $\mathcal{GP}(0, K)$ as $u = L_c c$. That is,

$$u(x) = \sum_{j=1}^{n} c_j r_j(x), \quad x \in [a, b].$$

The continuous Cholesky factorization is implemented in Chebfun2 [217], which is the extension of Chebfun for computing with two-dimensional functions. As an example,

the polynomial approximation, which is accurate up to essentially machine precision, of the squared-exponential covariance kernel K_{SE} with parameter $\ell = 0.01$ on $[-1, 1]^2$ yields a numerical rank of $n = 503$. The functions sampled from $\mathcal{GP}(0, K_{\mathrm{SE}})$ become more oscillatory as the length-scale parameter ℓ decreases and hence the numerical rank of the kernel increases or, equivalently, the associated eigenvalues sequence $\{\lambda_j\}$ decays more slowly to zero.

3.3.2 Influence of the kernel's eigenvalues and Mercer's representation

The covariance kernel can also be defined from its Mercer's representation as

$$K(x, y) = \sum_{j=1}^{\infty} \lambda_j \psi_j(x) \psi_j(y), \quad x, y \in D, \tag{3.6}$$

where $\{\psi_j\}$ is an orthonormal basis of $L^2(D)$ and $\lambda_1 \geq \lambda_2 \geq \cdots > 0$ [93, Thm. 4.6.5]. We prefer to construct K directly from Mercer's representations for several reasons. First, one can impose prior knowledge of the kernel of the HS operator on the eigenfunctions of K (such as periodicity or smoothness). Then, one can often generate samples from $\mathcal{GP}(0, K)$ efficiently using Equation (3.6). Finally, one can control the decay rate of the eigenvalues of K.

Hence, the quantity γ_k in the probability bound of Theorem 3.1 measures the quality of the covariance kernel K in $\mathcal{GP}(0, K)$ to generate random functions that can learn the HS operator $\mathcal{F}$. To minimize $1/\gamma_k$ we would like to select the eigenvalues $\lambda_1 \geq \lambda_2 \geq \cdots > 0$ of K so that they have the slowest possible decay rate while maintaining $\sum_{j=1}^{\infty} \lambda_j < \infty$. One needs $\{\lambda_j\} \in \ell^1$ to guarantee that $\omega \sim \mathcal{GP}(0, K)$ has finite expected squared L^2-norm, *i.e.*, $\mathbb{E}[\|\omega\|^2_{L^2(D)}] = \sum_{j=1}^{\infty} \lambda_j < \infty$. The best sequence of eigenvalues we know that satisfies this property is called the Rissanen sequence [189] and is given by $\lambda_j = R_j \coloneqq 2^{-L(j)}$, where

$$L(j) = \log_2(c_0) + \log_2^*(j), \quad \log_2^*(j) = \sum_{i=2}^{\infty} \max(\log_2^{(i)}(j), 0), \quad c_0 = \sum_{i=2}^{\infty} 2^{-\log_2^*(i)},$$

and $\log_2^{(i)}(j) = \log_2 \circ \cdots \circ \log_2(j)$ is the composition of $\log_2(\cdot)$ i times. Other options for the choice of eigenvalues include any sequence of the form $\lambda_j = j^{-\nu}$ for $\nu > 1$.

3.3.3 Jacobi covariance kernel

If $D = [-1, 1]$, then a natural choice of orthonormal basis of $L^2(D)$ to define the Mercer's representation of the kernel are weighted Jacobi polynomials [50, 166]. That

is, for a weight function $w_{\alpha,\beta}(x) = (1-x)^\alpha(1+x)^\beta$ with $\alpha, \beta > -1$, and any positive eigenvalue sequence $\{\lambda_j\}$, we consider the Jacobi kernel

$$K_{\text{Jac}}^{(\alpha,\beta)}(x,y) = \sum_{j=0}^{\infty} \lambda_{j+1} w_{\alpha,\beta}^{1/2}(x) \tilde{P}_j^{(\alpha,\beta)}(x) w_{\alpha,\beta}^{1/2}(y) \tilde{P}_j^{(\alpha,\beta)}(y), \quad x, y \in [-1,1], \tag{3.7}$$

where $\tilde{P}_j^{(\alpha,\beta)}$ is the scaled Jacobi polynomial of degree j and parameters (α, β) where $P_j^{(\alpha,\beta)}$ is defined by Rodrigues' formula [211, Eq. 4.3.1] as

$$w_{\alpha,\beta}(x) P_j^{(\alpha,\beta)}(x) = \frac{(-1)^j}{2^j n!} \frac{d^j}{dx^j} \left\{ w_{\alpha,\beta}(x)(1-x^2)^j \right\}.$$

The polynomials $\tilde{P}_j^{(\alpha,\beta)}$ are normalized such that $\|w_{\alpha,\beta}^{1/2} \tilde{P}_j^{(\alpha,\beta)}\|_{L^2([-1,1])} = 1$ and $\{\lambda_j\}$ is chosen such that $K_{\text{Jac}}^{(\alpha,\beta)} \in L^2([-1,1]^2)$. In this case, a random function can be sampled as

$$u(x) = \sum_{j=0}^{\infty} \sqrt{\lambda_{j+1}} c_j w_{\alpha,\beta}^{1/2} \tilde{P}_j^{(\alpha,\beta)}(x), \quad x \in [-1,1],$$

where $c_j \sim \mathcal{N}(0,1)$ for $0 \leq j \leq \infty$.

A desirable property of a covariance kernel is to be unbiased towards one spatial direction, *i.e.*, $K(x,y) = K(-y,-x)$ for $x, y \in [-1,1]$, which motivates us to always select $\alpha = \beta$. Moreover, it is desirable to have the eigenfunctions of $K_{\text{Jac}}^{(\alpha,\beta)}$ to be polynomial so that one can generate samples from $\mathcal{GP}(0,K)$ efficiently. This leads us to choose α and β to be even integers. The choice of $\alpha = \beta = 0$ gives the Legendre kernel [68, 83]. In the rest of this chapter, we will use Equation (3.7) with $\alpha = \beta = 2$ to ensure that functions sampled from the associated GP satisfy homogeneous Dirichlet boundary conditions (see Figure 3.3). We emphasize that covariance kernels on higher dimensional domains of the form $D = [-1,1]^d$, for $d \geq 2$, can be defined using tensor products of weighted Jacobi polynomials.

3.3.4 Smoothness of functions sampled from a GP with Jacobi kernel

We now connect the decay rate of the eigenvalues of the Jacobi covariance kernel $K_{\text{Jac}}^{(2,2)}$ to the smoothness of the samples from $\mathcal{GP}(0, K_{\text{Jac}}^{(2,2)})$. Hence, the Jacobi covariance function allows the control of the decay rate of the eigenvalues $\{\lambda_j\}$ as well as the smoothness of the resulting randomly generated functions. First, Lemma 3.4 asserts that if the coefficients of an infinite polynomial series have sufficient decay, then the resulting series is smooth with regularity depending on the decay rate. This result can be seen as a converse to [220, Thm. 7.1].

Lemma 3.4. *Let $\{p_j\}$ be a family of polynomials such that $\max_{x\in[-1,1]} |p_j(x)| = 1$ and $\deg(p_j) \leq j$. If $f_n(x) = \sum_{j=0}^{n} a_j p_j(x)$ with $|a_j| \leq j^{-\nu}$ for $\nu > 1$, then f_n converges uniformly to $f(x) = \sum_{j=0}^{\infty} a_j p_j(x)$ and f is μ times continuously differentiable for any integer μ such that $\mu < (\nu - 1)/2$.*

Proof. By Markov brothers' inequality [142], for all $j \geq 0$ and $0 \leq \mu \leq j$, we have $\max_{x\in[-1,1]} |p_j^{(\mu)}(x)| \leq j^{2\mu}$. Therefore, $|f_n^{(\mu)}(x)| \leq \sum_{j=0}^{n} |a_j| \|p_j^{(\mu)}\|_\infty \leq \sum_{j=0}^{n} j^{2\mu-\nu}$ so $|f_n^{(\mu)}(x)| < \infty$ if $\mu < (\nu-1)/2$. The result follows from a standard result on uniform convergence and differentiation [193, Thm. 7.17]. □

Note that the main application of this lemma occurs when $\deg(p_j) = j$ for all $j \geq 0$. We then prove a bound on ultraspherical polynomials in order to apply Lemma 3.4 to functions sampled from the GP with the Jacobi covariance kernel $K_{\text{Jac}}^{(2,2)}$. First, note that $\tilde{P}_j^{(2,2)}$ is a scaled ultraspherical polynomial $\tilde{C}_j^{(5/2)}$ with parameter $5/2$ and degree $j \geq 0$ so it can be bounded by the following proposition.

Proposition 3.2. *Let $\tilde{C}_j^{(5/2)}$ be the ultraspherical polynomial of degree j with parameter $5/2$, normalized such that $\int_{-1}^{1} (1-x^2)^2 \tilde{C}_j^{(5/2)}(x)^2 \,\mathrm{d}x = 1$. Then,*

$$\max_{x\in[-1,1]} |(1-x^2)\tilde{C}_j^{(5/2)}(x)| \leq 2\sqrt{j + 5/12}, \quad j \geq 0. \tag{3.8}$$

Proof. Let $j \geq 0$ and $x \in [-1, 1]$, according to [166, Table 18.3.1],

$$\tilde{C}_j^{(5/2)}(x) = 3\sqrt{\frac{j+5/2}{(j+1)(j+2)(j+3)(j+4)}} C_j^{(5/2)}(x), \tag{3.9}$$

where $C_j^{(5/2)}(x)$ is the standard ultraspherical polynomial. Using [166, (18.9.8)], we have

$$(1-x^2)C_j^{(5/2)}(x) = \frac{(j+3)(j+4)C_j^{(3/2)}(x) - (j+1)(j+2)C_{j+2}^{(3/2)}(x)}{6(j+5/2)}.$$

By using [166, (18.9.7)], we have $(C_{j+2}^{(3/2)}(x) - C_j^{(3/2)}(x))/2 = (j+5/2)C_{j+2}^{(1/2)}(x)$ and hence,

$$(1-x^2)C_j^{(5/2)}(x) = \frac{2}{3}C_j^{(3/2)}(x) - \frac{(j+1)(j+2)}{3}C_{j+2}^{(1/2)}(x).$$

We bound the two terms with [166, (18.14.4)] to obtain the following inequalities:

$$|C_j^{(3/2)}(x)| \leq \frac{(j+1)(j+2)}{2}, \quad |C_{j+2}^{(1/2)}(x)| \leq 1.$$

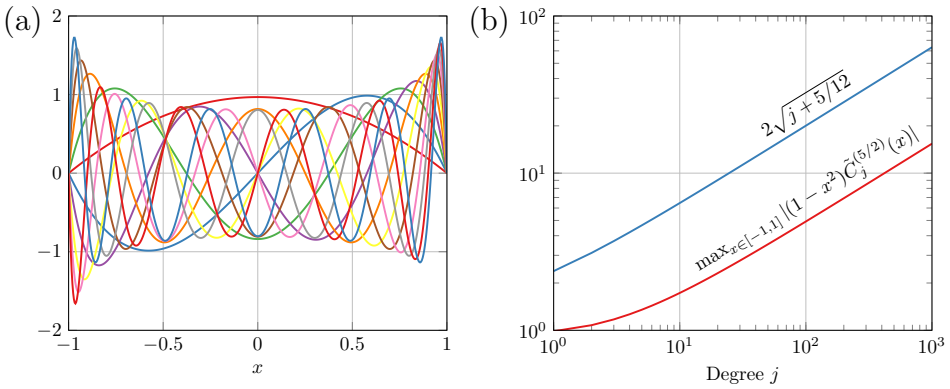


Figure 3.2: (a) Normalized ultraspherical polynomials $\tilde{C}_j^{(5/2)}$ up to degree 10. (b) Theoretical bound (blue line) for the maximum of weighted ultraspherical polynomials on $[-1,1]$, obtained in Proposition 3.2, against the one observed numerically (red line).

Hence, $|(1-x^2)C_j^{(5/2)}(x)| \leq 2(j+1)(j+2)/3$ and following Equation (3.9) we obtain

$$|(1-x^2)\tilde{C}_j^{(5/2)}(x)| \leq 2\sqrt{\frac{(j+1)(j+2)(j+5/2)}{(j+3)(j+4)}} \leq 2\sqrt{j+5/12},$$

which concludes the proof. □

The bound given in Proposition 3.2 differs initially by a factor of 4/3 from the numerically observed upper bound $(1.5\sqrt{j+5/12})$ as shown by Figure 3.2. We now state the following theorem about the regularity of functions sampled from $\mathcal{GP}(0, K_{\text{Jac}}^{(2,2)})$, which guarantees that if the eigenvalues are chosen such that $\lambda_j = \mathcal{O}(1/j^\nu)$ with $\nu > 3$, then $f \sim \mathcal{GP}(0, K_{\text{Jac}}^{(2,2)})$ is almost surely continuous. Moreover, a faster decay of the eigenvalues of $K_{\text{Jac}}^{(2,2)}$ implies higher regularity of the sampled functions, in an almost sure sense.

Theorem 3.2. *Let $\{\lambda_j\} \in \ell^1(\mathbb{R}^+)$ be a positive sequence such that $\lambda_j = \mathcal{O}(j^{-\nu})$ for $\nu > 3$. If f is sampled from $\mathcal{GP}(0, K_{Jac}^{(2,2)})$, then $f \in \mathcal{C}^\mu([-1,1])$ almost surely for any integer $\mu < (\nu-3)/2$.*

Proof. Since $f \sim \mathcal{GP}(0, K_{\text{Jac}}^{(2,2)})$, $f \sim \sum_{j=0}^{\infty} c_j \sqrt{\lambda_{j+1}}(1-x^2)\tilde{P}_j^{(2,2)}(x)$, where $c_j \sim \mathcal{N}(0,1)$ for $j \geq 0$. Let f_n denote the truncation of f after n terms. By letting $M > 0$ be the constant such that $\lambda_{j+1} \leq M(j+1)^{-\nu}$, we find that

$$\|f - f_n\|_\infty \leq S_n, \quad S_n := 2\sqrt{M} \sum_{j=n+2}^{\infty} |c_{j-1}| j^{(1-\nu)/2},$$

where we used $\max_{x\in[-1,1]} |(1-x^2)\tilde{P}_j^{(2,2)}(x)| \leq 2\sqrt{j+1}$ (cf. Proposition 3.2). Thus, we have

$$\mathbb{P}\left(\lim_{n\to\infty} \|f - f_n\|_\infty = 0\right) \geq \mathbb{P}\left(\lim_{n\to\infty} S_n = 0\right).$$

Here, $S_n \sim X_n = \sum_{j=n+2}^{\infty} Y_j j^{(1-\nu)/2}$, where Y_j follows a half-normal distribution [123] with parameter $\sigma = 1$ and the $(Y_j)_j$ are independent. We want to show that $X_n \xrightarrow{a.s.} 0$. For $\epsilon > 0$, using Chebyshev's inequality, we have:

$$\sum_{n=0}^{\infty} \mathbb{P}(|X_n| \geq \epsilon) \leq \frac{1}{\epsilon^2} \sum_{n=0}^{\infty} \left(1 - \frac{2}{\pi}\right) \sum_{j=n+2}^{\infty} \frac{1}{j^{\nu-1}} \leq \frac{1}{\epsilon^2} \left(1 - \frac{2}{\pi}\right) \frac{1}{\nu - 2} \sum_{n=1}^{\infty} \frac{1}{n^{\nu-2}},$$

which is finite if $\nu > 3$. Therefore, using the Borel–Cantelli Lemma [57, Chapt. 2.3], X_n converges to 0 almost surely and $\mathbb{P}(\lim_{n\to\infty} X_n = 0) = 1$. Finally,

$$\mathbb{P}\left(\lim_{n\to\infty} \|f - f_n\|_\infty = 0\right) \geq \mathbb{P}\left(\lim_{n\to\infty} X_n = 0\right) = 1,$$

which proves that $\{f_n\}$ converges uniformly and hence f is continuous with probability one. The statement for higher order derivatives follows the proof of Lemma 3.4. □

This theorem can be seen as a particular case of Driscoll's zero-one law [55], which characterizes the regularity of functions samples from GPs (see also [99]). Hence, one must have $\sum_{j=1}^{\infty} j\lambda_j < \infty$ so that the series of functions in Equation (3.7) converges uniformly and $K_{\text{Jac}}^{(2,2)}$ is a continuous kernel. Under this additional constraint, the best choice of eigenvalues is given by a scaled Rissanen sequence: $\lambda_j = R_j/j$, for $j \geq 1$ (cf. Section 3.3.2). In Figure 3.3, we display the Jacobi kernel of type $(2, 2)$ with functions sampled from the corresponding GP. We selected eigenvalue sequences of different decay rates: from the faster $1/j^4$ to the slower Rissanen sequence R_j/j (Section 3.3.2). For $\lambda_j = 1/j^3$ and $\lambda_j = R_j/j$, we observe a large variation of the randomly generated functions near $x = \pm 1$, indicating a potential discontinuity of the samples at these two points as $n \to \infty$. This is in agreement with Theorem 3.2, which only guarantees continuity (with probability one) of the randomly generated functions if $\lambda_j \sim 1/j^\nu$ with $\nu > 3$.

3.4 Numerical experiments

We now perform several numerical experiments with the randomized SVD to learn matrices using random vectors sampled for a multivariate Gaussian distribution and HS operators.

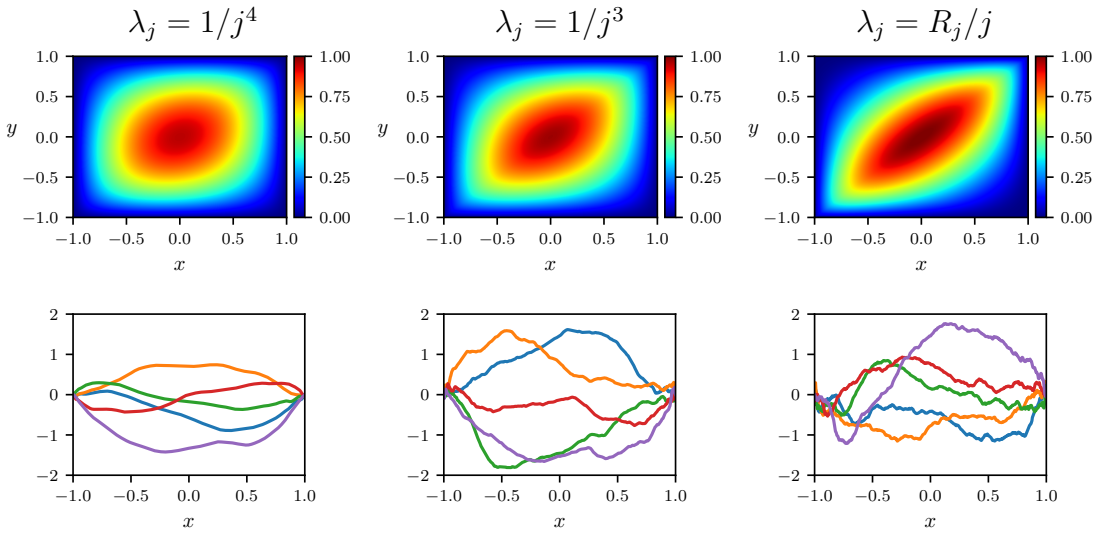


Figure 3.3: Covariance kernel $K_{\text{Jac}}^{(2,2)}$ constructed using Jacobi polynomials of type $(2, 2)$ with $\lambda_j = 1/j^4$, $1/j^3$, and R_j/j, where R_j is the Rissanen sequence (top). The bottom panels illustrate functions sampled from $\mathcal{GP}(0, K_{\text{Jac}}^{(2,2)})$ with the different eigenvalue sequences. The series for generating the random functions are truncated to $n = 500$.

3.4.1 Covariance matrix with prior knowledge

The approximation error bound in Theorem 3.1 depends on the eigenvalues of the covariance matrix, which dictates the distribution of the column vectors of the input matrix $\mathbf{\Omega}$. Roughly speaking, the more prior knowledge of the matrix $\mathbf{A}$ that can be incorporated into the covariance matrix, the better. In this numerical example, we investigate whether the standard randomized SVD, which uses the identity as its covariance matrix, can be improved by using a different covariance matrix. We then attempt to learn the discretized 2000×2000 matrix, *i.e.*, the discrete Green's function, of the inverse of the following differential operator:

$$\mathcal{L}u = d^2u/dx^2 - 100\sin(5\pi x)u, \quad x \in [0, 1].$$

We vary the number of columns (*i.e.* samples from the GP) in the input matrix $\mathbf{\Omega}$ from 1 to 2000.

In Figure 3.4(a), we compare the ratios between the relative error in the Frobenius norm given by the randomized SVD and the best approximation error, obtained by truncating the SVD of $\mathbf{A}$. The prior covariance matrix $\mathbf{K}$ consists of the discretized 2000×2000 matrix of the Green's function of the negative Laplace operator $\mathcal{L}u = -d^2u/dx^2$ on $[0, 1]$ to incorporate knowledge of the diffusion term in the matrix $\mathbf{A}$. We see that a nonstandard covariance matrix leads to a higher approximation

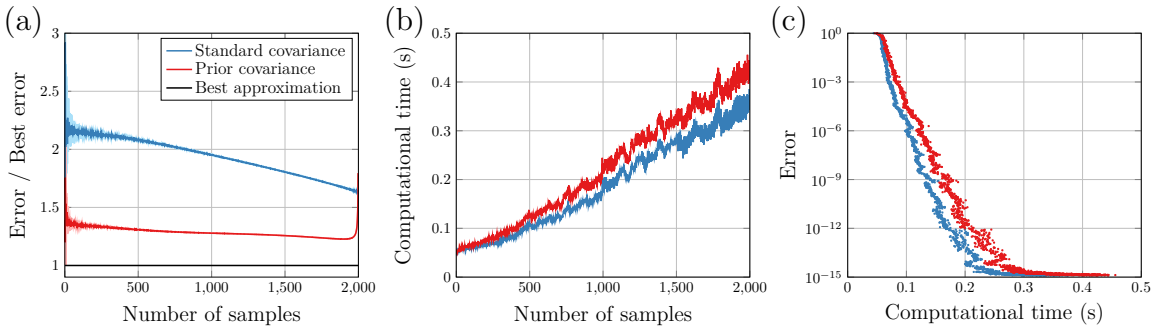


Figure 3.4: (a) Ratio between the average randomized SVD approximation error (over 10 runs) of the 2000×2000 matrix of the inverse of the differential operator $\mathcal{L}u = d^2u/dx^2 - 100\sin(5\pi x)u$ on $[0, 1]$, and the best approximation error. The error bars in light colour (blue and red) illustrate one standard deviation. (b) Average computational time of the algorithm (over 10 runs). The eigenvalue decomposition of the covariance matrix has been precomputed offline. (c) Randomized SVD approximation error with standard and prior covariance matrices with respect to the computational time.

accuracy, with a reduction of the error by a factor of 1.3-1.6 compared to the standard randomized SVD.

At the same time, the procedure is only 20% slower[1] on average (Figure 3.4(b)) as one can precompute the eigenvalue decomposition of the covariance matrix. Hence, sampling a random vector from a multivariate normal distribution with an arbitrary covariance matrix $\mathbf{K}$ can be computationally expensive when the dimension, n, of the matrix is large as it requires the computation of a Cholesky factorization, which can be done in $\mathcal{O}(n^3)$ operations. We highlight that this step can be precomputed once, such that the overhead of the generalized SVD can be essentially expressed as the cost of an extra matrix-vector multiplication. Then, the difference in timings between standard and prior covariance matrices is marginal as shown by Figure 3.4(b).

We observe in Figure 3.4(c) that using a standard covariance matrix offers a better trade-off between error and computational time. However, choosing a prior covariance matrix is of interest in applications where the sampling time is much higher than the numerical linear algebra costs to maximize the accuracy of the approximation matrix from a limited number of samples.

Additionally, we would like to highlight that prior covariance matrices can be designed and derived using physical knowledge of the problem, such as its diffusive

[1]Timings were performed on an Intel Xeon CPU E5-2667 v2 @ 3.30GHz using MATLAB R2020b without explicit parallelization.

nature, which can also significantly decrease the precomputation cost. In this example, we employ the discretized Green's function of the negative Laplacian operator with homogeneous Dirichlet boundary conditions, given by $\mathcal{L}u = -d^2u/dx^2$ on $[0,1]$, for which we know the eigenvalue decomposition. Hence, the eigenvalues and normalized eigenfunctions are respectively given by

$$\lambda_n = \frac{1}{\pi^2 n^2}, \quad \psi_n(x) = \sqrt{2}\sin(n\pi x), \quad x \in [0,1],\ n \geq 1.$$

Therefore, one can employ Mercer's representation (see Equation (3.6)) to sample the random vectors and precompute the covariance matrix in $\mathcal{O}(n^2)$ operations. For a problem of size $n = 2000$, it takes 0.16s to precompute the matrix.

3.4.2 Randomized SVD for Hilbert–Schmidt operators

We now apply the randomized SVD for HS operators to learn kernels of integral operators. In this first example, the kernel is defined as [215]

$$G(x,y) = \cos(10(x^2+y))\sin(10(x+y^2)), \quad x,y \in [-1,1],$$

and is displayed in Figure 3.5(a). We employ the squared-exponential covariance kernel K_{SE} with parameter $\ell = 0.01$ and $k = 100$ samples (see Equation (3.5)) to sample random functions from the associated GP. The learned kernel G_k is represented on the bottom panel of Figure 3.5(a) and has an approximation error around machine precision.

As a second application of the randomized SVD for HS operators, we learn the kernel $G(x,y) = \text{Ai}(-13(x^2y+y^2))$ for $x,y \in [-1,1]$, where Ai is the Airy function [166, Chapt. 9] defined by

$$\text{Ai}(x) = \frac{1}{\pi}\int_0^\infty \cos\left(\frac{t^3}{3} + xt\right)\,\mathrm{d}t, \quad x \in \mathbb{R}.$$

We plot the kernel and its low-rank approximant given by the randomized SVD for HS operators in Figure 3.5(b) and obtain an approximation error (measured in the L^2-norm) of 5.04×10^{-14}. The two kernels have a numerical rank equal to 42.

The last example consists of learning the HS operator associated with the kernel $G(x,y) = J_0(100(xy+y^2))$ for $x,y \in [-1,1]$, where J_0 is the Bessel function of the first kind [166, Chapt. 10] defined as

$$J_0(x) = \frac{1}{\pi}\int_0^\pi \cos(x\sin t)\,\mathrm{d}t, \quad x \in \mathbb{R},$$

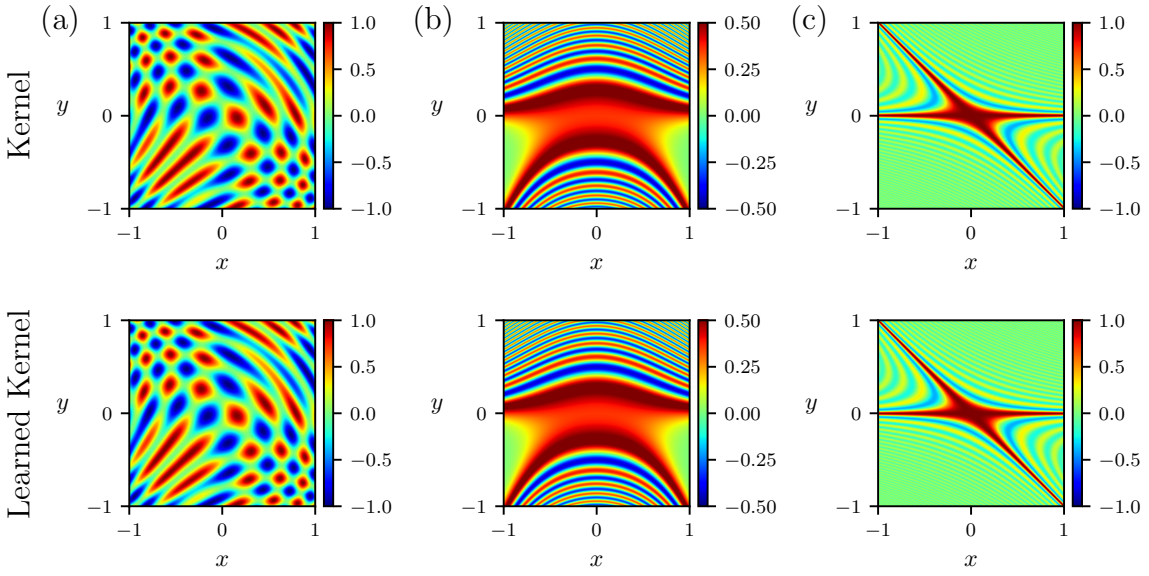


Figure 3.5: Kernels of three HS operators (top) together with the kernels learned by the randomized SVD for HS operators (bottom), using the squared-exponential covariance kernel K_{SE} with parameter $\ell = 0.01$ and one hundred functions sampled from $\mathcal{GP}(0, K_{\mathrm{SE}})$.

and plotted in Figure 3.5(c). The rank of this kernel is equal to 91 while its approximation is of rank 89 and the approximation error is equal to 4.88×10^{-13}. We observe that in the three numerical examples displayed in Figure 3.5, the differences between the learned and the original kernels are not visually perceptible.

Finally, we evaluate the influence of the choice of covariance kernel and number of samples in Figure 3.6. Here, we vary the number of samples from $k = 1$ to $k = 100$ and use the randomized SVD for HS operators with four different covariance kernels: the squared-exponential K_{SE} with parameters $\ell = 0.01, 0.1, 1$, and the Jacobi kernel $K_{\mathrm{Jac}}^{(2,2)}$ with eigenvalues $\lambda_j = 1/j^3$, for $j \geq 1$. In the left panel of Figure 3.6, we represent the eigenvalue ratio λ_j/λ_1 of the four kernels and observe that this quantity falls below machine precision for the squared-exponential kernel with $\ell = 1$ and $\ell = 0.1$ at $j = 13$ and $j = 59$, respectively. In Figure 3.6(right), we observe that these two kernels fail to approximate kernels of high numerical rank. The other two kernels have a much slower decay of eigenvalues and can capture (or learn) more complicated kernels. We then see in the right panel of Figure 3.6 that the relative approximation errors obtained using $K_{\mathrm{Jac}}^{(2,2)}$ and K_{SE} are close to the best approximation error given by the squared tail of the singular values of the integral kernel $G(x, y)$, *i.e.*, $(\sum_{j \geq k+1} \sigma_j^2)^{1/2}$. The overshoot in the error at $k = 100$ compared to the machine precision is due to the decay of the eigenvalues of the covariance kernels. Hence, spatial directions associated with

small eigenvalues are harder to learn accurately. This issue does not arise in finite dimensions with the standard randomized SVD because the covariance kernel used there is isotropic, *i.e.*, all its eigenvalues are equal to one. However, this choice is no longer possible for learning HS integral operators as the covariance kernel K must be squared-integrable. The relative approximation errors at $k = 100$ (averaged over 10 runs) using $K_{\text{Jac}}^{(2,2)}$ and K_{SE} with $\ell = 0.01$ are $\text{Error}(K_{\text{Jac}}^{(2,2)}) \approx 2.6 \times 10^{-11}$, and $\text{Error}(K_{\text{SE}}) \approx 5.7 \times 10^{-13}$, which gives a ratio of

$$\text{Error}(K_{\text{Jac}}^{(2,2)})/\text{Error}(K_{\text{SE}}) \approx 45.6. \tag{3.10}$$

However, the square-root of the ratio of the quality of the two kernels for $k = 91$ is equal to

$$\sqrt{\gamma_{91}(K_{\text{SE}})/\gamma_{91}(K_{\text{Jac}}^{(2,2)})} \approx 117.8, \tag{3.11}$$

which is of the same order of magnitude of Equation (3.10) as predicted by Theorem 3.1. In Equation (3.11), $\gamma_{91}(K_{\text{SE}}) \approx 5.88 \times 10^{-2}$ and $\gamma_{91}(K_{\text{Jac}}^{(2,2)}) \approx 4.24 \times 10^{-6}$ are both computed using Chebfun.

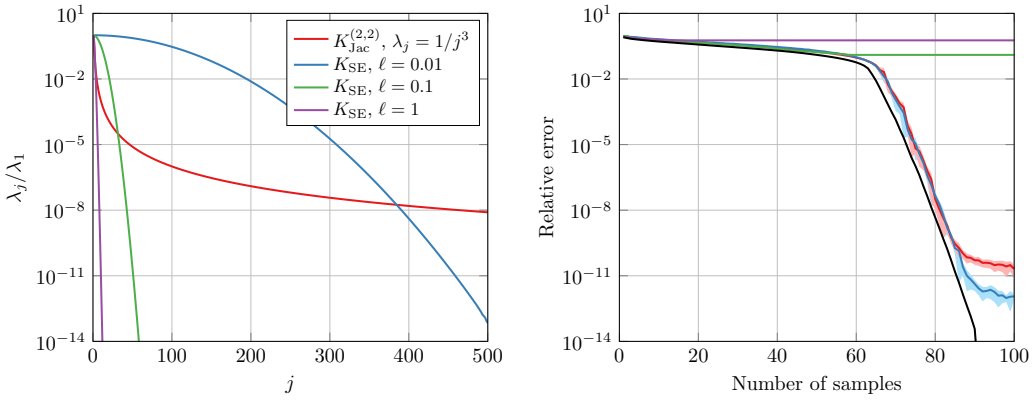


Figure 3.6: Left: Scaled eigenvalues of the Jacobi covariance kernel $K_{\text{Jac}}^{(2,2)}$ with sequence $\lambda_j = 1/j^3$ and squared-exponential kernels K_{SE} with parameters $\ell = 0.01, 0.1, 1$, respectively. Right: Average (over 10 runs) relative approximation error in the L^2-norm between the Bessel kernel $G(x, y) = J_0(100(xy + y^2))$ and its low-rank approximation $G_k(x, y)$, obtained from the randomized SVD by sampling the GPs k times. The error bars in light colour (blue and red) illustrate one standard deviation and the black line indicates the best approximation error given by the tail of the singular values of G.

In conclusion, this section provides numerical insights to motivate the choice of the covariance kernel to learn HS operators. Following Figure 3.6, a kernel with slowly decaying eigenvalues is preferable and yields better approximation errors or higher

learning rate with respect to the number of samples, especially when learning a kernel with a large numerical rank. The optimal choice from a theoretical viewpoint is to select a covariance kernel whose eigenvalues have a decay rate similar to the Rissanen sequence [189], but other choices may be preferable in practice to ensure smoothness of the sample functions (cf. Section 3.3.4).

Chapter 4

Rational neural networks*

A key question in designing deep learning architectures is the choice of the activation function to reduce the number of trainable parameters of the network while keeping the same approximation power [75]. While smooth activation functions such as sigmoid, logistic, or hyperbolic tangent are widely used, they suffer from the "vanishing gradient problem" [19] because their derivatives are zero for large inputs. Neural networks (NNs) based on polynomial activation functions are an alternative [40, 47, 77, 81, 139, 228], but can be numerically unstable due to large gradients for large inputs [19]. Moreover, polynomials do not approximate non-smooth functions efficiently [220], which can lead to optimization issues in classification problems. A popular choice of activation function is the Rectified Linear Unit (ReLU) defined as $\mathrm{ReLU}(x) = \max(x, 0)$ [97, 158]. It has numerous advantages, such as being fast to evaluate and zero for many inputs [73]. Many theoretical studies characterize and understand the expressiveness of shallow and deep ReLU neural networks from the perspective of approximation theory [52, 129, 150, 213, 242].

ReLU networks also suffer from drawbacks, which are most evident during training. The main disadvantage is that the gradient of ReLU is zero for negative real numbers. Therefore, its derivative is zero if the activation function is saturated [140]. Several adaptations to ReLU have been proposed over the past few years, such as Leaky ReLU [140], Exponential Linear Unit (ELU) [45], Parametric Linear Unit (PReLU) [87], and Scaled Exponential Linear Unit (SELU) [109], to improve the initialization and optimization of neural networks and avoid the use of batch normalization layers [95]. These modifications outperform ReLU in image classification

*This chapter is based on a paper with Yuji Nakatsukasa and Alex Townsend [30], published in NeurIPS 2020. Nakatsukasa and Townsend had advisory roles and proved Lemma 4.1. I proved the other theoretical results, performed the numerical experiments, and was the lead author in writing the paper.

applications, and some of these activation functions have trainable parameters, which are learned by gradient descent at the same time as the hyperparameters of the network. To obtain significant benefits for image classification and partial differential equation (PDE) solvers, one can also perform an exhaustive search over trainable activation functions constructed from standard units [96, 186].

In this chapter, we are motivated by designing activation functions with greater approximation power than ReLU. We study rational neural networks, which are neural networks with activation functions that are trainable rational functions. These will have improved theoretical guarantees on expressivity compared to ReLU, as we shall see in Section 4.2.

4.1 Definitions

We consider neural networks whose activation functions consist of rational functions with trainable coefficients a_i and b_j, *i.e.*, functions of the form:

$$F(x) = \frac{P(x)}{Q(x)} = \frac{\sum_{i=0}^{r_P} a_i x^i}{\sum_{j=0}^{r_Q} b_j x^j}, \quad a_P \neq 0,\, b_Q \neq 0, \tag{4.1}$$

where r_P and r_Q are the polynomial degrees of the numerator and denominator, respectively. We say that $F(x)$ is of type (r_P, r_Q) and degree $\max(r_P, r_Q)$.

The use of rational functions in deep learning is motivated by the theoretical work of Telgarsky, who proved error bounds on the approximation of ReLU neural networks by high-degree rational functions and vice versa [214]. On the practical side, neural networks based on rational activation functions are considered by Molina et al. [153], who defined a safe Padé Activation Unit (PAU) as

$$F(x) = \frac{\sum_{i=0}^{r_P} a_i x^i}{1 + |\sum_{j=1}^{r_Q} b_j x^j|}.$$

The denominator is selected so that $F(x)$ does not have poles located on the real axis. PAU networks can learn new activation functions and are competitive with state-of-the-art neural networks for image classification. However, this choice results in a non-smooth activation function and makes the gradient expensive to evaluate during training. In a closely related work, Chen et al. [39] propose high-degree rational activation functions in a neural network, which have benefits in terms of approximation power. However, this choice can significantly increase the number of parameters in the network, causing the training stage to be computationally expensive.

In this chapter, we use low-degree rational functions as activation functions, which are then composed together by the neural network to build high-degree rational functions. In this way, we can leverage the approximation power of high-degree rational functions without making training expensive. We highlight the approximation power of rational networks and provide optimal error bounds to demonstrate that rational neural networks theoretically outperform ReLU networks. Motivated by our theoretical results, we consider rational activation functions of type $(3, 2)$, *i.e.*, $r_P = 3$ and $r_Q = 2$. This type appears naturally in the theoretical analysis due to the composition property of Zolotarev sign functions (see Section 4.2.1): the degree of the overall rational function represented by the rational neural network is an enormous $3^{\#\text{layers}}$, while the number of trainable parameters only grows linearly with respect to the depth of the network. A low-degree activation function keeps the number of trainable parameters small, while the implicit composition in a neural network gives us the approximation power of high-degree rationals. This choice is also motivated empirically, and we do not claim that the type $(3, 2)$ is the best choice for all situations as the configurations may depend on the application as shown later by Figure 4.5. Our experiments[1] on the approximation of smooth functions and generative adversarial networks (GANs) suggest that rational neural networks are an attractive alternative to ReLU networks (see Section 4.3).

4.2 Theoretical results on rational neural networks

Here, we demonstrate the theoretical benefit of using neural networks based on rational activation functions due to their superiority over ReLU in approximating functions. We derive optimal bounds in terms of the total number of trainable parameters (also called size) needed by rational networks to approximate ReLU networks as well as functions in the Sobolev space $\mathcal{W}^{n,\infty}([0, 1]^d)$, where $n, d \geq 1$ are integers. Throughout this chapter, we take ϵ to be a small parameter with $0 < \epsilon < 1$. We first show that an ϵ-approximation on the domain $[-1, 1]^d$ of a ReLU network ($\mathcal{N}_{\text{ReLU}}$) by a rational neural network ($\mathcal{N}_{\text{Rational}}$) must have the following size (indicated in brackets):

$$\mathcal{N}_{\text{Rational}}[\Omega(\log(\log(1/\epsilon)))] \leq \mathcal{N}_{\text{ReLU}} \leq \mathcal{N}_{\text{Rational}}[\mathcal{O}(\log(\log(1/\epsilon)))], \tag{4.2}$$

where the constants only depend on the size and depth of the ReLU network. Here, the upper bound means that all ReLU networks can be approximated to within ϵ by a rational network of size $\mathcal{O}(\log(\log(1/\epsilon)))$. The lower bound means that there is a

[1] All code and hyperparameters are publicly available at [29].

ReLU network that cannot be ϵ-approximated by a rational network of size less than $C\log(\log(1/\epsilon))$, for some constant $C > 0$. In comparison, the size needed by a ReLU network to approximate a rational neural network within the tolerance of ϵ is given by the following inequalities:

$$\mathcal{N}_{\text{ReLU}}[\Omega(\log(1/\epsilon))] \leq \mathcal{N}_{\text{Rational}} \leq \mathcal{N}_{\text{ReLU}}[\mathcal{O}(\log(1/\epsilon))^3], \tag{4.3}$$

where the constants only depend on the size and depth of the rational neural network. This means that all rational networks can be approximated to within ϵ by a ReLU network of size $\mathcal{O}(\log(1/\epsilon))^3$, while there is a rational network that cannot be ϵ-approximated by a ReLU network of size less than $\Omega(\log(1/\epsilon))$. A comparison between (4.2) and (4.3) suggests that rational networks could be more expressive than ReLU.

4.2.1 Approximation of ReLU networks by rational neural networks

Telgarsky showed that neural networks and rational functions can approximate each other in the sense that there exists a rational function of degree $\mathcal{O}(\text{polylog}(1/\epsilon))$ that is ϵ-close to a ReLU network [214, Thm. 1.1], where $\epsilon > 0$ is a small number.

Theorem 4.1 (Telgarsky)**.** *Let $0 < \epsilon < 1$ and let $\|\cdot\|_1$ denote the vector 1-norm. The following two statements hold:*

1. Let k be a nonnegative integer and $p : [0,1]^d \to [-1,1]$, $q : [0,1]^d \to [2^{-k}, 1]$ be polynomials of degree $\leq r$, each with $\leq s$ monomials. Then, there exists a ReLU network $\mathcal{N}_{ReLU} : [0,1]^d \to \mathbb{R}$ of size

$$\mathcal{O}\left(k^7 \log(1/\epsilon)^3 + \min\left\{srk\log(sr/\epsilon), sdk^2\log(dsr/\epsilon)^2\right\}\right),$$

such that

$$\sup_{x\in[0,1]^d} \left|\mathcal{N}_{ReLU}(x) - \frac{p(x)}{q(x)}\right| \leq \epsilon.$$

2. Let $\mathcal{N}_{ReLU} : [-1,1]^d \to \mathbb{R}$ be a ReLU network with M layers and at most k nodes per layer, where each node computes $x \mapsto ReLU(a^\top x + b)$ and the pair (a,b) (possibly distinct across nodes) satisfies $\|a\|_1 + |b| \leq 1$. Then, there exists a rational function $R : [-1,1]^d \to \mathbb{R}$ with degree (maximum of numerator and denominator)

$$\mathcal{O}\left(k^M \log(M/\epsilon)^M\right),$$

such that

$$\sup_{x\in[-1,1]^d} |\mathcal{N}_{ReLU}(x) - R(x)| \leq \epsilon.$$

To prove this statement, Telgarsky used a rational function constructed with Newman polynomials [163] to obtain a rational approximation to the ReLU function that converges with square-root exponential accuracy. That is, Telgarsky needed a rational function of degree $\Omega(\log(1/\epsilon)^2)$ to achieve a tolerance of ϵ. A degree r rational function can be represented with $2(r+1)$ coefficients, *i.e.*, $a_0, \ldots, a_r$ and $b_0, \ldots, b_r$ in Equation (4.1). Therefore, the rational approximation to a ReLU network constructed by Telgarsky requires at least $\Omega(\text{polylog}(1/\epsilon))$ parameters. In contrast, for any rational function, Telgarsky showed that there exists a ReLU network of size $\mathcal{O}(\text{polylog}(1/\epsilon))$ that is an ϵ-approximation on $[0,1]^d$.

Our key observation is that by composing low-degree rational functions together, we can approximate a ReLU network much more efficiently in terms of the size (rather than the degree) of the rational network. Our theoretical work is based on a family of rationals called Zolotarev sign functions, which are the best rational approximation in the infinity norm on $[-1,-\ell] \cup [\ell,1]$, with $0 < \ell < 1$, to the sign function [3, 176], defined as

$$\text{sign}(x) = \begin{cases} -1, & x < 0, \\ 0, & x = 0, \\ 1, & x > 0. \end{cases}$$

We first show that a rational function can approximate the absolute value function $|x|$ on $[-1,1]$ with square-root exponential convergence using a composition of Zolotarev functions.

Lemma 4.1. *For any integer $k \geq 0$, we have*

$$\min_{r \in \mathcal{R}_{k,k}} \max_{x \in [-1,1]} ||x| - xr(x)| \leq 4e^{-\pi\sqrt{k/2}},$$

where $\mathcal{R}_{k,k}$ is the space of rational functions of type at most (k,k). Thus, $xr(x)$ is a rational approximant to $|x|$ of type at most $(k+1,k)$. Moreover, if $k = \prod_{i=1}^{p} k_i$ for some $p \geq 1$ and integers $k_1, \ldots, k_p \geq 2$, then r can be written as $r = R_p \circ \cdots \circ R_1$, where $R_i \in \mathcal{R}_{k_i,k_i}$.

Proof. Let $0 < \ell < 1$ be a real number and consider the sign function on the domain $[-1,-\ell] \cup [\ell,1]$, *i.e.*,

$$\text{sign}(x) = \begin{cases} -1, & x \in [-1,-\ell], \\ +1, & x \in [\ell,1]. \end{cases}$$

By [17, Equation (33)], we find that for any $k \geq 0$,

$$\min_{r \in \mathcal{R}_{k,k}} \max_{x \in [-1,-\ell] \cup [\ell,1]} |\text{sign}(x) - r(x)| \leq 4 \left[\exp\left(\frac{\pi^2}{2\log(4/\ell)} \right) \right]^{-k}.$$

Let $r(x)$ be the rational function of type (k,k) that attains the minimum [17, Equation (12)]. We refer to such $r(x)$ as the Zolotarev sign function. It is given by

$$r(x) = Mx\frac{\prod_{j=1}^{\lfloor (k-1)/2\rfloor} x^2 + c_{2j}}{\prod_{j=1}^{\lfloor k/2\rfloor} x^2 + c_{2j-1}}, \quad c_j = \ell^2\frac{\mathrm{sn}^2(jK(\kappa)/k;\kappa)}{1-\mathrm{sn}^2(jK(\kappa)/k;\kappa)}.$$

Here, M is a real constant selected so that $\mathrm{sign}(x) - r(x)$ equioscillates on $[-1,-\ell]\cup[\ell,1]$, $\kappa = \sqrt{1-\ell^2}$, $\mathrm{sn}(\cdot)$ is the first Jacobian elliptic function, and K is the complete elliptic integral of the first kind. Since $|x| = x\cdot\mathrm{sign}(x)$ we have the following inequality,

$$\begin{aligned}\max_{x\in[-1,-\ell]\cup[\ell,1]} ||x| - xr(x)| &= \max_{x\in[-1,-\ell]\cup[\ell,1]} |x\cdot\mathrm{sign}(x) - xr(x)| \\ &\leq \max_{x\in[-1,-\ell]\cup[\ell,1]} |\mathrm{sign}(x) - r(x)|\,.\end{aligned}$$

The last inequality follows because $|x| \leq 1$ on $[-1,-\ell]\cup[\ell,1]$. Moreover, since $xr(x) \geq 0$ for $x\in[-1,1]$ (see [17, Equation (12)]) we have

$$\max_{x\in[-\ell,\ell]} ||x| - xr(x)| \leq \max_{x\in[-\ell,\ell]} |x| \leq \ell.$$

Therefore,

$$\max_{x\in[-1,1]} ||x| - xr(x)| \leq \max\left\{\ell, 4\left[\exp\left(\frac{\pi^2}{2\log(4/\ell)}\right)\right]^{-k}\right\}.$$

Now, we select $0<\ell<1$ to minimize this upper bound. One finds that $\ell = 4\exp(-\pi\sqrt{k/2})$ and the result follows immediately.

For the final claim, let r be the Zolotarev sign function $Z_k(\cdot\,;\ell)$ of type (k,k) on $[-1,-\ell]\cup[\ell,1]$, with $k = \prod_{i=1}^p k_i$. By definition, $Z_k(\cdot;\ell)$ is the best rational approximation of degree k to the sign function on $[-1,-\ell]\cup[\ell,1]$. We know from [117, 160] that there exist p Zolotarev sign functions $R_1,\ldots,R_p$, where each R_i is of type (k_i,k_i), such that

$$r(x) \coloneqq Z_k(x;\ell) = R_p(\cdots(R_2(R_1(x)))\cdots). \tag{4.4}$$

□

A composition of $k\geq 1$ Zolotarev sign functions of type $(3,2)$ has type $(3^k, 3^k-1)$ but can be represented with $7k$ parameters instead of $2\times 3^k+1$. This property enables the construction of a rational approximation to ReLU using compositions of low-degree Zolotarev sign functions with $\mathcal{O}(\log(\log(1/\epsilon)))$ parameters in Lemma 4.2. The proof of Lemma 4.2 is a direct consequence of the previous lemma and the properties of Zolotarev sign functions.

Lemma 4.2. *Let $0 < \epsilon < 1$. There exists a rational network $\mathcal{N}_{Rational} : [-1, 1] \to [-1, 1]$ of size $\mathcal{O}(\log(\log(1/\epsilon)))$ such that*

$$\|\mathcal{N}_{Rational} - ReLU\|_\infty \coloneqq \max_{x\in[-1,1]} |\mathcal{N}_{Rational}(x) - ReLU(x)| \leq \epsilon.$$

Moreover, no rational network of size smaller than $\Omega(\log(\log(1/\epsilon)))$ can achieve this.

Proof. Let $0 < \epsilon < 1$, $0 < \ell < 1$, $k \geq 1$, and r be the Zolotarev sign function $Z_{3^k}(\cdot\,; \ell)$ of type $(3^k, 3^k - 1)$. Again from [117, 160], we see that there exist k Zolotarev sign functions $R_1, \ldots, R_k$ of type $(3, 2)$ such that their composition equals $Z_{3^k}(x; \ell)$, *i.e.*,

$$r(x) \coloneqq Z_{3^k}(x; \ell) = R_k(\cdots(R_2(R_1(x))\cdots). \tag{4.5}$$

Following the proof of Lemma 4.1, we have the inequality

$$\max_{x\in[-1,1]} ||x| - xr(x)| \leq 4e^{-\pi\sqrt{3^k/2}}, \tag{4.6}$$

where we chose $\ell = 4\exp(-\pi\sqrt{3^k/2})$. Now, we take

$$k = \left\lceil \frac{\ln(2/\pi^2) + 2\ln(\ln(4/\epsilon))}{\ln(3)} \right\rceil, \tag{4.7}$$

so that the right-hand side of Equation (4.6) is bounded by ϵ. Finally, we use the identity

$$\mathrm{ReLU}(x) = \frac{|x| + x}{2}, \quad x \in \mathbb{R},$$

to define a rational approximation to the ReLU function on the interval $[-1, 1]$ as

$$\tilde{r}(x) = \frac{1}{2}\left(\frac{xr(x)}{1+\epsilon} + x\right).$$

Therefore, we have the following inequalities for $x \in [-1, 1]$,

$$\begin{aligned}|\mathrm{ReLU}(x) - \tilde{r}(x)| &= \frac{1}{2}\left||x| - \frac{xr(x)}{1+\epsilon}\right| \leq \frac{1}{2(1+\epsilon)}(||x| - xr(x)| + \epsilon|x|) \\ &\leq \frac{\epsilon}{1+\epsilon} \leq \epsilon.\end{aligned}$$

Then, r is a composition of k rational functions of type $(3, 2)$ and can be represented using at most $7k$ coefficients (see Equation (4.4)). Moreover, using Equation (4.7), we see that $k = \mathcal{O}(\log(\log(1/\epsilon)))$, which means that $\tilde{r}$ is representable by a rational network of size $\mathcal{O}(\log(\log(1/\epsilon)))$. Finally, $|\tilde{r}(x)| \leq 1$ for $x \in [-1, 1]$.

The lower bound on the rational networks size will be proved separately later in Proposition 4.1. □

The upper bound on the complexity of the neural network obtained in Lemma 4.2 is optimal, as proved by Vyacheslavov [232].

Theorem 4.2 (Vyacheslavov)**.** *The following inequalities hold:*

$$C_1 e^{-\pi\sqrt{k}} \leq \max_{x\in[-1,1]} ||x| - r_k(x)| \leq C_2 e^{-\pi\sqrt{k}}, \quad k \geq 0, \tag{4.8}$$

where r_k is the best rational approximation to $|x|$ in $[-1,1]$ from $\mathcal{R}_{k,k}$. Here, $C_1, C_2 > 0$ are constants that are independent of k.

We first deduce the following corollary, giving lower and upper bounds on the optimal rational approximation to the ReLU function.

Corollary 4.1. *The following inequalities hold:*

$$\frac{C_1}{2} e^{-\pi\sqrt{k}} \leq \|ReLU - r_k\|_\infty \leq \frac{C_2}{2} e^{-\pi\sqrt{k}}, \quad k \geq 0, \tag{4.9}$$

where r_k is the best rational approximation to ReLU on $[-1,1]$ in $\mathcal{R}_{k,k}$ and $C_1, C_2 > 0$ are constants given by Theorem 4.2.

Proof. Let k be an integer and let $r_k \in \mathcal{R}_{k,k}$ be any rational function of degree $\leq k$. Now, define $r_{\text{abs}}(x) = 2r_k(x) - x$. Since $\text{ReLU}(x) = (|x| + x)/2$, we have

$$\begin{aligned}\|\text{ReLU} - r_k\|_\infty &= \max_{x\in[-1,1]} \left|\frac{1}{2}(r_{\text{abs}}(x) + x) - \frac{1}{2}(|x| + x)\right| = \max_{x\in[-1,1]} \frac{1}{2}\left|r_{\text{abs}}(x) - |x|\right| \\ &\geq \frac{1}{2} C_1 e^{-\pi\sqrt{k}},\end{aligned}$$

where the inequality is from Theorem 4.2. Now, let $r_k \in \mathcal{R}_{k,k}$ be the best rational approximation to $|x|$ on $[-1,1]$. Now, define $r_{\text{ReLU}}(x) = (r_k(x) + x)/2$. We find that

$$\begin{aligned}\|\text{ReLU} - r_{\text{ReLU}}\|_\infty &= \max_{x\in[-1,1]} \left|\frac{1}{2}(|x| + x) - \frac{1}{2}(r_k(x) + x)\right| = \max_{x\in[-1,1]} \frac{1}{2}\left||x| - r_k(x)\right| \\ &\leq \frac{1}{2} C_2 e^{-\pi\sqrt{k}},\end{aligned}$$

which proves that the best approximation to ReLU satisfies the upper bound. □

We now show that a rational neural network must be at least $\Omega(\log(\log(1/\epsilon)))$ in size (total number of nodes) to approximate the ReLU function to within ϵ.

Proposition 4.1. *Let $0 < \epsilon < 1$. A rational neural network that approximates the ReLU function on $[-1,1]$ to within ϵ has size of at least $\Omega(\log(\log(1/\epsilon)))$.*

Proof. Let $\mathcal{N}_{\text{Rational}} : [-1,1] \to \mathbb{R}$ be a rational neural network with $k_1, \ldots, k_M \geq 1$ nodes at each of its M layers, and assume that its activation functions are rational functions of type at most (r_P, r_Q). Let $d_r = \max(r_P, r_Q)$ be the maximum of the degrees of the activation functions of $\mathcal{N}_{\text{Rational}}$. Such a network has size $\sum_{i=1}^{M} k_i$. Note that $\mathcal{N}_{\text{Rational}}$ itself is a rational function of degree d, where from additions and compositions of rational functions we have $d \leq d_r^M \prod_{i=1}^{M} k_i$. If $\mathcal{N}_{\text{Rational}}$ is an ϵ-approximation to the ReLU function on $[-1,1]$, we know by Corollary 4.1 that

$$\frac{C_1}{2} e^{-\pi\sqrt{d}} \geq \epsilon, \quad d \geq \left(\frac{1}{\pi} \ln\left(\frac{C_1}{2\epsilon} \right) \right)^2. \tag{4.10}$$

The statement follows by minimizing the size of $\mathcal{N}_{\text{Rational}}$, *i.e.*, $\sum_{i=1}^{M} k_i$ subject to

$$d_r^M \prod_{i=1}^{M} k_i \geq \left(\frac{1}{\pi} \ln\left(\frac{C_1}{2\epsilon} \right) \right)^2.$$

That is,

$$\sum_{i=1}^{M} \ln(k_i) + M \ln(d_r) \geq 2 \ln\left(\ln\left(\frac{C_1}{2\epsilon} \right) \right) - 2\ln(\pi). \tag{4.11}$$

We introduce a Lagrange multiplier $\lambda \in \mathbb{R}$ and define the Lagrangian of this optimization problem as

$$\mathcal{L}(k_1, \ldots, k_M, \lambda) = \sum_{i=1}^{M} k_i + \lambda \left[2 \ln\left(\ln\left(\frac{C_1}{2\epsilon} \right) \right) - 2\ln(\pi) - \sum_{i=1}^{M} \ln(k_i) - M \ln(d_r) \right].$$

One finds using the Karush–Kuhn–Tucker conditions [115] that $k_1 = \cdots = k_M = \lambda$. Then, using Equation (4.11), we find that λ satisfies

$$\ln(\lambda) \geq \frac{2}{M} \left[\ln\left(\ln\left(\frac{C_1}{2\epsilon} \right) \right) - \ln(\pi) \right] - \ln(d_r) =: \ln(\lambda^*). \tag{4.12}$$

Therefore, the rational network $\mathcal{N}_{\text{Rational}}$ with M layers that approximates the ReLU function to within ϵ on $[-1,1]$ has a size of at least $s(M) \coloneqq M\lambda^*$, where λ^* is given by Equation (4.12) and depends on M. We now minimize $s(M)$ with respect to the number of layers $M \geq 1$. We remark that minimizing s is equivalent of minimizing $\ln(s)$, where

$$\ln(s(M)) = \ln(M) + \ln(\lambda^*) = \ln(M) + \frac{2}{M} \left[\ln\left(\ln\left(\frac{C_1}{2\epsilon} \right) \right) - \ln(\pi) \right] - \ln(d_r).$$

One finds that one should take $k_1 = \cdots = k_M = \lambda^* = \mathcal{O}(1)$ and $M = \Omega(\log(\log(1/\epsilon)))$. The result follows. □

The proof of Proposition 4.1 shows that the bound obtained in Lemma 4.2 is optimal in the sense that a rational network requires at least $\Omega(\log(\log(1/\epsilon)))$ parameters to approximate the ReLU function on $[-1,1]$ to within the tolerance $\epsilon > 0$. The convergence of the Zolotarev sign functions to the ReLU function is much faster, with respect to the number of parameters, than the rational constructed with Newman polynomials (see Figure 4.1(left)). We also include in this panel the algebraic convergence of $\mathcal{O}(1/\epsilon)$ obtained by polynomials [220] as a comparison.

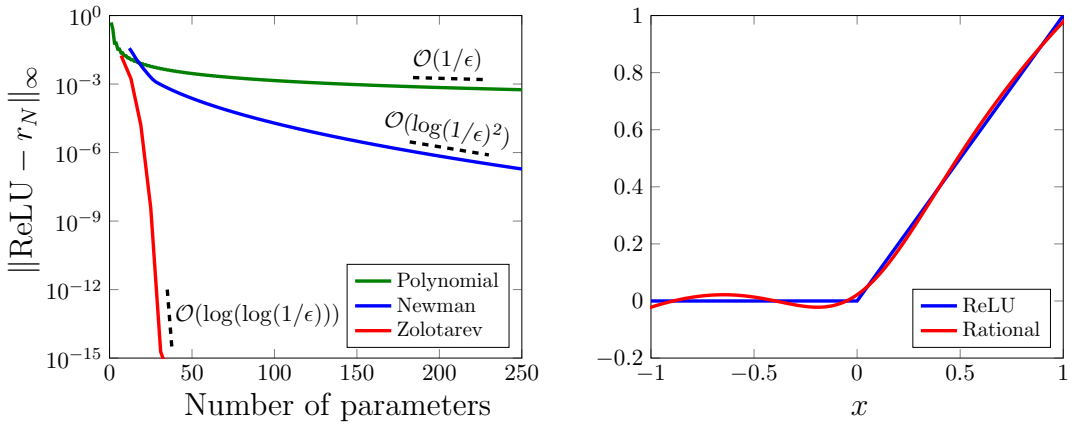


Figure 4.1: Left: Approximation error $\|\mathrm{ReLU} - r_N\|_\infty$ of the Newman (blue), Zolotarev sign functions (red), and best polynomial approximation [168] of degree $N-1$ (green) r_N to ReLU with respect to the number of parameters required to represent r_N. Right: Best rational function of type $(3,2)$ (red) that approximates the ReLU function (blue). We use this to initialize the rational activation functions when training a rational neural network.

The converse of Lemma 4.2, which is a consequence of a theorem proved by Telgarsky [214, Theorem 1.1], shows that any rational function can be approximated by a ReLU network of size at most $\mathcal{O}(\log(1/\epsilon)^3)$.

Lemma 4.3. *Let $0 < \epsilon < 1$. If $R : [-1,1] \to [-1,1]$ is a rational function, then there exists a ReLU network $\mathcal{N}_{ReLU} : [-1,1] \to [-1,1]$ of size $\mathcal{O}(\log(1/\epsilon)^3)$ such that $\|R - \mathcal{N}_{ReLU}\|_\infty \leq \epsilon$.*

Proof. Let $0 < \epsilon < 1$ and $R : [-1,1] \to [-1,1]$ be a rational function. Take $\tilde{R}(x) = R(2x-1)$, which is still a rational function. Without loss of generality, we can assume that $\tilde{R}$ is an irreducible rational function (otherwise cancel factors till it is irreducible). Since $\tilde{R}$ is a rational, it can be written as $\tilde{R} = p/q$ with $\max_{x\in[0,1]} |q(x)| = 1$. Moreover, we know that $\tilde{R}(x) \in [-1,1]$ for $x \in [0,1]$ so we can assume that $q(x) \geq 0$ for $x \in [0,1]$ (it is either positive or negative by continuity). Since R is

continuous on $[-1,1]$, there is an integer $n \geq 1$ such that $q(x) \in [2^{-n}, 1]$ for $x \in [0,1]$. Furthermore, we find that $|p(x)| \leq 1$ for $x \in [0,1]$ because $|R(x)| \leq 1$ and $|q(x)| \leq 1$ for $x \in [0,1]$. By [214, Theorem 1.1], there exists a ReLU network $\mathcal{N}_{\mathrm{ReLU}} : [0,1] \to \mathbb{R}$ of size $\mathcal{O}(n^7 \log(1/\epsilon)^3)$ such that

$$\max_{x \in [0,1]} \left| \mathcal{N}_{\mathrm{ReLU}}(x) - \frac{p(x)}{q(x)} \right| \leq \frac{\epsilon}{2}.$$

We now define a scaled ReLU network $\tilde{\mathcal{N}}_{\mathrm{ReLU}}(x) = \mathcal{N}_{\mathrm{ReLU}}(x)/(1+\epsilon/2)$ such that $|\tilde{\mathcal{N}}_{\mathrm{ReLU}}(x)| \leq 1$ for $x \in [0,1]$. Therefore, for all $x \in [0,1]$,

$$\left| \tilde{\mathcal{N}}_{\mathrm{ReLU}}(x) - \tilde{R}(x) \right| = \left| \frac{\mathcal{N}_{\mathrm{ReLU}}(x)}{1+\epsilon/2} - \frac{p(x)}{q(x)} \right| \leq \frac{1}{1+\epsilon/2} \left(\left| \mathcal{N}_{\mathrm{ReLU}}(x) - \frac{p(x)}{q(x)} \right| + \frac{\epsilon}{2} \left| \frac{p(x)}{q(x)} \right| \right) \leq \epsilon.$$

Therefore, $x \mapsto \tilde{\mathcal{N}}_{\mathrm{ReLU}}((x+1)/2)$ is a ReLU neural network of size $\mathcal{O}(\log(1/\epsilon)^3)$ that is an ϵ-approximation to R on $[-1,1]$. □

To demonstrate the improved approximation power of rational neural networks over ReLU networks ($\mathcal{O}(\log(\log(1/\epsilon)))$ versus $\mathcal{O}(\log(1/\epsilon)^3)$), it is known that a ReLU network that approximates x^2, which is rational, to within ϵ on $[-1,1]$ must be of size at least $\Omega(\log(1/\epsilon))$ [129, Theorem 11].

We can now state our main theorem based on Lemmas 4.2 and 4.3. Theorem 4.3 provides bounds on the approximation power of ReLU networks by rational neural networks and vice versa. We regard Theorem 4.3 as an analogue of [214, Thm. 1.1] for our Zolotarev sign functions, where we are counting the number of training parameters instead of the degree of the rational functions. In particular, our rational networks have high degrees but can be represented with few parameters due to compositions, making training more computationally efficient. While Telgarsky required a rational function with $\mathcal{O}(k^M \log(M/\epsilon)^M)$ parameters to approximate a ReLU network with fewer than k nodes in each of M layers to within a tolerance of ϵ, we construct a rational network that only has size $\mathcal{O}(kM \log(\log(M/\epsilon)))$.

Theorem 4.3. *Let $0 < \epsilon < 1$ and let $\|\cdot\|_1$ denote the vector 1-norm. The following two statements hold:*

1. Let $\mathcal{N}_{Rational} : [-1,1]^d \to [-1,1]$ be a rational network with M layers and at most k nodes per layer, where each node computes $x \mapsto r(a^\top x + b)$ and r is a rational function with Lipschitz constant L (a, b, and r are possibly distinct across nodes). Suppose further that $\|a\|_1 + |b| \leq 1$ and $r : [-1,1] \to [-1,1]$. Then, there exists a ReLU network $\mathcal{N}_{ReLU} : [-1,1]^d \to [-1,1]$ of size

$$\mathcal{O}\left(kM \log(ML^M/\epsilon)^3\right)$$

such that $\max_{x\in[-1,1]^d} |\mathcal{N}_{Rational}(x) - \mathcal{N}_{ReLU}(x)| \leq \epsilon$.
2. Let $\mathcal{N}_{ReLU} : [-1,1]^d \to [-1,1]$ *be a ReLU network with* M *layers and at most* k *nodes per layer, where each node computes* $x \mapsto ReLU(a^\top x + b)$ *and the pair* (a,b) *(possibly distinct across nodes) satisfies* $\|a\|_1 + |b| \leq 1$. *Then, there exists a rational network* $\mathcal{N}_{Rational} : [-1,1]^d \to [-1,1]$ *of size*

$$\mathcal{O}(kM \log(\log(M/\epsilon)))$$

such that $\max_{x\in[-1,1]^d} |\mathcal{N}_{ReLU}(x) - \mathcal{N}_{Rational}(x)| \leq \epsilon$.

Proof. The statement of Theorem 4.3 comes in two parts, and we prove them separately. The structure of the proof closely follows [214, Lemma 1.3].

1. Consider the subnetwork H of the rational network $\mathcal{N}_{\text{Rational}}$, consisting of the layers of $\mathcal{N}_{\text{Rational}}$ up to the Jth layer for some $1 \leq J \leq M-1$. Let H_{ReLU} denote the ReLU network obtained by replacing each rational function r_{ij} in H by a ReLU network approximation $f_{r_{ij}}$ at a given tolerance $\epsilon_j > 0$ for $1 \leq j \leq J$ and $1 \leq i \leq k_j$, such that $|H_{\text{ReLU}}(x)| \leq 1$ for $x \in [-1,1]$ (see Lemma 4.3). Let $x \mapsto r_{i,J+1}(a_{i,J+1}^\top H(x) + b_{i,J+1})$ be the output of the rational network $\mathcal{N}_{\text{Rational}}$ at layer $J+1$ and node i for $1 \leq i \leq k_J$. Now, approximate node i in the $(J+1)$st layer by a ReLU network $f_{r_{i,J+1}}$ with tolerance $\epsilon_{J+1} > 0$ (see Lemma 4.3). The approximation error $E_{i,J+1}$ between the rational and the approximating ReLU network at layer $J+1$ and node i satisfies

$$\begin{aligned}
E_{i,J+1} &= |f_{r_{i,J+1}}(a_{i,J+1}^\top H_{\text{ReLU}}(x) + b_{i,J+1}) - r_{i,J+1}(a_{i,J+1}^\top H(x) + b_{i,J+1})| \\
&\leq \underbrace{|f_{r_{i,J+1}}(a_{i,J+1}^\top H_{\text{ReLU}}(x) + b_{i,J+1}) - r_{i,J+1}(a_{i,J+1}^\top H_{\text{ReLU}}(x) + b_{i,J+1})|}_{(1)} \\
&+ \underbrace{|r_{i,J+1}(a_{i,J+1}^\top H_{\text{ReLU}}(x) + b_{i,J+1}) - r_{i,J+1}(a_{i,J+1}^\top H(x) + b_{i,J+1})|}_{(2)}.
\end{aligned}$$

The first term is bounded by

$$(1) \leq \max_{x\in[-1,1]} \left| r_{i,J+1}(x) - f_{r_{i,J+1}} \right| \leq \epsilon_{J+1},$$

since $\left|a_{i,J+1}^\top H_{\text{ReLU}}(x) + b_{i,J+1}\right| \leq \|a_{i,J+1}\|_1 + |b_{i,J+1}| \leq 1$ by assumption. The second term is bounded as the Lipschitz constant of $r_{i,J+1}$ is at most L. That is,

$$(2) \leq L\|a_{i,J+1}\|_1 \max_{x\in[-1,1]^d} \|H_{\text{ReLU}}(x) - H(x)\|_\infty \leq L \max_{x\in[-1,1]^d} \|H_{\text{ReLU}}(x) - H(x)\|_\infty,$$

where we used the fact that $\|a_{i,J+1}\|_1 \leq 1$ and $\|H_{\mathrm{ReLU}}(x)\|_\infty \leq 1$ for $x \in [-1,1]^d$. We find that we have the following set of inequalities:

$$\max_{1\leq i\leq k_{j+1}} E_{i,j+1} \leq L \max_{1\leq i\leq k_j} E_{i,j} + \epsilon_{j+1}, \quad 1 \leq i \leq k_j, \quad 1 \leq j \leq J+1,$$

with $E_{i,0} = 0$. If we select $\epsilon_j = \epsilon L^{j-J-1}/(J+1)$, then we find that $\max_{1\leq i\leq k_{J+1}} E_{i,J+1} \leq \epsilon$. When $J = M-1$, the ReLU network approximates the original rational network, $\mathcal{N}_{\mathrm{Rational}}$, and the ReLU network has size

$$\mathcal{O}\left(k \sum_{j=1}^{M} \log\left(\frac{M}{L^{j-M}\epsilon}\right)^3\right).$$

where we used the fact that $k_j \leq k$ for $1 \leq j \leq M$. This can be simplified a little since

$$\sum_{j=1}^{M} \log\left(\frac{M}{L^{j-M}\epsilon}\right)^3 = \sum_{j=1}^{M} \big(\log(ML^M/\epsilon) + j\log(1/L)\big)^3 = \mathcal{O}\big(M\log(ML^M/\epsilon)^3\big).$$

2. Telgarsky proved in [214, Lemma 1.3] that if H_R is a neural network obtained by replacing all the ReLU activation functions in $\mathcal{N}_{\mathrm{ReLU}}$ by rational functions R for $1 \leq j \leq M$, which satisfies $R(x) \in [-1,1]$ and $|R(x) - \mathrm{ReLU}(x)| \leq \epsilon/M$ for $x \in [-1,1]$, then

$$\max_{x\in[-1,1]^d} |\mathcal{N}_{\mathrm{ReLU}}(x) - H_R(x)| \leq \epsilon.$$

Let $\tilde{R}$ be a rational neural network approximating ReLU with a tolerance of ϵ/M, constructed by Lemma 4.2. Then, $\tilde{R}$ is rational network of size $\mathcal{O}(\log(\log(M/\epsilon)))$ and thus, $H_{\tilde{R}}$ is a rational neural network of size $\mathcal{O}(Mk\log(\log(M/\epsilon)))$. □

Theorem 4.3 highlights the improved approximation power of rational neural networks over ReLU networks. ReLU networks of size $\mathcal{O}(\mathrm{polylog}(1/\epsilon))$ are required to approximate rational networks while rational networks of size only $\mathcal{O}(\log(\log(1/\epsilon)))$ are sufficient to approximate ReLU networks.

4.2.2 Approximation of functions by rational networks

A important question is the required size and depth of deep neural networks to approximate smooth functions [129, 154, 242]. In this section, we consider the approximation theory of rational networks. In particular, we consider the approximation

of functions in the Sobolev space $\mathcal{W}^{n,\infty}([0,1]^d)$, where $n \geq 1$ is the regularity of the functions and $d \geq 1$. The norm of a function $f \in \mathcal{W}^{n,\infty}([0,1]^d)$ is defined as

$$\|f\|_{\mathcal{W}^{n,\infty}([0,1]^d)} = \max_{|\mathbf{n}| \leq n} \operatorname*{ess\,sup}_{\mathbf{x} \in [0,1]^d} |D^{\mathbf{n}} f(\mathbf{x})|,$$

where $\mathbf{n}$ is the multi-index $\mathbf{n} = (n_1, \ldots, n_d) \in \{0, \ldots, n\}^d$, and $D^{\mathbf{n}} f$ is the corresponding weak derivative of f. In this section, we consider the approximation of functions from

$$F_{d,n} \coloneqq \{f \in \mathcal{W}^{n,\infty}([0,1]^d), \quad \|f\|_{\mathcal{W}^{n,\infty}([0,1]^d)} \leq 1\}.$$

By the Sobolev embedding theorem [33], $F_{d,n}$ contains the functions in $\mathcal{C}^{n-1}([0,1]^d)$, which is the class of functions whose first $n-1$ derivatives are Lipschitz continuous. Yarotsky derived upper bounds on the size of neural networks with piecewise linear activation functions needed to approximate functions in $F_{d,n}$ [242, Thm. 1]. In particular, Yarotsky constructed an ϵ-approximation to functions in $F_{d,n}$ with a ReLU network of size at most $\mathcal{O}(\epsilon^{-d/n} \log(1/\epsilon))$ and depth smaller than $\mathcal{O}(\log(1/\epsilon))$.

Theorem 4.4 (Yarotsky)**.** *Let $d \geq 1$, $n \geq 1$, $0 < \epsilon < 1$, and $f \in F_{d,n}$. There exists a ReLU neural network $\mathcal{N}_{ReLU}$ of size*

$$\mathcal{O}(\epsilon^{-d/n} \log(1/\epsilon))$$

and maximum depth $\mathcal{O}(\log(1/\epsilon))$ such that $\|f - \mathcal{N}_{ReLU}\|_\infty \leq \epsilon$.

The term $\epsilon^{-d/n}$ in Theorem 4.4 is introduced by a local Taylor approximation, while the $\log(1/\epsilon)$ term is the size of the ReLU network needed to approximate monomials, *i.e.*, x^j for $j \geq 0$, in the Taylor series expansion. We now present an analogue of Theorem 4.4 for a rational neural network.

Theorem 4.5. *Let $d \geq 1$, $n \geq 1$, $0 < \epsilon < 1$, and $f \in F_{d,n}$. There exists a rational neural network $\mathcal{N}_{Rational}$ of size*

$$\mathcal{O}(\epsilon^{-d/n} \log(\log(1/\epsilon)))$$

and maximum depth $\mathcal{O}(\log(\log(1/\epsilon)))$ such that $\|f - \mathcal{N}_{Rational}\|_\infty \leq \epsilon$.

The proof of Theorem 4.5 consists of approximating f by a local Taylor expansion. One needs to approximate the piecewise linear functions and monomials arising in the Taylor expansion by rational networks. The main distinction between Yarotsky's argument and the proof of Theorem 4.5 is that monomials can be represented by rational neural networks with a size that does not depend on the accuracy of ϵ. In

contrast, ReLU networks require $\mathcal{O}(\log(1/\epsilon))$ parameters. Meanwhile, while ReLU neural networks can exactly approximate piecewise linear functions with a constant number of parameters, rational networks can approximate them with a size of a most $\mathcal{O}(\log(\log(1/\epsilon)))$ (see Lemma 4.2). That is, rational neural networks approximate piecewise linear functions much faster than ReLU networks approximate polynomials. This allows the existence of a rational network approximation to f with exponentially smaller depth ($\mathcal{O}(\log(\log(1/\epsilon)))$) than the ReLU networks constructed by Yarotsky.

We first show that the construction in Lemma 4.2 can approximate any piecewise linear function on $[-1, 1]$.

Proposition 4.2. *Let $0 < \epsilon < 1$ and let $g : [0, 1] \to \mathbb{R}$ be any continuous piecewise linear function with $m \geq 1$ breakpoints and Lipschitz constant $L > 0$. Then, there exists a rational neural network $\mathcal{N}_{Rational} : [0, 1] \to \mathbb{R}$ of size at most*

$$\mathcal{O}(m \log(\log(L/\epsilon)))$$

such that $\max_{x\in[0,1]} |g(x) - \mathcal{N}_{Rational}(x)| \leq \epsilon$.

Proof. Let $0 \leq b_1 < \cdots < b_M \leq 1$ be the breakpoints of g. In a similar way to the proof of [242, Proposition 1], we first express ρ as the following sum:

$$g(x) = c_0 \mathrm{ReLU}(b_1 - x) + \sum_{j=1}^{m} c_j \mathrm{ReLU}(x - b_j) + c_{m+1}, \tag{4.13}$$

for some constants $c_0, \ldots, c_{m+1} \in \mathbb{R}$. Therefore, g can be exactly represented using a ReLU network with $m + 1$ nodes and one layer, *i.e.*,

$$g(x) = \begin{pmatrix} c_0 & c_1 & \cdots & c_m \end{pmatrix} \begin{pmatrix} \mathrm{ReLU}(-x + b_1) \\ \mathrm{ReLU}(x - b_1) \\ \vdots \\ \mathrm{ReLU}(x - b_m) \end{pmatrix} + c_{m+1}.$$

Since g has a Lipschitz constant of L, we find that $|c_0| \leq L$ and $\sum_{j=1}^{m} |c_j| \leq L$. Using Lemma 4.2 we can approximate a ReLU function on $[-1, 1]$ with tolerance $\epsilon/(2L)$ by a rational network R_{ReLU} of size $\mathcal{O}(\log(\log(2L/\epsilon)))$. Now, we construct $\mathcal{N}_{\mathrm{Rational}} : [0, 1] \to \mathbb{R}$ as a rational network obtained by replacing the ReLU functions in g by R_{ReLU}. We have the following error estimate:

$$\max_{x\in[0,1]} |g(x) - \mathcal{N}_{\mathrm{Rational}}(x)| \leq |c_0| \|\mathrm{ReLU} - R_{\mathrm{ReLU}}\|_\infty + \sum_{j=1}^{m} |c_j| \|\mathrm{ReLU} - R_{\mathrm{ReLU}}\|_\infty \leq \frac{\epsilon}{2} + \frac{\epsilon}{2} \leq \epsilon.$$

The result follows as $\mathcal{N}_{\mathrm{Rational}}$ is of size $\mathcal{O}(m \log(\log(L/\epsilon)))$. □

We remark that the size of the rational network required to approximate a piecewise linear function depends on ϵ. In contrast, ReLU neural networks can represent piecewise linear functions exactly. In the next proposition, we show that a rational neural network can represent x^n, for some integer n, exactly.

Proposition 4.3. *Let $n \geq 1$, $r_P \geq 2$, and $r_Q \geq 0$. There exists a rational network $\mathcal{N}_{Rational}$, with rational activation functions of type (r_P, r_Q), of size at most $5\lfloor \log_{r_P}(n) \rfloor^2 + 1$ such that $\mathcal{N}_{Rational}(x) = x^n$ for all $x \in \mathbb{R}$.*

Proof. We start by expressing n in base r_P, *i.e.*,

$$n = \sum_{\ell=0}^{\lfloor \log_{r_P}(n) \rfloor} c_\ell r_P^\ell, \qquad c_\ell \in \{0, 1, \ldots, r_P - 1\}.$$

This means we can represent x^n as

$$x^n = \prod_{\ell=0}^{\lfloor \log_{r_P}(n) \rfloor} x^{c_\ell r_P^\ell}. \tag{4.14}$$

Note that $x^{c_\ell r_P^\ell}$ is just x^{r_P} composed ℓ times as well as composed with x^{c_ℓ} so can be represented by a rational neural network with $\ell + 1$ layers, each with one node. Therefore, all the $x^{c_\ell r_P^\ell}$ terms can be represented in rational networks that in total have size

$$\sum_{\ell=0}^{\lfloor \log_{r_P}(n) \rfloor} (\ell + 1) = \frac{1}{2}(\lfloor \log_{r_P}(n) \rfloor)^2 + \frac{3}{2}\lfloor \log_{r_P}(n) \rfloor + 1.$$

The function x^n can be formed by multiplying all the $x^{c_\ell r_P^\ell}$ terms together. Since $xy = (x^2 + y^2 - (x-y)^2)/2$, there is a rational network with one layer and three nodes that represents the multiplication operation. Therefore, multiplying all the terms together requires a rational network of size at most $3\lfloor \log_{r_P}(n) \rfloor$ (see Equation (4.14)). The result follows by noting that $x^2/2 + 9x/2 + 1 \leq 5x^2 + 1$ for $x \geq 1$. □

We can now prove Theorem 4.5 using the two previous propositions.

Proof of Theorem 4.5. The proof is based on the proof of [242, Theorem 1] and consists of replacing the piecewise linear functions and monomials arising in the local Taylor approximation of the function f by rational networks using the previous approximation results.

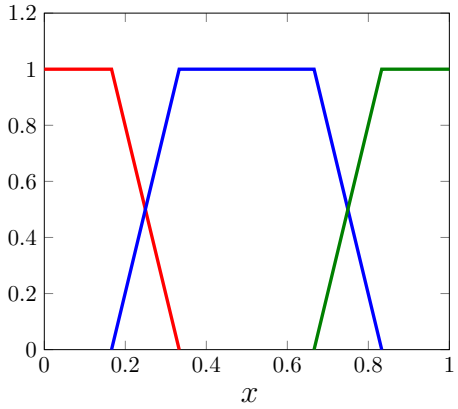


Figure 4.2: Partition of unity: ψ_0 (red), ψ_1 (blue), and ψ_2 (green), for $N = 2$.

Let $N \geq 1$ be an integer and consider a partition of unity of $(N+1)^d$ functions $\phi_{\mathbf{m}}$ on the domain $[0,1]^d$, *i.e.*,

$$\sum_{\mathbf{m}\in\{0,\ldots,N\}^d} \phi_{\mathbf{m}}(\mathbf{x}) = 1, \quad \phi_{\mathbf{m}}(\mathbf{x}) = \prod_{k=1}^{d} \psi_{m_k}(x_k), \quad \mathbf{x} = (x_1, \ldots, x_d),$$

where $\mathbf{m} = (m_1, \ldots, m_d)$, and ψ_{m_k} is given by

$$\psi_{m_k}(x) = \begin{cases} 1, & \text{if } \left|x_k - \frac{m_k}{N}\right| < \frac{1}{3N}, \\ 0, & \text{if } \left|x_k - \frac{m_k}{N}\right| > \frac{2}{3N}, \\ 2 - 3N\left|x_k - \frac{m_k}{N}\right|, & \text{otherwise.} \end{cases}$$

Examples of the functions ψ_{m_k} are shown in Figure 4.2 when $N = 2$. We now define a local Taylor approximation of f by

$$f_N(\mathbf{x}) = \sum_{\mathbf{m}\in\{0,\ldots,N\}^d} \phi_{\mathbf{m}}(\mathbf{x}) P_{\mathbf{m}}(\mathbf{x}),$$

where $P_{\mathbf{m}}$ denotes the degree $n-1$ Taylor polynomial of f at $\mathbf{x} = \mathbf{m}/N$. That is,

$$P_{\mathbf{m}}(\mathbf{x}) = \sum_{|\mathbf{n}|<n} \frac{D^{\mathbf{n}} f(\frac{\mathbf{m}}{N})}{\mathbf{n}!} \left(\mathbf{x} - \frac{\mathbf{m}}{N}\right)^{\mathbf{n}}, \tag{4.15}$$

where $|\mathbf{n}| = \sum_{k=1}^{d} n_k$, $\mathbf{n}! = \prod_{k=1}^{d} n_k!$, and $(\mathbf{x} - \mathbf{m}/N)^{\mathbf{n}} = \prod_{k=1}^{d} (x_k - m_k/N)^{n_k}$. Let $\mathbf{x} \in [0,1]^d$ and note that

$$\text{support}(\phi_{\mathbf{m}}) \subset \left\{ \mathbf{x} = (x_1, \ldots, x_d) : \left|x_k - \frac{m_k}{N}\right| < \frac{1}{N} \right\}, \quad \mathbf{m} \in \{0, \ldots, N\}^d.$$

Hence, the approximation error between f and its local Taylor approximation satisfies

$$|f(\mathbf{x}) - f_N(\mathbf{x})| = \left| \sum_{\mathbf{m}\in\{0,\dots,N\}^d} \phi_{\mathbf{m}}(f(\mathbf{x}) - P_{\mathbf{m}}(\mathbf{x})) \right| \leq \sum_{\mathbf{m}:\left|x_k - \frac{m_k}{N}\right| < \frac{1}{N}} |f(\mathbf{x}) - P_{\mathbf{m}}(\mathbf{x})|$$
$$\leq \frac{2^d d^n}{n!} \left(\frac{1}{N}\right)^n \max_{|\mathbf{n}|=n} \operatorname*{ess\,sup}_{\mathbf{x}\in[0,1]^d} |D^{\mathbf{n}} f(\mathbf{x})| \leq \frac{2^d d^n}{n!} \left(\frac{1}{N}\right)^n.$$

We now select (see [242, Theorem 1] for a similar idea)

$$N = \left\lceil \left(\frac{n!}{2^d d^n} \frac{\epsilon}{2} \right)^{-1/n} \right\rceil,$$

so that

$$\max_{\mathbf{x}\in[0,1]^d} |f(\mathbf{x}) - f_N(\mathbf{x})| \leq \epsilon/2. \tag{4.16}$$

We now approximate the function f_n by a rational network using Propositions 4.2 and 4.3. First, we write f_N as

$$f_N(\mathbf{x}) = \sum_{\mathbf{m}\in\{0,\dots,N\}^d} \sum_{|\mathbf{n}|<n} a_{\mathbf{m},\mathbf{n}} \phi_{\mathbf{m}}(\mathbf{x}) \left(\mathbf{x} - \frac{\mathbf{m}}{N}\right)^{\mathbf{n}}, \tag{4.17}$$

where $|a_{\mathbf{m},\mathbf{n}}| \leq 1$ and the monomials are uniformly bounded by 1 (see Equation (4.15)). Equation (4.17) consists of at most $d^n(N+1)^d$ terms of the form $\phi_{\mathbf{m}}(\mathbf{x})(\mathbf{x} - \mathbf{m}/N)^{\mathbf{n}}$. The monomial part $(\mathbf{x} - \mathbf{m}/N)^{\mathbf{n}}$ in Equation (4.17) is representable by a rational network of size $\mathcal{O}(d\log(n)^2)$ using Proposition 4.3, including the fact that the multiplication is a rational network with one layer and three nodes. Let $0 < \delta < 1$ be a small number, for each $m_k \in \{0,\dots,N\}$ the piecewise linear function ψ_{m_k} has a Lipschitz constant of $L = 3N$. Therefore, it can be approximated with a tolerance δ by a rational network $\tilde{\psi}_{m_k}$ of size $\mathcal{O}(\log(\log(N/\delta)))$ (see Proposition 4.2). We can assume $\|\tilde{\psi}_{m_k}\|_\infty = 1$ by increasing the size of the network by a constant. This yields the following approximation error between a term in Equation (4.17) and the rational network constructed using $\tilde{\psi}_{m_k}$:

$$\left| \phi_{\mathbf{m}}(\mathbf{x}) \left(\mathbf{x} - \frac{\mathbf{m}}{N}\right)^{\mathbf{n}} - \prod_{k=1}^d \tilde{\psi}_{m_k}(x_k) \left(\mathbf{x} - \frac{\mathbf{m}}{N}\right)^{\mathbf{n}} \right| \leq \left| \prod_{k=1}^d \psi_{m_k}(x_k) - \prod_{k=1}^d \tilde{\psi}_{m_k}(x_k) \right|$$
$$\leq \left|\psi_{m_1}(x_1) - \tilde{\psi}_{m_1}(x_1)\right| \left| \prod_{k=2}^d \psi_{m_k}(x_k) \right| + \left|\tilde{\psi}_{m_1}(x_1)\right| \left| \prod_{k=2}^d \psi_{m_k}(x_k) - \prod_{k=2}^d \tilde{\psi}_{m_k}(x_k) \right|$$
$$\leq \left|\psi_{m_1}(x_1) - \tilde{\psi}_{m_1}(x_1)\right| + \left| \prod_{k=2}^d \psi_{m_k}(x_k) - \prod_{k=2}^d \tilde{\psi}_{m_k}(x_k) \right|$$
$$\leq \delta + \left| \prod_{k=2}^d \psi_{m_k}(x_k) - \prod_{k=2}^d \tilde{\psi}_{m_k}(x_k) \right| \leq d\delta.$$

Here, the final inequality is derived by repeating the argument of the previous inequalities for $x_2, \ldots, x_d$. If we denote by $\mathcal{N}_{\text{Rational}}$ the rational network approximation to f_N constructed above, then, for all $\mathbf{x} \in [0,1]^d$, we have

$$\begin{aligned}|f_N(\mathbf{x}) - \mathcal{N}_{\text{Rational}}(\mathbf{x})| &\leq \sum_{\mathbf{m}\in\{0,\ldots,N\}^d} \sum_{|\mathbf{n}|<n} |a_{\mathbf{m},\mathbf{n}}| \left| \phi_{\mathbf{m}}(\mathbf{x}) \Big(\mathbf{x} - \frac{\mathbf{m}}{N}\Big)^{\mathbf{n}} - \prod_{k=1}^{d} \tilde{\psi}_{m_k}(x_k) \Big(\mathbf{x} - \frac{\mathbf{m}}{N}\Big)^{\mathbf{n}} \right| \\ &\leq 2^d d^{n+1} \delta.\end{aligned}$$

Therefore, we select $\delta = \epsilon/(2^{d+1} d^{n+1})$ so that $\max_{\mathbf{x}\in[0,1]^d} |f_N(\mathbf{x}) - \tilde{f}_N(\mathbf{x})| \leq \epsilon/2$. Then, by Equation (4.16), we have

$$\max_{\mathbf{x}\in[0,1]^d} |f(\mathbf{x}) - \mathcal{N}_{\text{Rational}}(\mathbf{x})| \leq \frac{\epsilon}{2} + \frac{\epsilon}{2} \leq \epsilon.$$

The statement of the theorem follows as the rational network $\mathcal{N}_{\text{Rational}}$ has size at most

$$\mathcal{O}(d^n (N+1)^d \log(\log(N/\delta))) = \mathcal{O}(\epsilon^{-d/n} \log(\log(1/\epsilon^{1+1/n}))) = \mathcal{O}(\epsilon^{-d/n} \log(\log(1/\epsilon))).$$

□

A theorem proved by DeVore et al. [52] gives a lower bound of $\Omega(\epsilon^{-d/n})$ on the number of parameters needed by a neural network to express any function in $F_{d,n}$ with an error ϵ, under the assumption that the weights are chosen continuously. Comparing $\mathcal{O}(\epsilon^{-d/n} \log(\log(1/\epsilon)))$ and $\mathcal{O}(\epsilon^{-d/n} \log(1/\epsilon))$, we find that rational neural networks require exponentially fewer nodes than ReLU networks with respect to the optimal bound of $\Omega(\epsilon^{-d/n})$ to approximate functions in $F_{d,n}$.

4.3 Experiments using rational neural networks

In this section, we consider neural networks with trainable rational activation functions of type $(3, 2)$. We select the type $(3, 2)$ based on empirical performance; roughly, a low-degree (but higher than 1) rational function is ideal for generating high-degree rational functions by composition, with a small number of parameters. The rational activation units can be easily implemented in the open-source TensorFlow library [1] by using the `polyval` and `divide` commands for function evaluations. The coefficients of the numerators and denominators of the rational activation functions are trainable parameters, determined at the same time as the weights and biases of the neural network by backpropagation and a gradient descent optimization algorithm.

One crucial question is the initialization of the coefficients of the rational activation functions [39, 153]. A badly initialized rational function might contain poles on the real axis, leading to exploding values, or converge to a local minimum in the optimization process. Our experiments, supported by the empirical results of Molina et al. [153], show that initializing each rational function with the best rational approximation to the ReLU function (as described in Lemma 4.2) produces good performance. The underlying idea is to initialize rational networks near a network with ReLU activation functions, widely used for deep learning. Then, the adaptivity of the rational functions allows for further improvements during the training phase. We represent the initial rational function used in our experiments in Figure 4.1(right). The coefficients of this function are obtained by using the `minimax` command, available in the Chebfun software [56, 67] for numerically computing rational approximations, and are given in Table 4.1.

Table 4.1: Initialization coefficients of the rational activation functions.

a_0	a_1	a_2	a_3	b_0	b_1	b_2
1.1915	1.5957	0.5000	0.0218	2.3830	0.0000	1.0000

In the following experiments, we use a single rational activation function of type $(3, 2)$ at each layer, instead of different functions at each node to reduce the number of trainable parameters and the computational training expense. This adds 7 degrees of freedom per layer.

4.3.1 Approximation of functions

Raissi, Perdikaris, and Karniadakis [180, 184] introduce a framework called *deep hidden physics models* for discovering nonlinear partial differential equations (PDEs) from observations. This technique requires to solving the following interpolation problem: given the observation data $(u_i)_{1\leq i\leq N}$ at the spatio-temporal points $(x_i, t_i)_{1\leq i\leq N}$, find a neural network $\mathcal{N}$ (called the identification network), that minimizes the loss function

$$\mathcal{L} = \frac{1}{N}\sum_{i=1}^{N} |\mathcal{N}(x_i, t_i) - u_i|^2. \tag{4.18}$$

This technique has successfully discovered hidden models in fluid mechanics [185], solid mechanics [85], and nonlinear PDEs such as the Korteweg–de Vries (KdV) equation [184]. Raissi et al. use an identification network, consisting of 4 layers and 50

nodes per layer, to interpolate samples from a solution to the KdV equation. Moreover, they observe that networks based on smooth activation functions, such as the hyperbolic tangent ($\tanh(x)$) or the sinusoid ($\sin(x)$), outperform ReLU neural networks [180, 184]. However, the performance of these smooth activation functions highly depends on the application.

Moreover, these functions might not be adapted to approximate non-smooth or highly oscillatory solutions. Recently, Jagtap, Kawaguchi, and Karniadakis [96] proposed and analyzed different adaptive activation functions to approximate smooth and discontinuous functions with physics-informed neural networks. More specifically, they use an adaptive version of classical activation functions such as sigmoid, hyperbolic tangent, ReLU, and Leaky ReLU. The choice of these trainable activation functions introduces another parameter in the design of the neural network architecture, which may not be ideal for use for a black-box data-driven PDE solver.

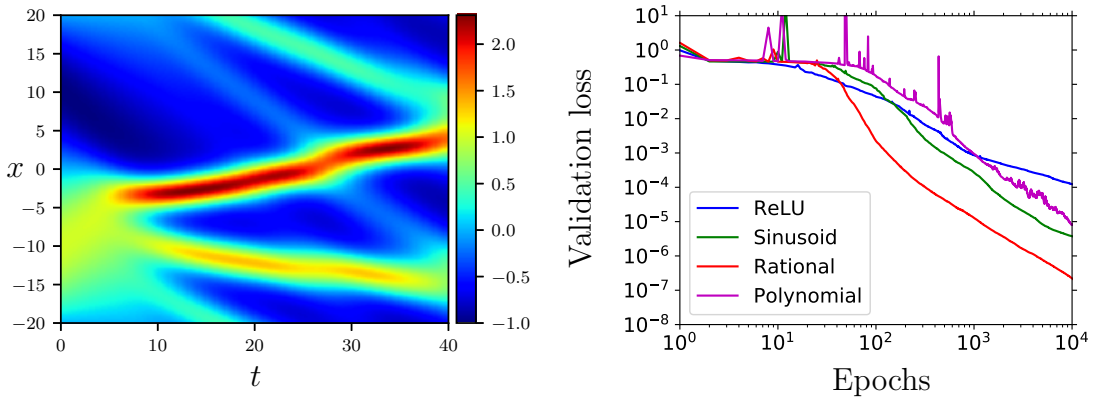


Figure 4.3: Solution to the KdV equation used as training data (left) and validation loss of a ReLU (blue), sinusoid (green), rational (red), and polynomial (purple) neural networks with respect to the number of optimization steps (right).

We illustrate that rational neural networks can address the issues mentioned above due to their adaptivity and approximation power (see Section 4.2). Similarly to Raissi [180], we use a solution u to the KdV equation:

$$u_t = -uu_x - u_{xxx}, \quad u(x,0) = -\sin(\pi x/20),$$

as training data for the identification network (see the left panel of Figure 4.3). We use the TensorFlow implementation[2] of the deep hidden physics model framework to build and train the identifier network $\mathcal{N}$ that approximates a solution u to the KdV equation. The true solution is computed on the domain $(x,t) \in [-20,20] \times [0,40]$ by

[2]We adapt the code that is publicly available [181].

Raissi [180] using the Chebfun package [56] with a spectral Fourier discretization of 512 and a time-step of $\Delta t = 10^{-4}$. Moreover, the solution is stored after every 2000 time steps, giving a testing data set of approximately 10^5 spatio-temporal points in $[-20, 20] \times [0, 40]$. We then constituted the training and validation sets (of 10^4 points each) by randomly subsampling the solution at 2×10^4 points in $[-20, 20] \times [0, 40]$.

In a similar manner to [180], we use a fully connected identification network to approximate u with 4 hidden layers with 50 nodes per layer. The network is trained using the L-BFGS optimization algorithm with 10,000 iterations. We train and compare four networks with the following activation functions: ReLU, sinusoid, trainable rational functions of type $(3, 2)$, and trainable polynomials of degree 3. Furthermore, the rational activation functions are initialized to be the best approximation to the ReLU function, using the initial coefficients reported in Table 4.1.

The mean squared error (MSE) of the neural networks on the validation set throughout the training phase is reported in the right panel of Figure 4.3. We observe that the rational neural network outperforms the sinusoid network, despite having the same asymptotic convergence rate. The network with polynomial activation functions (chosen to be of degree 3 in this example) is harder to train than the rational network, as shown by the non-smooth validation loss (see the right panel of Figure 4.3). We highlight that rational neural networks are never much bigger in terms of trainable parameters than ReLU networks since the increase is only linear with respect to the number of layers. Here, the ReLU network has 8000 parameters (consisting of weights and biases), while the rational network has $8000 + 7 \times \#\text{layers} = 8035$. The ReLU, sinusoid, rational, and polynomial networks achieve the following mean square errors after 10^4 epochs:

$$\text{MSE}(u_{\text{ReLU}}) = 1.9 \times 10^{-4}, \qquad \text{MSE}(u_{\text{Sinusoid}}) = 3.3 \times 10^{-6},$$
$$\text{MSE}(u_{\text{Rational}}) = 1.2 \times 10^{-7}, \qquad \text{MSE}(u_{\text{Polynomial}}) = 3.6 \times 10^{-5}.$$

The rational neural network is approximately five times more accurate than the sinusoid network used by Raissi and twenty times more accurate than the ReLU network. The absolute approximation errors between the different neural networks and the exact solution to the KdV equation is illustrated in Figure 4.4. We find that the approximation errors made by the ReLU network are not uniformly distributed in space and time and located in specific regions, indicating that a network with non-smooth activation functions is not appropriate to resolve smooth solutions to PDEs.

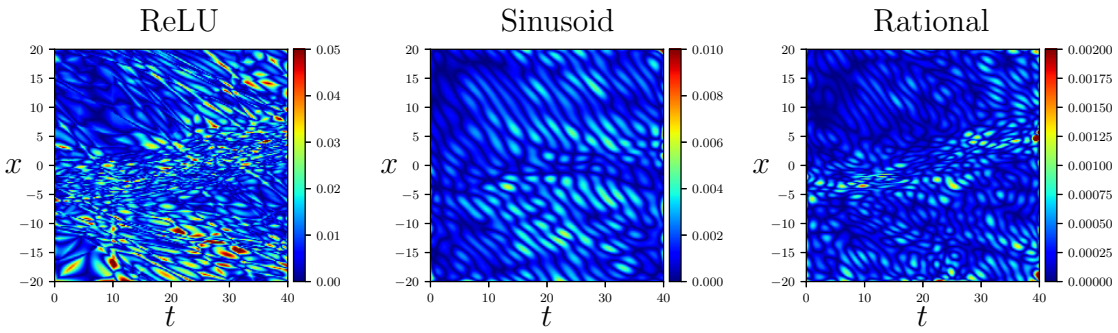


Figure 4.4: Approximation errors of the neural networks with ReLU, sinusoid, and rational activation layers. Note the different scales of the errors.

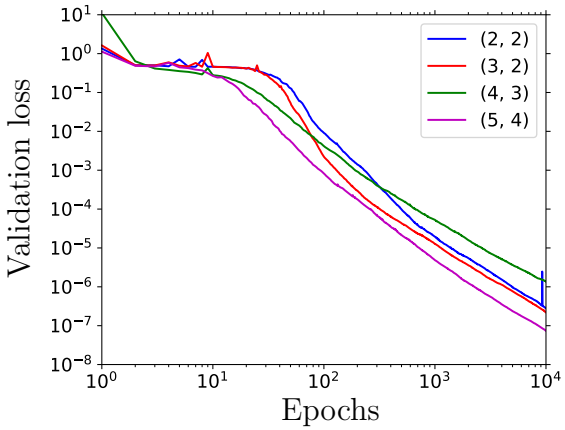


Figure 4.5: Validation loss of rational networks of types $(2, 2)$, $(3, 2)$, $(4, 3)$, and $(5, 4)$ with respect to the number of epochs.

Finally, in Figure 4.5, we compare rational neural networks with different degree activation functions (each initialized to approximate the ReLU function using the MATLAB code `initial_rational_coeffs.m` available at [29]) and find that they all performed better than ReLU networks. While a type $(3, 2)$ rational offers a good trade-off between the number of parameters and quality of approximation according to the theoretical results presented in Section 4.2, the type of rational function might well depend on the application considered.

4.3.2 Generative adversarial networks

Generative adversarial networks are used to generate synthetic examples from an existing dataset [76]. They consist of two networks: a generator to produce synthetic samples and a discriminator to evaluate the samples of the generator with the training dataset. Radford et al. [179] describe deep convolutional generative adversarial networks (DCGANs) to build good image representations using convolutional architectures. They evaluate their model on the MNIST and ImageNet image datasets [51, 120].

This section highlights the simplicity of using rational activation functions in existing neural network architectures by training an Auxiliary Classifier GAN (ACGAN) [165] on the MNIST dataset. In particular, the neural network, referred to as the ReLU network in this section, consists of convolutional generator and discriminator networks with ReLU and Leaky ReLU [140] activation units (respectively) and is used as a reference GAN. We adapt the Keras example in [43] to train an Auxiliary Classifier GAN with rational activation functions on the MNIST. The hyperparameters used for the GAN experiment are given in Table 4.2. Moreover, the GAN is trained on 20 epochs with a batch size of 100 by Adam's optimization algorithm [108] and the following parameters: $\alpha = 0.0002$ and $\beta_1 = 0.5$, as suggested by [179].

As in the experiment described in Section 4.3.1, we replace the activation units of the generative and discriminator networks by a rational function with trainable coefficients (see Figure 4.1). We initialize the activation functions in the training phase with the best rational function that approximates the ReLU function on $[-1, 1]$.

We show images of digits from the first five classes generated by a ReLU and rational GANs at different epochs of the training in Figure 4.6 (the samples are generated randomly and are not manually selected). We observe that a rational network can generate realistic images with a broader range of features than the ReLU

Table 4.2: Hyper-parameters of the GAN experiment, BN denotes the presence of a Batch normalization layer. The Generator and Discriminator networks are trained with ReLU and rational activation functions, initialized with the coefficients reported in Table 4.1. Transposed convolution layers and rational activation functions are respectively abbreviated as "Transp. Conv." and "Rat.".

Operation	Kernel	Strides	Features	BN	Dropout	Activation
Generator						
Linear	N/A	N/A	3456	✗	0.0	ReLU / Rat.
Transp. Conv.	5×5	1×1	192	✓	0.0	ReLU / Rat.
Transp. Conv.	5×5	2×2	96	✓	0.0	ReLU / Rat.
Transp. Conv.	5×5	2×2	1	✗	0.0	Tanh
Discriminator						
Convolution	3×3	2×2	32	✗	0.3	Leaky ReLU / Rat.
Convolution	3×3	1×1	64	✗	0.3	Leaky ReLU / Rat.
Convolution	3×3	2×2	128	✗	0.3	Leaky ReLU / Rat.
Convolution	3×3	1×1	256	✗	0.3	Leaky ReLU / Rat.
Linear	N/A	N/A	11	✗	0.0	Soft-Sigmoid

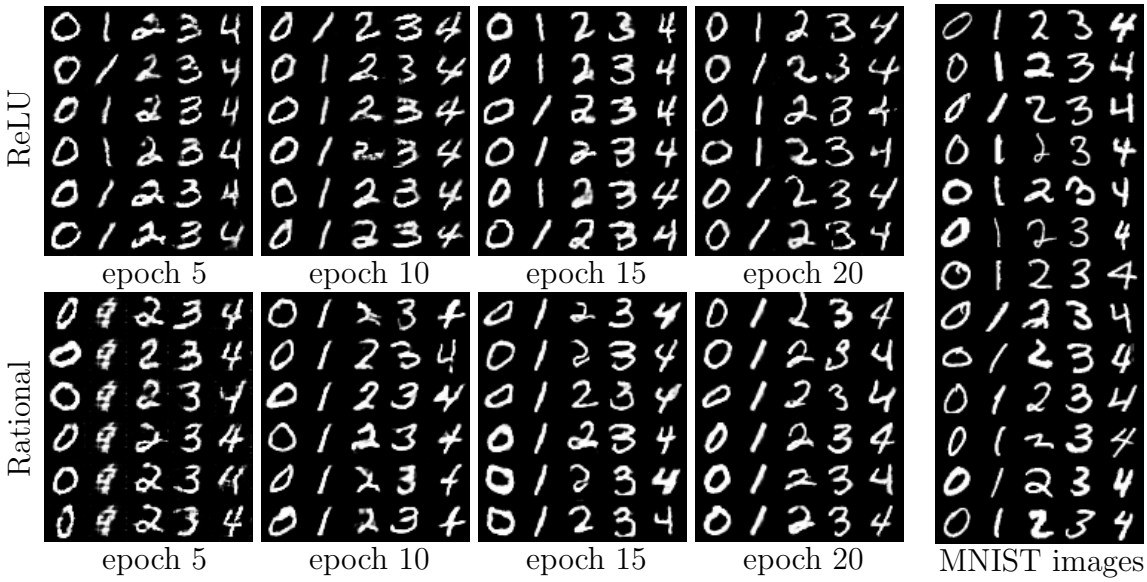
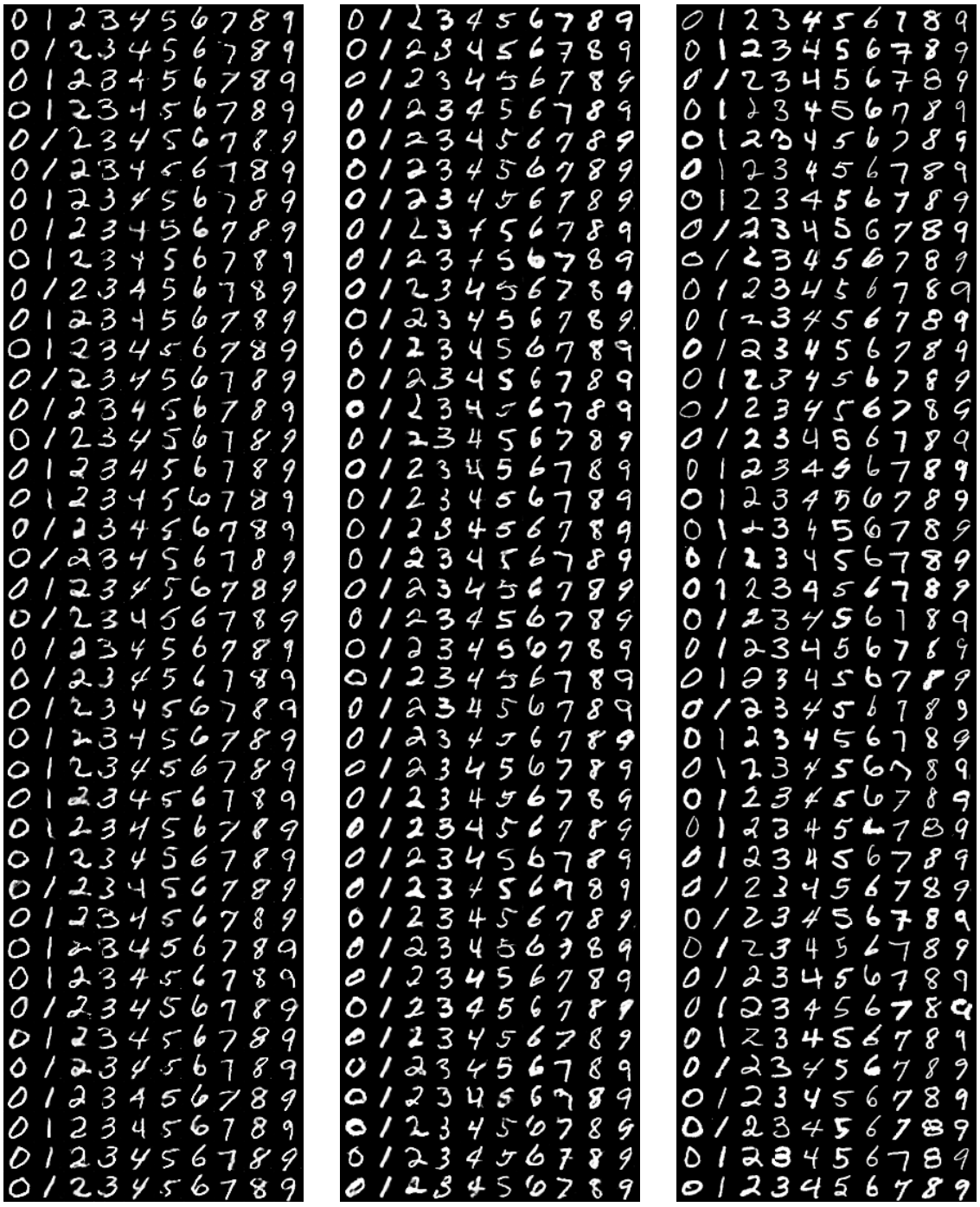


Figure 4.6: Digits generated by a ReLU (top) and rational (bottom) auxiliary classifier generative adversarial network. The right panel contains samples from the first five classes of the MNIST dataset for comparison.

network, as illustrated by the presence of bold numbers at the epoch 20 in the bottom panel of Figure 4.6.

We report in Figure 4.7 samples of the 10 classes present in the MNIST dataset (right) and images generated at the 20th epoch by the GAN with ReLU/Leaky ReLU units (left) and rational activation functions (middle). We observe that the digits one generated by the rational network are identical, suggesting that the rational GAN suffers from mode collapse. It should be noted that generative adversarial networks are notoriously tricky to train [75]. The hyper-parameters of the reference model are intensively tuned for a piecewise linear activation function (as shown by the use of Leaky ReLU in the discriminator network). Moreover, many stabilization methods have been proposed to resolve the mode collapse and non-convergence issues in training, such as Wasserstein GAN [10], Unrolled Generative Adversarial Networks [149], and batch normalization [95]. These techniques could be explored and combined with rational networks to address the mode collapse issue observed in this experiment.



ReLU Rational MNIST images

Figure 4.7: Forty images generated by a ReLU network and a rational network after 20 epochs, together with real images from the MNIST dataset.

Chapter 5

Data-driven discovery of Green's functions with deep learning*

Deep learning (DL) holds promise as a scientific tool for discovering elusive patterns within the natural and technological world [75, 118]. These patterns hint at undiscovered partial differential equations (PDEs) that describe governing phenomena within biology and physics. From sparse and noisy laboratory observations, we aim to learn mechanistic laws of nature [35, 102]. Recently, scientific computing and machine learning have successfully converged on PDE discovery [36, 195, 197, 245], PDE learning [65, 71, 126, 135, 180, 185], and symbolic regression [201, 224] as promising means for applying machine learning to scientific investigations. These methods attempt to discover the coefficients of a PDE model or learn the operator that maps excitations to system responses. The recent DL techniques addressing the latter problem are based on approximating the solution operator associated with a PDE by a neural network (NN) [65, 71, 126, 135, 180]. While excellent for solving PDEs, we consider them as "black box" and focus here on a data-driven strategy that improves human understanding of the governing PDE model.

We then offer a radically different, alternative approach that is backed by theory [32] and infuse an interpretation in the model by learning well-understood mathematical objects that imply underlying physical laws. We devise a DL method, employed for learning the Green's functions [207] associated with unknown governing linear PDEs, and train the neural networks by collecting physical system responses from random excitation functions drawn from a Gaussian process (GP). The empirically derived Green's functions relate the system's response (or PDE solution) to a

*This chapter is based on a paper with Christopher Earls and Alex Townsend [27], published in Scientific Reports. Earls and Townsend had an advisory role; I designed the deep learning method, performed the numerical experiments, and was the lead author in writing the paper.

forcing term, and can then be used as a fast reduced-order PDE solver. The existing graph kernel network [126] and DeepGreen [71] techniques also aim to learn solution operators of PDEs based on Green's functions. While they show competitive performance in predicting the solution of the PDE for new forcing functions, the errors between the exact and learned Green's functions are relatively large, which makes the extraction of qualitative and quantitative features of the physical system challenging.

Our secondary objective is to study the discovered Green's functions for clues regarding the physical properties of the observed systems. Our approach relies on the rational neural networks introduced in the previous chapter, which have higher approximation power than standard networks and carry human-understandable features of the PDE, such as shock and singularity locations, as we shall see later.

In this chapter, we use techniques from deep learning to discover the Green's function of linear differential equations $\mathcal{L}u = f$ from input-output pairs (f, u), as opposed to directly learning $\mathcal{L}$, or model parameters. In this sense, our approach is agnostic to the forward PDE model, but nonetheless offers insights into its physical properties. There are several advantages to learning the Green's function. First, once the Green's function is learned by a neural network, it is possible to compute the solution, u, for a new forcing term, f, by evaluating an integral (see Equation (5.2)); which is more efficient than training a new NN. Second, the Green's function associated with $\mathcal{L}$ contains information about the operator, $\mathcal{L}$, and the type of boundary constraints that are imposed; which helps uncover mechanistic understanding from experimental data. Finally, as discussed in Chapter 2, it is easier to train NNs to approximate Green's functions, which are square-integrable functions under sufficient regularity conditions [53, 80, 207], than trying to approximate the action of the linear differential operator, $\mathcal{L}$, which is not bounded [113]. Also, any prior mathematical and physical knowledge of the operator, $\mathcal{L}$, can be exploited in the design of the NN architecture, which could enforce a particular structure such as symmetry of the Green's function.

5.1 Learning Green's functions

We consider linear differential operators, $\mathcal{L}$, defined on a bounded domain $\Omega \subset \mathbb{R}^d$, where $d \in \{1, 2, 3\}$ denotes the spatial dimension. The aim of our method is to discover properties of the operator, $\mathcal{L}$, using N input-output pairs $\{(f_j, u_j)\}_{j=1}^N$, consisting of forcing functions, $f_j : \Omega \to \mathbb{R}$, and system responses, $u_j : \Omega \to \mathbb{R}$, which are

solutions to the following equation:

$$\mathcal{L}u_j = f_j, \quad \mathcal{D}(u_j, \Omega) = g, \tag{5.1}$$

where $\mathcal{D}$ is a linear operator acting on the solutions, u, and the domain, Ω; with g being the constraint. We assume that the forcing terms have sufficient regularity, and that the operator, $\mathcal{D}$, is a constraint so that Equation (5.1) has a unique solution [207]. An example of constraint is the imposition of homogeneous Dirichlet boundary conditions on the solutions: $\mathcal{D}(u_j, \Omega) \coloneqq u_j|_{\partial\Omega} = 0$. Note that boundary conditions, integral conditions, jump conditions, or non-standard constraints, are all possible (see Section 5.4.1).

5.1.1 Definitions

A Green's function [9, 64, 157, 207] of the operator, $\mathcal{L}$, is defined as the solution to the following equation:

$$\mathcal{L}G(x, y) = \delta(y - x), \quad x, y \in \Omega,$$

where $\mathcal{L}$ is acting on the function $x \mapsto G(x, y)$ for fixed $y \in \Omega$, and $\delta(\cdot)$ denotes the Dirac delta function. The Green's function is well-defined and unique under mild conditions on $\mathcal{L}$, and suitable solution constraints imposed via an operator, $\mathcal{D}$ (see Equation (5.1)) [207]. Moreover, if (f, u) is an input-output pair, satisfying Equation (5.1) with $g = 0$, then

$$u(x) = \int_\Omega G(x, y) f(y) \, \mathrm{d}y, \quad x \in \Omega.$$

Therefore, the Green's function associated with $\mathcal{L}$ can be thought of as the right inverse of $\mathcal{L}$.

Let u_{hom} be the homogeneous solution to (5.1), so that

$$\mathcal{L}u_{\text{hom}} = 0, \quad \mathcal{D}(u_{\text{hom}}, \Omega) = g.$$

Using superposition, we can construct solutions, u_j, to Equation (5.1) as $u_j = \tilde{u}_j + u_{\text{hom}}$, where $\tilde{u}_j$ satisfies

$$\mathcal{L}\tilde{u}_j = f_j, \quad \mathcal{D}(\tilde{u}_j, \Omega) = 0.$$

Then, the relation between the system's response, u_j, and the forcing term, f_j, can be expressed via the Green's function as

$$u_j(x) = \int_\Omega G(x, y) f_j(y) \, \mathrm{d}y + u_{\text{hom}}(x), \quad x \in \Omega.$$

In this chapter, we focus on learning Green's functions and homogeneous solutions from a fixed boundary constraint g but one could also approximate a second Green's function associated with u_{hom} from multiple boundary constraints. Therefore, we train two NNs: $\mathcal{N}_G : \Omega \times \Omega \to \mathbb{R} \cup \{\pm\infty\}$ and $\mathcal{N}_{\text{hom}} : \Omega \to \mathbb{R}$, to learn the Green's function, and also the homogeneous solution associated with $\mathcal{L}$ and the constraint operator $\mathcal{D}$. Note that this procedure allows us to discover boundary conditions, or constraints, directly from the input-output data without imposing it in the loss function (which often results in training instabilities [236]).

5.1.2 Theoretical justification

Our approach for learning Green's functions associated with linear differential operators has a theoretically rigorous underpinning. Indeed, we showed in Chapter 2 that uniformly elliptic operators in three dimensions have an intrinsic *learning rate*, which characterizes the number of training pairs needed to construct an ϵ-approximation in the L^2-norm of the Green's function, G, with high probability, for $0 < \epsilon < 1$. The number of training pairs depends on the quality of the covariance kernel used to generate the random forcing terms, $\{f_j\}_{j=1}^N$. Our choice of covariance kernel (Section 5.2.1) is motivated by the GP quality measure (cf. Section 2.1.4), to ensure that our set of training forcing terms is sufficiently diverse to capture the action of the solution operator, $f \mapsto u(x) = \int_\Omega G(x,y)f(y)\,\mathrm{d}y$, on a diverse set of functions.

Similarly, the choice of rational NNs to approximate the Green's function, and the homogeneous solution, is justified by the higher approximation power of these networks over ReLU as observed in Chapter 4. Other adaptive activation functions have been proposed for learning or solving PDEs with NNs [96], but they are only motivated by empirical observations. Both theory and experiments support rational NNs for regression problems. The number of trainable parameters, consisting of weight matrices, bias vectors, and rational coefficients, needed by a rational NN to approximate smooth functions within $0 < \epsilon < 1$, can be completely characterized [30]. This motivates our choice of NN architecture for learning Green's functions.

5.2 Deep learning method

Our DL approach (see Figure 5.1) begins with excitations (or forcing terms), $\{f_j\}_{j=1}^N$, sampled from a Gaussian process having a carefully designed covariance kernel, and corresponding system responses, $\{u_j\}_{j=1}^N$ (see Chapter 3). It is postulated that there is an unknown linearized governing PDE so that $\mathcal{L}u_j = f_j$. The selection of random

forcing terms is theoretically justified by Chapter 2 and enables us to learn the dominant eigenmodes of the solution operator, using only a small number, N, of training pairs. The Green's function, G, and homogeneous solution, u_{hom}, which encodes the boundary conditions associated with the PDE, satisfy

$$u_j(x) = \int_\Omega G(x,y) f_j(y)\,\mathrm{d}y + u_{\text{hom}}(x), \quad x \in \Omega, \tag{5.2}$$

and are approximated by two rational neural networks: $\mathcal{N}_G$ and $\mathcal{N}_{\text{hom}}$.

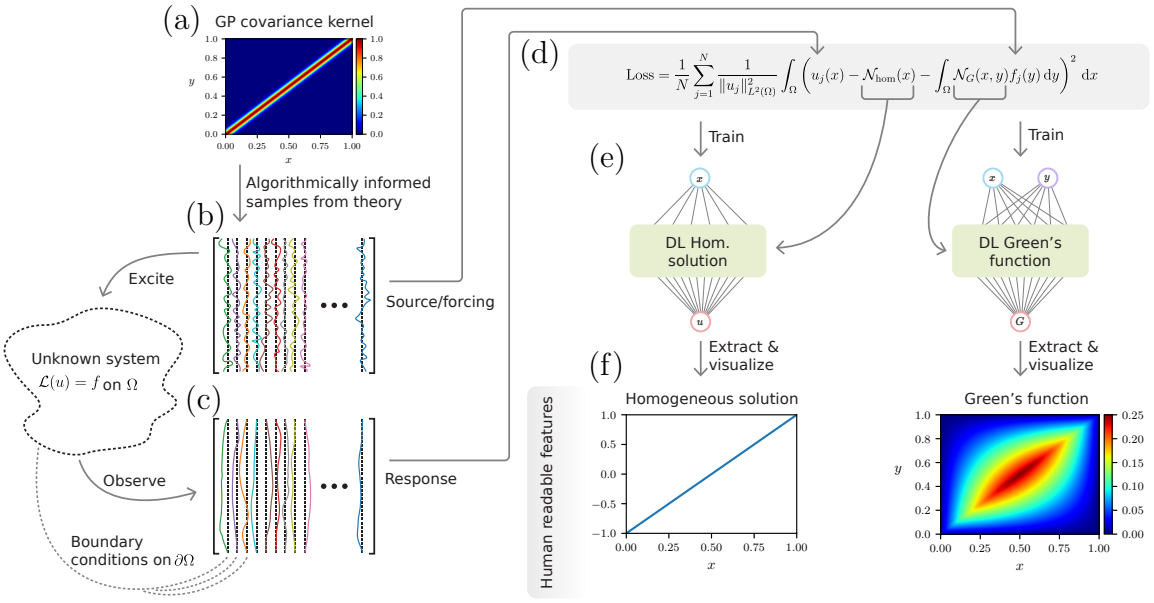


Figure 5.1: Schematic of our DL method for learning Green's functions from input-output pairs. (a) The covariance kernel of the Gaussian process, which is used to generate excitations. (b) The system's response to each excitation is computed and recorded (c). (d) A loss function is minimized to train rational NNs (e). (f) The learned Green's function and homogeneous solution are visualized by sampling the NNs.

The parameters of the NNs representing the Green's function and homogeneous solution are simultaneously learned through minimization of the loss function displayed in Figure 5.1(d). We discretize the integrals in the loss function at the specified measurement locations $\{x_i\}_{i=1}^{N_u}$, within the domain, Ω, and forcing term sample points, $\{y_i\}_{i=1}^{N_f}$, respectively, using a quadrature rule.

In this section, we detail the deep learning method used to learn Green's functions. Our DL technique is data-driven and requires minimal by-hand parameter tuning. In

fact, all the numerical examples described in this chapter are performed using a single rational NN architecture, initialization procedure, and optimization algorithm[1].

5.2.1 Generating the training data

We create a training dataset, consisting of input-output functions, $\{(f_j\, u_j)\}$ for $1 \leq j \leq N$, in three steps: (1) Generating the forcing terms by sampling random functions from a Gaussian process, (2) Solving Equation (5.1) for the generated forcing terms, and (3) Sampling the forcing terms, f_j, at the points $\{y_1, \ldots, y_{N_f}\} \subset \Omega$ and the system's responses, u_j, at $\{x_1, \ldots, x_{N_u}\} \subset \Omega$. Here, N_f and N_u are the forcing and solution discretization sizes, respectively. We recommend that all the forcing terms are sampled on the same grid and similarly for the system's responses. This minimizes the number of evaluations of $\mathcal{N}_G$ during the training phase and reduces the computational and memory costs of training.

The spatial locations of points $\{y_i\}$ and the forcing discretization size, N_f, are chosen arbitrarily to train the NNs as the forcing terms are assumed to be known over Ω. In practice, the number, N_u, and location of the measurement points, $\{x_i\}$, are imposed by the nature of the experiment, or simulation, performed to measure the system's response. When Ω is an interval, we always select $N_f = 200$, $N_u = 100$, and equally-spaced sampled points for the forcing and response functions.

Unless otherwise stated, the training data comprises $N = 100$ forcing and solution pairs, where the forcing terms are drawn at random from a Gaussian process, $\mathcal{GP}(0, K_{\text{SE}})$, where K_{SE} is the squared-exponential covariance kernel [187] defined as

$$K_{\text{SE}}(x, y) = \exp\left(-\frac{|x-y|^2}{2\ell^2}\right), \quad x, y \in \Omega. \tag{5.3}$$

As discussed in Section 3.3, the parameter $\ell > 0$ in Equation (5.3) is called the length-scale parameter, and characterizes the correlation between the values of $f \sim \mathcal{GP}(0, K_{\text{SE}})$ at x and y for $x, y \in \Omega$. A small parameter, ℓ, yields highly oscillatory random functions, f, and determines the ability of the GP to generate a diverse set of training functions. This last property is crucial for capturing different modes within the operator, $\mathcal{L}$, and for learning the associated Green's function accurately [32]. Other possible choices of covariance kernels include the periodic kernel [187]:

$$K_{\text{Per}}(x, y) = \exp\left(-\frac{2\sin^2(\pi|x-y|)}{\ell^2}\right), \quad x, y \in \Omega,$$

[1]All data and codes used in this chapter are publicly available on the GitHub and Zenodo repositories at `https://github.com/NBoulle/greenlearning/` [26] to reproduce the numerical experiments and figures. A software package, including additional examples and documentation, is also available at `https://greenlearning.readthedocs.io/`.

which is used to sample periodic random functions for problems with periodic boundary conditions (Figure 5.8(b)). Another possibility is a kernel from the Matérn family [187] or the Jacobi kernel introduced in Section 3.3.3.

When Ω is an interval $[a, b]$, we introduce a normalized length-scale parameter $\lambda = \ell/(b-a)$, so that the method described does not depend on the length of the interval. In addition, we choose $\lambda = 0.03$, so that the length-scale, ℓ, is larger than the forcing spatial discretization size, which allows us to adequately resolve the functions sampled from the GP with the discretization. More precisely, we make sure that $\ell \geq (b-a)/N_f$ so that $1/N_f \leq \lambda$. In Figure 5.2, we display the squared-exponential covariance kernel on the domain $\Omega = [-1, 1]$, along with ten random functions sampled from $\mathcal{GP}(0, K_{\text{SE}})$.

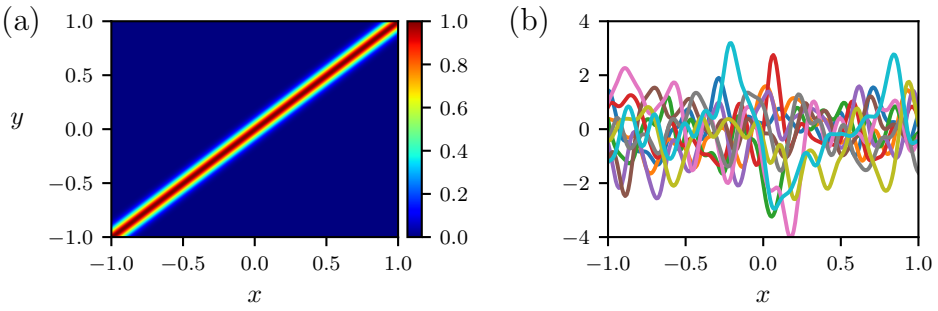


Figure 5.2: Random forcing terms. (a) Squared exponential covariance kernel K_{SE} on $[-1, 1]^2$ with normalized length-scale $\lambda = 0.03$ (b) 10 functions sampled from the Gaussian process $\mathcal{GP}(0, K_{\text{SE}})$.

When (5.1) is a boundary-value problem, we generate training pairs by solving the PDE with a spectral method [219] using the Chebfun software system [56], written in MATLAB, and using a tolerance of 5×10^{-13}. We also solve the homogeneous problem with zero-forcing, to compare the learned and exact homogeneous solutions. The exact homogeneous solution is not included in the training dataset. When the homogeneous solution is zero, the solutions, $\{u_j\}_{j=1}^N$, and forcing terms, $\{f_j\}_{j=1}^N$, are rescaled, so that $\max_{1 \leq j \leq N} \|u_j\|_{L^\infty(\Omega)} = 1$. By doing this, we facilitate the training of the NNs by avoiding disproportionately small-scale or large-scale data. In the presence of real data, with no known homogeneous solution, one could instead normalize the output of the NNs, $\mathcal{N}_G$ and $\mathcal{N}_{\text{hom}}$, to facilitate the training procedure.

5.2.2 Rational neural networks

As introduced in Chapter 4, rational NNs consist of NNs with adaptive rational activation functions $x \mapsto \sigma(x) = p(x)/q(x)$, where p and q are two polynomials, whose

coefficients are trained at the same time as the other parameters of the networks, such as the weights and biases. These coefficients are shared between all the neurons in a given layer but generally differ between the network's layers. This type of network was proven to have better approximation power than standard Rectified Linear Unit (ReLU) networks [73, 242], which means that they can approximate smooth functions more accurately with fewer layers and network parameters (see Section 4.2). It is also observed in Section 4.3 that rational NNs require fewer optimization steps in practice and therefore can be more efficient to train than other activation functions.

The NNs, $\mathcal{N}_G$ and $\mathcal{N}_{\text{hom}}$, which approximate the Green's function and homogeneous solution associated with Equation (5.1), respectively, are chosen to be rational NNs with 4 hidden layers and 50 neurons in each layer. We choose the polynomials, p and q, within the activation functions to be of degree 3 and 2, respectively, and initialize the coefficients of all the rational activation functions so that they are the best $(3, 2)$ rational approximant to a ReLU (see Section 4.3 for details). The motivation is that the flexibility of the rational functions brings extra benefit in the training and accuracy over the ReLU activation function. We highlight that the increase in the number of trainable parameters, due to the adaptive rational activation functions, is only linear with respect to the number of layers and negligible compared to the total number of parameters in the network as:

$$\text{number of rational coefficients} = 7 \times \text{number of hidden layers} = 28.$$

The weight matrices of the NNs are initialized using Glorot normal initializer [72], while the biases are initialized to zero.

Another advantage of rational NNs is the potential presence of poles, *i.e.*, zeros of the polynomial q. While the initialization of the activation functions avoids training issues due to potential spurious poles, the poles can be exploited to learn physical features of the differential operator (see Section 5.4.5). Therefore, the architecture of the NNs also supports the aim of a human-understandable approach for learning PDEs. In higher dimensions, such as $d = 2$ or $d = 3$, the Green's function is not necessarily bounded along the diagonal, *i.e.*, $\{(x, x),\, x \in \Omega\}$; thus making the poles of the rational NNs crucial.

Finally, we emphasize that the enhanced approximation properties of rational NNs make them ideal for learning Green's functions and, more generally, approximating functions within regression problems. These networks may also be of benefit

to other approaches for solving and learning PDEs with DL techniques, such as DeepGreen [71], Neural operator [126], Fourier neural operator [127], DeepONet [135], and PINNs [184].

5.2.3 Loss function

The NNs, $\mathcal{N}_G$ and $\mathcal{N}_{\text{hom}}$, are trained by minimizing a mean square relative error (in the L^2-norm) regression loss, defined as:

$$\text{Loss} = \frac{1}{N}\sum_{j=1}^{N} \frac{1}{\|u_j\|^2_{L^2(\Omega)}} \int_\Omega \left(u_j(x) - \mathcal{N}_{\text{hom}}(x) - \int_\Omega \mathcal{N}_G(x,y) f_j(y)\,\mathrm{d}y \right)^2 \mathrm{d}x. \quad (5.4)$$

Unless otherwise stated, the integrals in Equation (5.4) are discretized by a trapezoidal rule [210] using training data values that coincide with the forcing discretization grid, $\{y_i\}_{i=1}^{N_f}$, and measurement points, $\{x_i\}_{i=1}^{N_u}$. As an example, for $1 \leq j \leq N$, the squared L^2-norm of u_j, on a one-dimensional domain $\Omega = [a,b] \subset \mathbb{R}$, is approximated as

$$\|u_j\|^2_{L^2(\Omega)} = \int_a^b u_j(x)^2\,\mathrm{d}x \approx \sum_{i=2}^{N_u} \frac{u_j(x_{i-1})^2 + u_j(x_i)^2}{2} \Delta_{x_i},$$

where $\Delta_{x_i} = x_i - x_{i-1}$ is the length of the ith subinterval $[x_{i-1}, x_i]$.

Later in Section 5.3.4, we compare the results obtained by using trapezoidal integration, described above, and a Monte-Carlo integration [21]:

$$\|u_j\|^2_{L^2(\Omega)} \approx \frac{b-a}{N_u} \sum_{i=1}^{N_u} u_j(x_i)^2,$$

which has a lower convergence rate to the integral with respect to the number of points, N_u. This integration technique is, however, particularly suited for approximating integrals in high dimensions, or with complex geometries [21]. One could also use a mesh of the domain and compute the integrals with a quadrature rule on each cell.

It is also possible to incorporate some prior knowledge about the Green's function in the loss function, by adding a penalty term. If the differential operator is self-adjoint, then depending on the constraint operator $\mathcal{D}$, the associated Green's function is symmetric, *i.e.*, $G(x,y) = G(y,x)$ for all $x, y \in \Omega$. In this case, one can train a symmetric NN $\mathcal{N}_G$ defined as

$$\mathcal{N}_G(x,y) = \mathcal{N}(x,y) + \mathcal{N}(y,x), \quad x, y \in \Omega.$$

However, our numerical experiments reveal that the NNs can learn both boundary conditions and symmetry properties directly, from the training data, without additional constraints on the loss function or network architectures.

5.2.4 Optimization algorithm

The NNs are implemented with single-precision floating-point format within the TensorFlow DL library [1], and are trained[2] using a two-step optimization procedure to minimize the loss function. First, we use Adam's algorithm [108] for the first 1000 optimization steps (or epochs), with default learning rate 0.001 and parameters $\beta_1 = 0.9$, $\beta_2 = 0.999$. Then, we employ the limited memory BFGS, with bound constraints (L-BFGS-B) optimization algorithm [37, 132], implemented in the SciPy library [229], with a maximum of 5×10^4 iterations. This training procedure is used by Lu *et al.* to train physics-informed NNs (PINNs) and mitigate the risk of the optimizer getting stuck at poor local minima [136]. The L-BFGS-B algorithm is also successful for PDE learning [180] and PDE solvers using DL techniques [136, 184]. Moreover, this optimization algorithm takes advantage of the smoothness of the loss function by using quasi-Newton approximations to second-order derivatives and often converges in fewer iterations than Adam's algorithm and other methods based on stochastic gradient descent [136]. Within this setting, rational NNs are beneficial because the activation functions are smooth while maintaining an initialization close to ReLU.

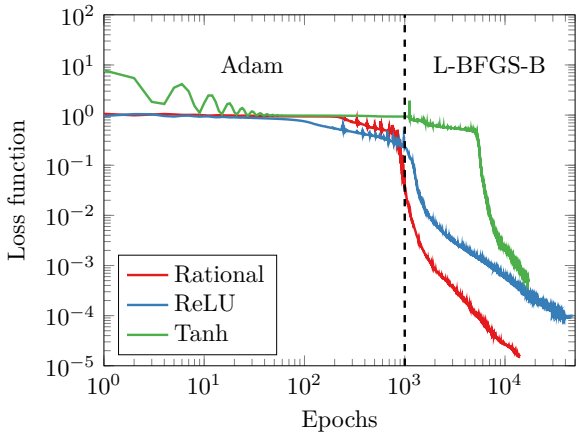


Figure 5.3: Loss function during training. Loss function magnitudes for the ReLU, tanh, and rational NNs with respect to the number of epochs. The networks are trained to learn the Green's function of the Helmholtz operator with homogeneous Dirichlet boundary conditions and frequency $K = 15$. Adam's optimizer is used until 1000 epochs (before the dashed line) and L-BFGS-B is employed thereafter.

In Figure 5.3, we display the value of the loss function during the training of the NNs with different activation functions: rational, ReLU, and hyperbolic tangent

[2]The numerical experiments are performed on a desktop computer with a Intel® Xeon® CPU E5-2667 v2 @ 3.30GHz and a NVIDIA® Tesla® K40m GPU.

(tanh). In this example, we aim to learn the Green's function of a high-frequency Helmholtz operator with homogeneous Dirichlet boundary conditions on $\Omega = [0, 1]$:

$$\mathcal{L}u = \frac{d^2u}{dx^2} + K^2u, \quad u(0) = u(1) = 0, \tag{5.5}$$

where $K = 15$ denotes the Helmholtz frequency. Note that the operator defined by Equation (5.5) is indefinite but invertible. We first remark in Figure 5.3 that the rational NN is easier to train than the other NNs, as it minimizes the loss function to 10^{-5} with ≈ 15000 epochs, while a ReLU NN requires three times as many epochs to reach 10^{-4}. We also see that the loss function for the ReLU and rational NN becomes more oscillatory [18] and harder to minimize before epoch 1000, while it converges much faster after switching to L-BFGS-B. In theory, one could introduce a variable learning rate that improves the behavior of Adam's optimizer [69, 204]. However, that introduces an additional parameter, which is not desirable in the context of PDE learning. We aim to design an adaptive and easy-to-use method that does not require extensive hyperparameter tuning. We also observe that the tanh NN has a similar convergence rate to the rational NN due to the smoothness of the activation function, but this network exhibits instability during training, as indicated by the high value of the loss function when the optimization terminates. Rational NNs do not suffer from this issue, thanks to the initialization close to a ReLU NN, as can be observed in Figure 5.3, when focusing on the value of the loss function corresponding to the early optimization steps.

5.2.5 Measuring the results

Once the NNs have been trained, we visualize the Green's functions by sampling the networks on a fine 1000×1000 grid of $\Omega \times \Omega$. In the case where the exact Green's function G_{exact} is known, we measure the accuracy of the trained NN, $\mathcal{N}_G$, using a relative error in the L^2-norm:

$$\text{Relative Error} = 100 \times \|G_{\text{exact}} - \mathcal{N}_G\|_{L^2(\Omega)} / \|G_{\text{exact}}\|_{L^2(\Omega)}. \tag{5.6}$$

Here, we multiplied by 100 to obtain the relative error as a percentage (%). This illustrates an additional advantage of using a Green's function formulation: we can create test case problems with known Green's functions and evaluate the method using relative error and offer performance guarantees on benchmark problems. The standard approaches in the literature often use best-case and worst-case examples as testing procedures and therefore do not guarantee that the solution operator is

accurately learned. The "worst-case" examples can be misleading if they consist of functions with similar behavior to the forcing terms already included in the training dataset. Furthermore, since the space of possible forcing terms is of infinite dimension, it is not possible to evaluate the trained NNs on all these functions to obtain a true worst-case example.

5.3 Robustness of the method

We test the robustness of our DL method for learning Green's functions and homogeneous solutions of differential equations, with respect to the number of training pairs, the discretization of the solutions and forcing terms, and the noise perturbation of the training solutions, $\{u_j\}_{j=1}^N$. For consistency, we perform numerical experiments where we learn the Green's function of the Helmholtz operator with parameter $K = 15$ and homogeneous Dirichlet boundary conditions (see Equation (5.5)). The performance is measured using the relative error in the L^2-norm defined in Equation (5.6) between the trained network, $\mathcal{N}_G$, and the exact Green's function, G_{exact}, whose analytic expression is given by

$$G_{\text{exact}}(x,y) = \begin{cases} \frac{\sin(15x)\sin(15(y-1))}{15\sin(15)}, & \text{if } x \leq y, \\ \frac{\sin(15y)\sin(15(x-1))}{15\sin(15)}, & \text{if } x > y, \end{cases}$$

where $x, y \in [0,1]$.

5.3.1 Influence of the activation function on the accuracy

We first compare the performances of different activation functions for learning the Green's functions of the Helmholtz operator by training the NNs, $\mathcal{N}_G$ and $\mathcal{N}_{\text{hom}}$, with rational, ReLU, and tanh activation functions. The numerical experiments are repeated ten times to study the statistical effect of the random initialization of the network weights and the stochastic nature of Adam's optimizer. The rational NN achieves a mean relative error of 1.2% (with a standard deviation of 0.2%), while the ReLU NN reaches an average error of 3.3% (with a standard deviation of 0.2%), which is about three times larger. Note that the ten times difference in the loss function between ReLU and Rational NNs, displayed in Figure 5.3, is consistent with the factor of three in the relative error since the loss is a mean squared error and $\sqrt{10} \approx 3$. This indicates that the rational neural networks are not overfitting the training dataset. One of the numerical experiments with a tanh NN terminated early due to the training instabilities mentioned in Section 5.2.3, achieving a relative error

of 99%. We excluded this problematic run when comparing the ReLU and rational NN's accuracy, limiting ourselves only to cases where the training was successful. The ReLU and rational NNs did not suffer from such issues and were always successful. The averaged relative error of the tanh NN, over the nine remaining experiments, is equal to 3.9% (with a standard deviation of 1.4%), which is slightly worse than the ReLU NN, with higher volatility of the results.

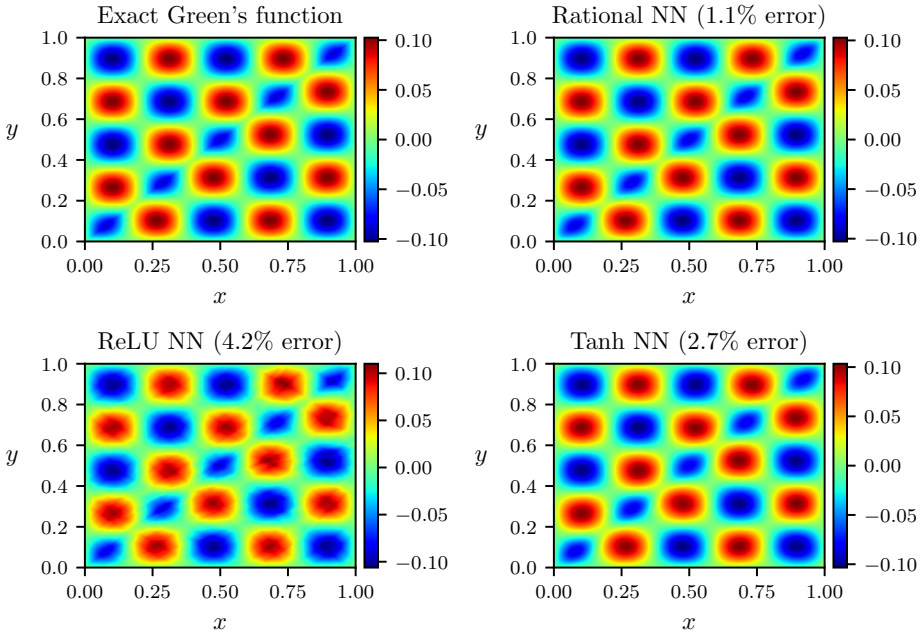


Figure 5.4: Comparison of activation functions. Exact and learned Green's functions of the Helmholtz operator by a rational, ReLU, and tanh NN. The relative error in the L^2 norm is reported in the titles of the panels.

The exact and learned Green's functions with rational, ReLU, and tanh NNs are displayed in Figure 5.4. We see that the rational and tanh NNs are smooth approximations of the exact Green's function, while visual artifacts are present for the ReLU NN as it is piecewise linear, despite its good accuracy.

5.3.2 Number of training pairs and spatial measurements

This section describes our method's accuracy as we change the number of training pairs and the size of the spatial discretization. First, we fix the number of spatial measurements to be $N_u = 100$, and then vary the number of input-output pairs, $\{(f_j, u_j)\}_{j=1}^{N}$, of the training dataset for the Helmholtz operator with Dirichlet boundary conditions (see Equation (5.5)). As we increase N from 1 to 100, we report the

relative error of the Green's function learned by a rational NN in Figure 5.5(a). Next, in Figure 5.5(b), we display the relative error on the learned Green's function as we increase N_u from 3 to 100, with $N = 100$ input-output pairs. Note that we only perform the numerical experiments once since we obtained a low variation of the relative errors in Section 5.3.1 when the networks, $\mathcal{N}_G$ and $\mathcal{N}_{\text{hom}}$, have rational activation functions. We observe similar behavior in Figure 5.5(a) and (b), where the relative error first rapidly (exponentially) decreases as we increase the number of functions in our dataset or spatial measurements of the solutions to the Helmholtz equations with random forcing terms. One important thing to notice is our method's ability to learn the Green's function of a high-frequency Helmholtz operator, with only 1% relative error, using very few training pairs. The learning rate of our deep learning technique for this operator appears to be poly-logarithmic, *i.e.*, the number of input-output pairs required to learn the Green's function within accuracy $0 < \epsilon < 1$ behaves like $\mathcal{O}(\text{polylog}(1/\epsilon))$, as predicted by the remark in Section 2.3.1.

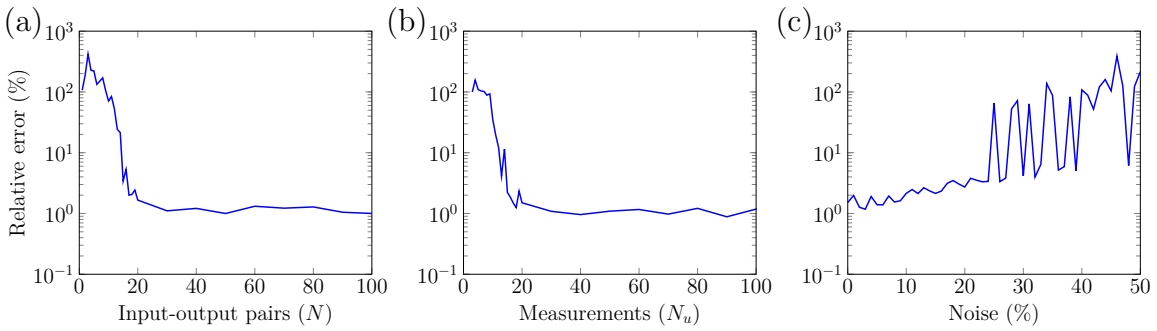


Figure 5.5: Robustness of the method. Relative error of the learned Green's function of the Helmholtz operator with respect to the number of input-output pairs (a), spatial measurements (b), and level of Gaussian noise perturbation (c).

The performance reaches a plateau at $N \approx 20$ and $N_u \approx 20$, respectively, and ceases to improve. However, the stagnation of the relative error for more numerous training data and spatial measurements is expected and can be explained by our choice of covariance kernel length-scale, which restricts the GP's ability to generate a wide variety of forcing terms. Following Section 5.2.1, we chose a normalized length-scale parameter $\lambda = 0.03$, which yields approximately 20 eigenvalues greater than 10^{-2}. This issue can be resolved by decreasing the length-scale parameter and concomitantly increasing the forcing discretization size or choosing another covariance kernel with a less pronounced eigenvalue decay rate (see Section 3.3). In summary, the number of

spatial measurements should be larger than $1/\lambda$ to resolve the forcing terms and the number of input-output pairs should correspond to the number of covariance kernel eigenvalues greater than 10^{-2}.

5.3.3 Noise perturbation

The impact of noise in the training dataset on the accuracy of the learned Green's function is gauged experimentally by perturbing the system's response measurements with Gaussian noise as

$$u_j^{\text{noise}}(x_i) = u_j(x_i)(1 + \delta c_{i,j}), \tag{5.7}$$

where the coefficients $c_{i,j}$ are i.i.d. Gaussian random variables for $1 \leq i \leq N_u$ and $1 \leq j \leq N$, and δ denotes the noise level (in percent). We then vary the level of Gaussian noise perturbation from 0% to 50%, train the NNs, N_G and N_{hom}, for each choice of the noise level, and report the relative error in Figure 5.5(c). We first observe a low impact of the noise level on the accuracy of the learned Green's function, as a perturbation of the system's responses measurements with 20% noise only increases the relative error by a factor of 2 from 1.5% (no noise) to 2.7%. When the level of noise exceeds 25%, we notice large variations of the relative errors and associated higher volatility in results, characterized by a large standard deviation in error associated with repeated numerical experiments. We consider our DL approach relatively robust to noise in the training dataset.

5.3.4 Location of the measurements

As described in Section 5.2.1, by default, we use a uniform grid for spatial measurements of the training dataset, and thus we discretize the integrals in the loss function (cf. Equation (5.4)) using a trapezoidal rule. We conducted additional numerical experiments on the Helmholtz example to study the influence of the measurements' location and quadrature rule on the relative error of the learned Green's function. We report the relative errors between the learned and exact Green's functions in Table 5.1, using a Monte-Carlo or a trapezoidal rule to approximate the integrals and uniform or random spatial measurements. In the latter case, the measurement points $\{x_i\}_{i=1}^{N_u}$ are independently and identically sampled from a uniform distribution, $\mathcal{U}(0,1)$, where $\Omega = [0,1]$ is the domain. We find that the respective relative errors vary between 0.96% and 1.3%. Therefore, we do not observe statistically significant differences in the relative error computed by rational NNs. These results support the

claim that our method is relatively robust to the type of spatial measurements in the training dataset.

Table 5.1: Choice of quadrature rules. Relative error of the Green's function of the Helmholtz operator with frequency $K = 15$ learned by a rational NN with respect to the type of spatial measurements and quadrature rule (Monte-Carlo or trapezoidal rule) used.

Spatial measurements	Monte-Carlo	Trapezoidal rule
Random	1.1%	1.3%
Uniform	1.3%	0.96%

5.3.5 Missing measurements data

Since experimental data may be partially corrupted or unavailable at some spatial locations, we assess our method's accuracy with respect to missing measurement data in the training dataset. We consider the high-frequency Helmholtz operator, defined on the domain $\Omega = [0, 1]$ by Equation (5.5), with homogeneous Dirichlet boundary conditions. We introduce a gap in the spatial measurements located at $x \in [0.5, 0.7]$ by sampling the system's responses, $\{u_j\}_{j=1}^N$, uniformly on the domain $[0, 0.5] \cup [0.7, 1]$. Note that the forcing terms, $\{f_j\}_{j=1}^N$, are still sampled uniformly on the whole domain since they are assumed to be known. The Green's function and homogeneous solution learned by the rational NNs are displayed in Figure 5.6(a) and (b), respectively. Surprisingly, we find that the NN, $\mathcal{N}_G$, can capture the high-frequency pattern of the Green's function and achieves a relative error of 8.2%, despite the large gap within the measurement data for $x \in [0.5, 0.7]$. Another interesting outcome of this numerical experiment is that the lack of spatial measurements in a specific interval does not influence the accuracy of our method outside this location, *i.e.*, for $x \in [0, 0.5] \cup [0.7, 1]$ and $y \in [0, 1]$ in this example. This phenomenon might be explained by the existence of non-local effects when expressing the solution operator associated with the PDE as an integral operator.

5.4 Human-understandable features

The trained NNs contain both the desired Green's function and homogeneous solution, which we evaluate and visualize to glean novel insights concerning the underlying governing PDE (Figure 5.7). In this way, we achieve one part of our human interpretation goal: finding a link between the properties of the Green's function and that of the underlying differential operator and solution constraints.

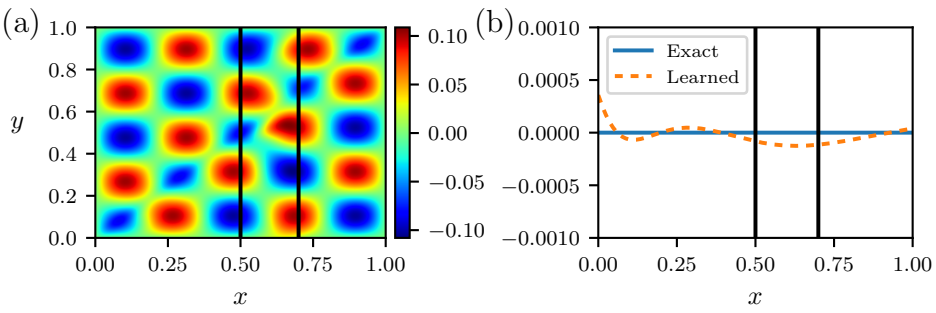


Figure 5.6: Gap in measurements. (a) Green's function of the Helmholtz operator and its homogeneous solution (b) learned by a rational NN with no measurement points for $x \in [0.5, 0.7]$. The space between the vertical black lines indicates where there is a lack of spatial measurements.

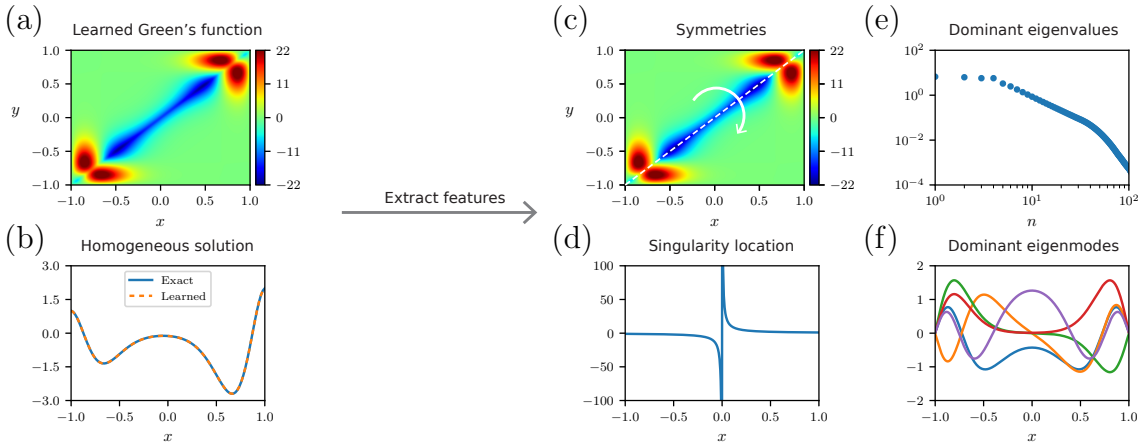


Figure 5.7: Feature extraction from learned Green's functions. The NNs for the learned Green's function (a) and homogeneous solution (b) enable the extraction of qualitative and quantitative features associated with the differential operator. For example, the symmetries in the Green's function reveal PDE invariances (c), poles of rational NNs identify singularity type and location (d), the dominant eigenvalues (e) and eigenmodes (f) of the learned Green's function are related to the eigenvalues and eigenmodes of the differential operator.

As an example, if the Green's function is symmetric, *i.e.*, $G(x, y) = G(y, x)$ for all $x, y \in \Omega$, then the operator $\mathcal{L}$ is self-adjoint. Another aspect of human interpretability is that the poles of the trained rational NN tend to cluster in a way that reveal the location and type of singularities in the homogeneous solution, discussed further in Section 5.4.5 below. Finally, there is a direct correspondence between the dominant eigenmodes and eigenvalues (as well as the singular vectors and singular values) of the learned Green's function and those of the differential operator. The correspondence gives insight into the important eigenmodes that govern the system's behavior (see Sections 5.4.2 and 5.4.3 below). This section highlights that several features of the differential operators can be extracted from the learned Green's function, which supports our aim of uncovering mechanistic understanding from input-output pairs of forcing terms and solutions.

5.4.1 Linear constraints and symmetries

We first remark that boundary constraints, such as the constraint operator, $\mathcal{D}$, of Equation (5.1), can be recovered from the Green's function, G, of the differential operator, $\mathcal{L}$. Let $f \in C_c^\infty(\Omega)$ be any infinitely differentiable function with compact support on Ω, and u be the solution to Equation (5.1) with forcing term, f, such that

$$u(x) = \int_\Omega G(x, y) f(y) \, \mathrm{d}y + u_{\mathrm{hom}}(x), \quad x \in \Omega.$$

Under sufficient regularity conditions, the linearity of the operator, $\mathcal{D}$, implies that $\mathcal{D}(G(\cdot, y), \Omega) = 0$ for all $y \in \Omega$. For instance, if $\mathcal{D}$ is the Dirichlet operator: $\mathcal{D}(u, \Omega) = u_{|\partial\Omega}$, then the Green's function satisfies $G(x, y) = 0$ for all $x \in \partial\Omega$.

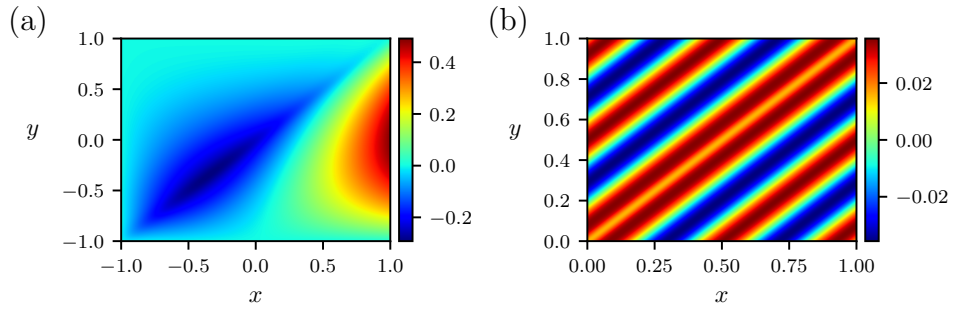


Figure 5.8: Extraction of linear constraints. (a) Learned Green's functions of a second-order differential operator with an integral constraint defined in Equation (5.8). (b) Green's function of the Helmholtz operator with periodic boundary conditions learned by a rational NN.

As an example, we display in Figure 5.8(a) the learned Green's function of the following second-order differential operator on $\Omega = [-1, 1]$ with an integral constraint on the solution:

$$\mathcal{L}u = \frac{du^2}{dx^2} + x^2 u, \quad u(-1) = 1, \quad \int_{-1}^{1} u(x)\,\mathrm{d}x = 2. \tag{5.8}$$

We observe that $G(-1, y) = 0$ for all $y \in [-1, 1]$ and one can verify that the relation $\int_{-1}^{1} G(x, y)\,\mathrm{d}x = 0$ holds for any $y \in [-1, 1]$. In a second example, we learn the Green's function of the Helmholtz operator on $\Omega = [0, 1]$ with frequency $K = 15$ and periodic boundary conditions: $u(0) = u(1)$. One can see in Figure 5.8(b) that the Green's function itself is periodic and that $G(0, y) = G(1, y)$ for all $y \in [0, 1]$, as expected. The periodicity of the Green's function in the y-direction: $G(x, 0) = G(x, 1)$ for $x \in [0, 1]$, is due to the fact that the Helmholtz operator is self-adjoint, which implies symmetry in the associated Green's function. Furthermore, any linear constraint $\mathcal{C}(u) = 0$ such as linear conservation laws or symmetries [167], satisfied by all the solutions to Equation (5.1), under forcing $f \in C_c^\infty(\Omega)$, is also satisfied by the Green's function, G, such that $\mathcal{C}(G(\cdot, y)) = 0$ for all $y \in \Omega$, and is therefore witnessed by the Green's function.

5.4.2 Eigenvalue decomposition

Let $\mathcal{L}$ be a self-adjoint operator and consider the following eigenvalue problem:

$$\mathcal{L}v = \lambda v, \quad \mathcal{D}(v, \Omega) = 0, \tag{5.9}$$

where v is an eigenfunction of the differential operator, $\mathcal{L}$, satisfying the homogeneous constraints with associated eigenvalue, $\lambda > 0$. The eigenfunction, v, can be expressed using the Green's function, G, of $\mathcal{L}$ as

$$v(x) = \lambda \int_\Omega G(x, y)v(y)\,\mathrm{d}y, \quad x \in \Omega,$$

which implies that v is also an eigenfunction of the integral operator with kernel G, but with eigenvalue $1/\lambda$. Consider now the eigenvalue problem associated with the Green's function, itself:

$$\int_\Omega G(x, y)w(y)\,\mathrm{d}y = \mu w(x), \quad x \in \Omega,$$

where $\mu > 0$. Then, we find that $(w, 1/\mu)$ are solutions to the eigenvalue problem (5.9). Consequently, the differential operator, $\mathcal{L}$, and integral operator with kernel, G, share the same eigenfunctions, but possess reciprocal eigenvalues [207]. Thus,

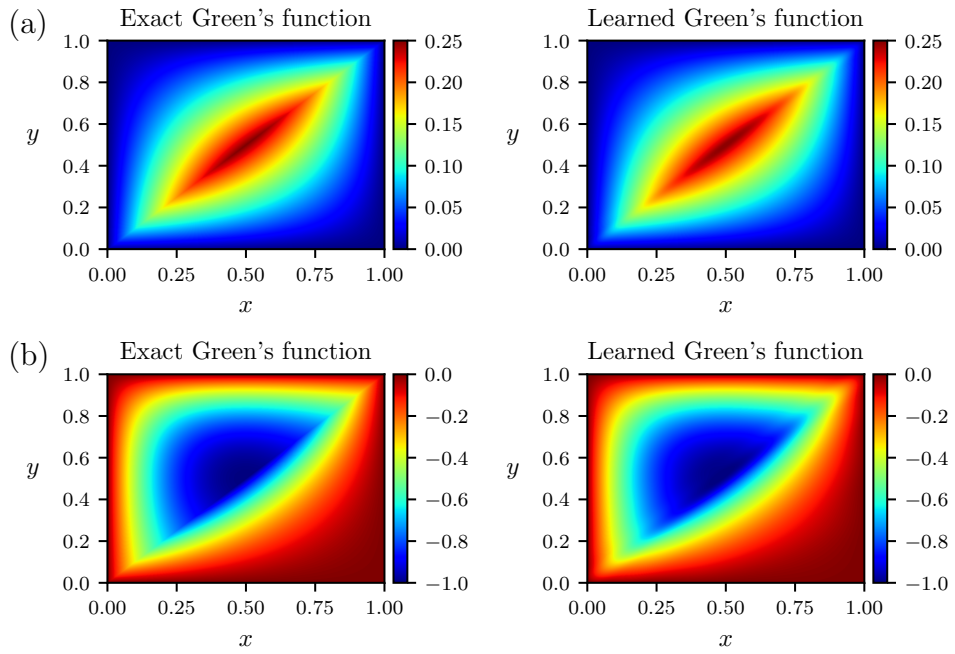


Figure 5.9: Laplace and advection-diffusion operators. Exact and learned Green's functions of the Laplace (a) and advection-diffusion (b) operators.

we can effectively compute the lowest eigenmodes of $\mathcal{L}$ from the learned Green's function.

We now evaluate our method's ability to accurately recover the eigenfunctions of the Green's function that are associated with the largest eigenvalues, in magnitude, from input-output pairs. We train a NN to learn the Green's function of the Laplace operator $\mathcal{L}u = -d^2u/dx^2$ on $[0, 1]$, with homogeneous Dirichlet boundary conditions, and numerically compute its eigenvalue decomposition. In Figure 5.9(a), we display the learned and exact Green's function, whose expression is given for $x, y \in [0, 1]$ by

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

The one hundred largest eigenvalues in magnitude, along with the corresponding first five eigenfunctions, are visualized for the exact and learned Green's functions in Figure 5.10. Note that the eigenvectors of the learned Green's functions are normalized and flipped to match the ones of the exact Green's function because eigenfunctions are unique up to a scalar multiple when the eigenvalues are all distinct. We find that we can recover the largest eigenvalues and eigenfunctions of the learned Green's function and that the first 20 largest eigenvalues remain accurate. Therefore, the approximation error between the learned and exact Green's functions mainly affects the

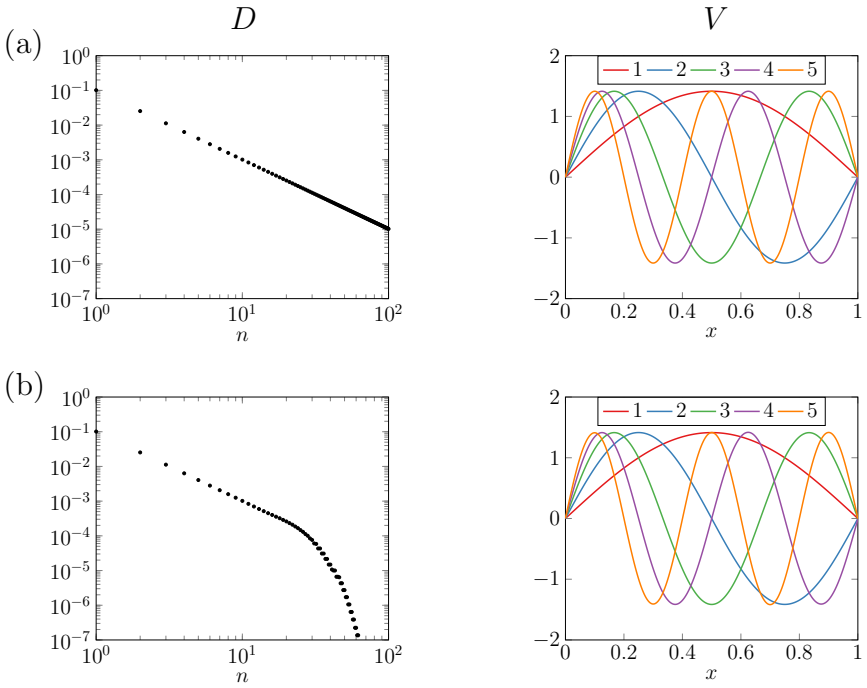


Figure 5.10: Eigenvalue decomposition. The first 100 largest eigenvalues and first five eigenfunctions of the exact (a) and learned (b) Green's functions of the Laplace operator. The eigenvalues are represented in the left panels, while the right panels illustrate the first five eigenfunctions of the Green's function.

smallest eigenvalues. This is an essential feature of our method since the dominant eigenmodes of the differential operator $\mathcal{L}$ are associated with the largest eigenvalues of the Green's functions, which can be learned accurately. The exponential decay of the smallest eigenvalues of the learned Green's function in the left panel of Figure 5.10(b) is because the rational NN is a smooth approximation to the exact Green's function.

5.4.3 Singular value decomposition

When the Green's function of the differentiation operator, $\mathcal{L}$, is square-integrable, its associated Hilbert–Schmidt integral operator admits a singular value decomposition (SVD) (see Section 1.6). Then, there exist a positive sequence $\sigma_1 \geq \sigma_2 \geq \cdots > 0$, and two orthonormal bases, $\{\phi_j\}$ and $\{\psi_j\}$, of $L^2(\Omega)$ such that

$$u(x) = \int_\Omega G(x,y) f(y)\,\mathrm{d}y + u_{\mathrm{hom}}(x) = \sum_{\substack{j=1 \\ \sigma_j > 0}}^{\infty} \sigma_j \langle \phi_j, f \rangle \psi_j(x) + u_{\mathrm{hom}}(x), \quad x \in \Omega, \tag{5.10}$$

where u is the solution to Equation (5.1) with forcing term f, and $\langle \cdot, \cdot \rangle$ denotes the inner product in $L^2(\Omega)$. Therefore, the action of the solution operator $f \mapsto u$ can be approximated using the SVD of the integral operator. Similarly to Section 5.4.2 with the eigenvalue decomposition, the dominant terms in the expansion of Equation (5.10) are associated with the largest singular values of the integral operator.

We now show that one can accurately recover the first singular values and singular vectors from the Green's function learned by a rational NN. We train a rational NN to learn the Green's function of an advection-diffusion operator $\mathcal{L}$ on $\Omega = [0,1]$ with Dirichlet boundary conditions, defined as

$$\mathcal{L}u = \frac{1}{4}\frac{d^2u}{dx^2} + \frac{du}{dx} + u, \quad u(0) = 1,\ u(1) = -2. \tag{5.11}$$

The learned Green's function is illustrated in Figure 5.9(a), next to the exact Green's function given by:

$$G_{\mathrm{exact}}(x,y) = \begin{cases} 4x(y-1)\exp(-2(x-y)), & \text{if } x \leq y, \\ (x-1)y, & \text{if } y < x, \end{cases}$$

for $x, y \in [0,1]$. In Figure 5.11, we display the first five left and right singular vectors and the singular values of the exact and learned Green's functions. We observe that the first fifteen singular values of the learned Green's functions are accurate. This leads us to conclude that our method enables the construction of a low-rank representation of the solution operator associated with the differential operator, $\mathcal{L}$, and allows us to compute and analyze its dominant modes.

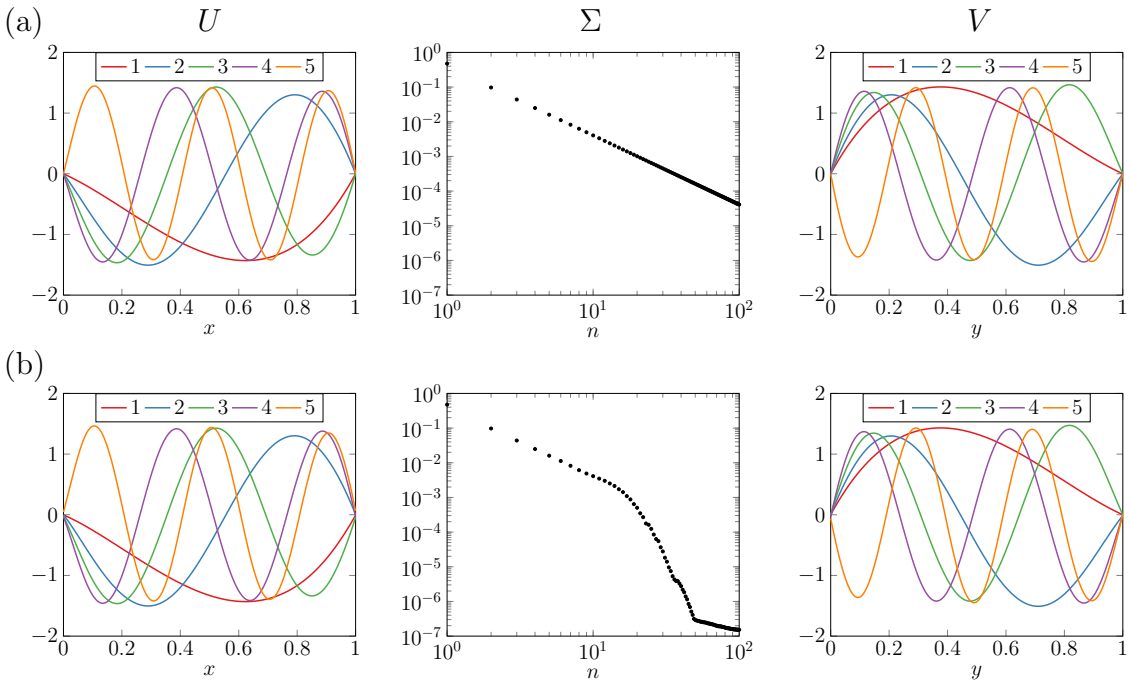


Figure 5.11: Singular value decomposition. Singular value decomposition of the exact (a) and learned (b) Green's functions of the advection-diffusion operator defined by Equation (5.11). The left and right panels, respectively, show the first five left and right singular vectors, $\{\phi\}_{n=1}^{5}$ and $\{\psi\}_{n=1}^{5}$, of the exact and learned Green's functions. The singular values of the Green's functions are plotted in the middle panel.

5.4.4 Schrödinger equation with double-well potential

We highlight the ability of our DL method to learn physical features of an underlying system by considering the steady-state one-dimensional Schrödinger operator on $\Omega = [-3, 3]$:

$$\mathcal{L}(u) = -h^2 \frac{d^2u}{dx^2} + V(x)u, \quad u(-3) = u(3) = 0,$$

with double-well potential $V(x) = x^2 + 1.5\exp(-(4x)^4)$ and $h = 0.1$ [222]. The potential $V(x)$ is illustrated in Figure 5.12, along with the Green's function learned by the rational NN from pairs of forcing terms and the system's responses. First, the shape of the well potential can be visualized along the diagonal of the Green's function in Figure 5.12(b). Next, in Figure 5.12, we compute the first ten eigenstates of the Schrödinger operator in Chebfun [56] and plot them using a similar representation as [222, Figure 6.9]. Similarly to Section 5.4.2, we compute the eigenvalue decomposition of the Green's function learned by a rational NN and plot the eigenstates (shifted by the corresponding eigenvalues) in Figure 5.12. Note that the eigenvalues of the operator and the Green's functions are reversed. We observe a perfect agreement between the first ten exact and learned eigenstates. These energy levels capture information about the states of atomic particles modeled by the Schrödinger equation.

5.4.5 Singularity location and type

The input-output function of a rational NN is a high-degree rational function, which means that it has poles (isolated points for which it is infinite). In rational function approximation theory, it is known that the poles of a near-optimal rational approximant tend to cluster near a function's singularities [223]. The clustering of the poles near the singularity is needed for the rational approximant to have excellent global approximation [205, 206]. Moreover, the type of clustering (algebraic, exponential, beveled exponential) can reveal the type of singularity (square-root, blow-up, non-differentiable) at that location. This feature of rational approximants is used in other settings [20].

We show that the rational NNs also cluster poles in a way that identifies its location and type. In Figure 5.13(c), we display the complex argument of the trained rational NN for the Green's function of a second-order differential operator with a jump condition, defined on $\Omega = [0, 1]$ as

$$\mathcal{L}u = 0.2\frac{d^2u}{dx^2} + \frac{du}{dx}, \quad u(0) = u(1) = 0,\ u(0.7^-) = 2,\ u(0.7^+) = 1.$$

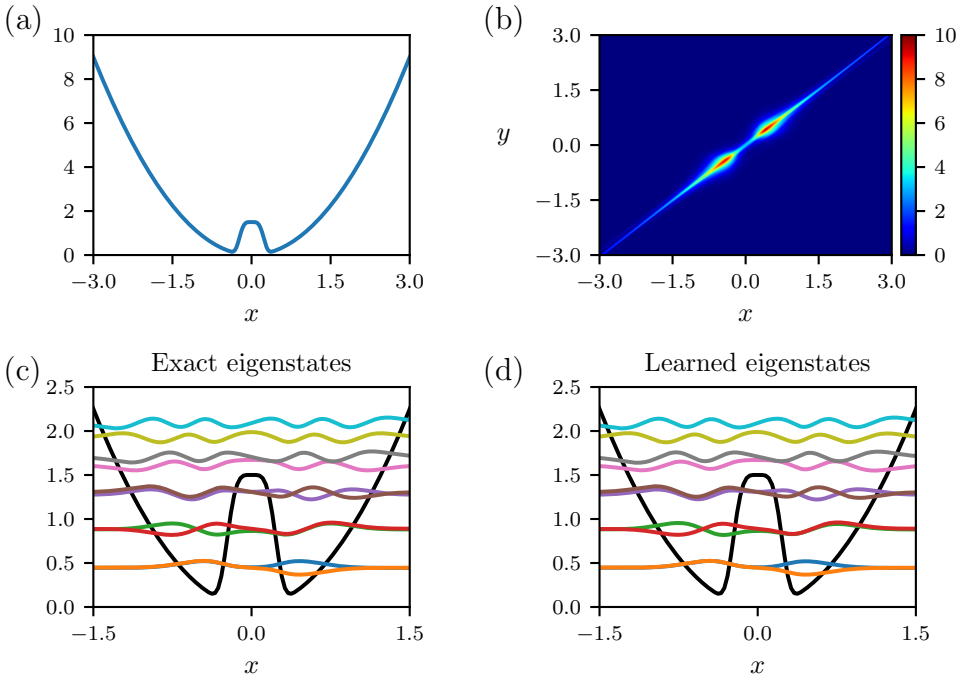


Figure 5.12: Schrödinger equation. (a) Double well potential $V(x) = x^2 + 1.5\exp(-(4x)^4)$. (n) Learned Green's function of the Schrödinger equation with potential $V(x)$. (c) First ten exact eigenstates computed numerically from the Schrödinger operator and (d) eigenstates computed from the learned Green's function displayed in (b). The eigenfunctions are shifted by an amount corresponding to the eigenvalue. The double-well potential is shown as a black curve.

These diagrams are known as phase portraits and are useful for illustrating complex analysis [235]. A pole of the rational function can be identified as a point in the complex plane for which the full colormap goes around that point in a clockwise fashion. In particular, in Figure 5.13(c), we see that the poles of the rational function cluster quite closely to the real-line (where $Im(z) = 0$) at $x = 0.7$. If the clustering is examined more closely, it may be possible to reveal that the singularity in the Green's function at $x = 0.7$ is due to a jump condition.

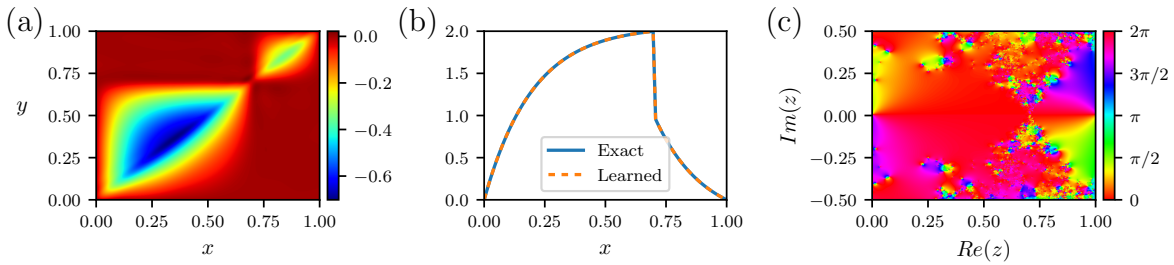


Figure 5.13: Singularity location. (a) Learned Green's function of a second-order differential operator with a jump condition at $x = 0.7$. Homogeneous solution of the operator with jump condition (b) and argument of the rational NN representing the homogeneous solution in the complex plane (c).

Rational NNs are also important for resolving Green's function with boundary layers as the NN can resolve the boundary layer by clustering its poles in the complex plane. In Figure 5.14, we see a learned Green's function of a differential equation with a boundary layer at $x = 0$ with $\nu = 10^{-2}$:

$$\mathcal{L}u = -\nu\frac{d^2u}{dx^2} - \frac{du}{dx}, \quad u(0) = u(1) = 0, \quad \Omega = [0, 1].$$

The analytical expression for the Green's function is given by the following equation:

$$G_{\text{exact}}(x, y) = \begin{cases} \frac{1}{e^{1/\nu}-1}\left(1 - e^{-x\nu}\right)\left(e^{1/\nu} - e^{y/\nu}\right), & \text{if } x \leq y, \\ \frac{1}{e^{1/\nu}-1}\left(1 - e^{(1-x)/\nu}\right)\left(1 - e^{y/\nu}\right), & \text{if } y < x, \end{cases}$$

While the Green's function is not smooth, our rational NN still resolves it with relatively good accuracy, as shown by the sharp interface along the diagonal.

5.5 Viscous shock and multiphysics examples

In this section, we focus on two physical models and analyse the Green's functions discovered by our deep learning approach.

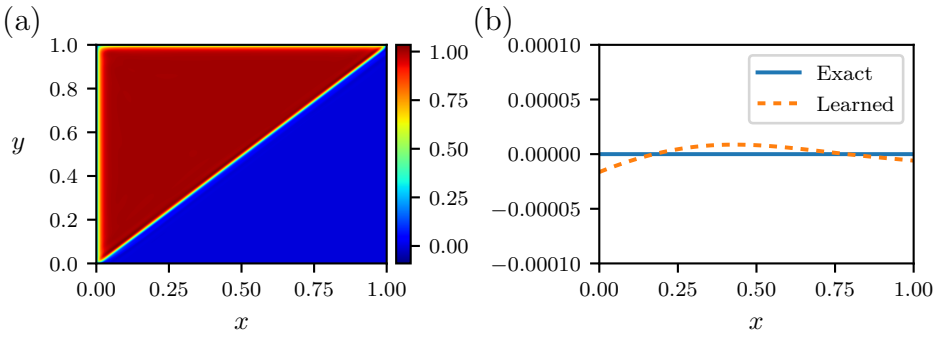


Figure 5.14: Boundary layer. Learned Green's function (a) and homogeneous solution (b) to a differential equation with a boundary layer around $x = 0$.

5.5.1 Viscous shock

As a first example, we consider a second-order differential operator having suitable variable coefficients to model a viscous shock at $x = 0$ [122]:

$$\mathcal{L}u = 10^{-3}\frac{d^2u}{dx^2} + 2x\frac{du}{dx}, \quad u(-1) = -1,\ u(1) = 1.$$

The system's responses are obtained by solving the PDE, with Dirichlet boundary conditions, using a spectral numerical solver for each of the $N = 100$ random forcing terms, sampled from a GP having a squared-exponential covariance kernel [32]. The learned Green's function is displayed in Figure 5.15(a) and satisfies the following symmetry relation: $G(x, y) = G(-x, -y)$, indicating the presence of a reflective symmetry group within the underlying PDE. Indeed, if u is a solution to $\mathcal{L}u = f$ with homogeneous boundary conditions, then $u(-x)$ is a solution to $\mathcal{L}v = f(-x)$. We also observe in Figure 5.15(b) and (c) that the homogeneous solution is accurately captured and that the poles of the homogeneous rational NN cluster near the real axis around $x = 0$: the location of the singularity induced by the shock (cf. Section 5.4.5).

Next, we reproduce the same viscous shock numerical experiment, except that this time we remove measurements of the system's response from the training dataset in the interval $[-0.2, 0.2]$: adjacent to the shock front. By comparing Figure 5.15(d)-(f) and Figure 5.15(a)-(c), we find that the Green's function and homogeneous solution, learned by the rational NNs, may not be affected in the region outside of the interval with missing data. In some cases, the NNs can still accurately capture the main features of the Green's function and homogeneous solution in the region lacking measurements. The robustness of our method to noise perturbation and corrupted or missing data is of significant interest and promising for real applications with experimental data.

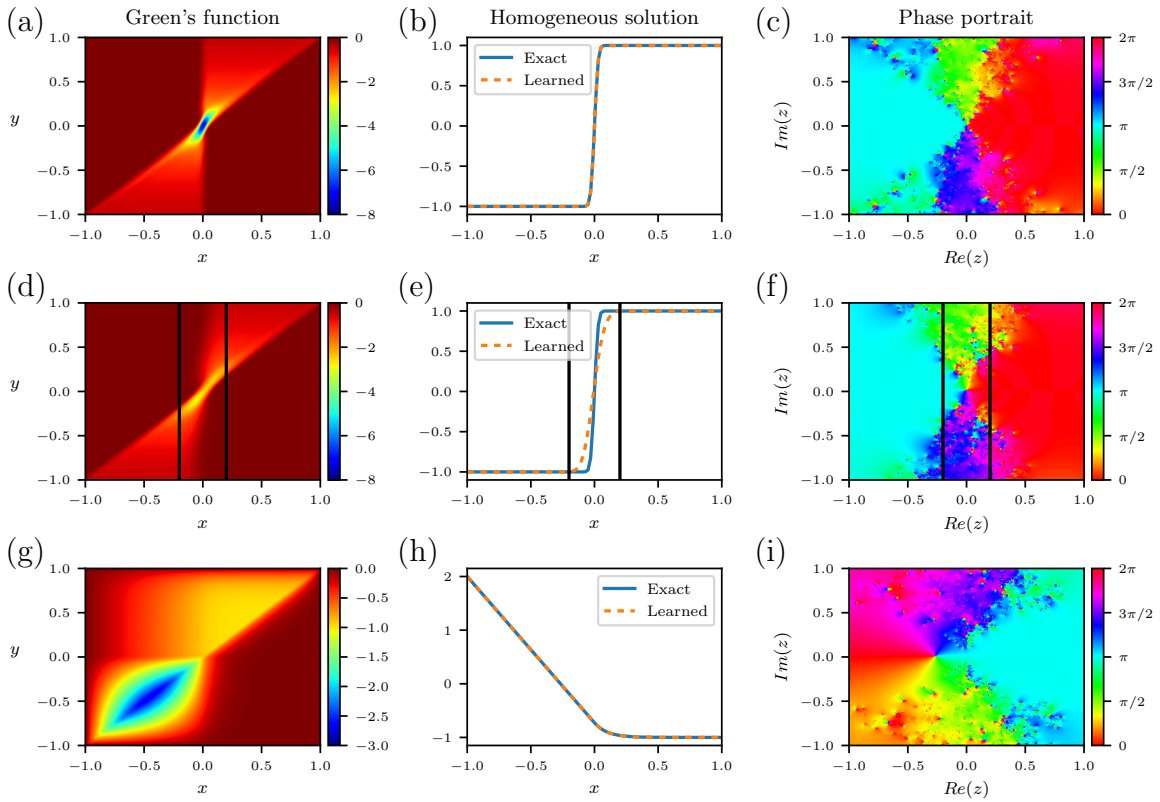


Figure 5.15: Green's functions learned by rational neural networks. (a) Green's function of a differential operator with a viscous shock at $x = 0$, learned by a rational NN. (b) Learned and exact (computed by a classical spectral method) homogeneous solution to the differential equation with zero forcing term. (c) Phase portrait of the homogeneous rational NN evaluated on the complex plane. (d)-(f) Similar to (a)-(c), but without any system's response measurements in $x \in [-0.2, 0.2]$ (see vertical black lines) near the shock. (g) Learned Green's function and homogeneous solution (h) of an advection-diffusion operator with advection occurring for $x \geq 0$. (i) Phase portrait of the homogeneous NN on the complex plane.

5.5.2 Advection-diffusion operator

We next apply our DL method to discover the Green's function and homogeneous solution of an advection-diffusion operator, where the advection is dominant only within the right half of the domain:

$$\mathcal{L}u = 0.1\frac{d^2u}{dx^2} + \mathbb{I}_{(x\geq 0)}\frac{du}{dx}, \quad u(-1) = 2,\, u(1) = -1,$$

on $\Omega = [-1, 1]$. Here, $\mathbb{I}_{(x\geq 0)}$ denotes the characteristic function on $x \geq 0$. The resulting equation is diffusive on the left half of the domain, while the advection is turned on for $x \geq 0$. The output of the Green's function NN is plotted in Figure 5.15(g), where we observe the disparate spatial behaviors of the dominant physical mechanisms. This can be recognized when observing the restriction of the Green's function to the subdomain $[-1, 0] \times [-1, 0]$, where the observed solution is reminiscent of the Green's function for the Laplacian; thus indicating that the PDE is diffusive on the left half of the domain. Similarly, the restriction of the learned Green's function to $[0, 1] \times [0, 1]$ is characteristic of advection.

In Figure 5.15(h) and (i), we display the homogeneous solution NN, along with the phase of the rational NN, evaluated on the complex plane. The agreement between the exact and learned homogeneous solution illustrates the ability of the DL method to accurately capture the behavior of a system within "multiphysics" contexts. The choice of rational NNs is crucial here: to deepen our understanding of the system, as the poles of the homogeneous rational NN characterize the location and type of singularities in the homogeneous solution. Here the change in behavior of the differential operator from diffusion to advection is delineated by the location of the poles of the rational NN.

5.6 Two-dimensional operators and systems

Our deep learning technique for learning Green's functions generalizes well in two dimensions and for systems of linear partial differential equations as we will see in this section.

5.6.1 Differential operators in two dimensions

We demonstrate the ability of our method to learn Green's functions associated with two-dimensional operators by repeating the numerical experiment of [126], which

consists of learning the Green's function of the Poisson operator on the unit disk $\Omega = D(0,1)$, with homogeneous Dirichlet boundary conditions:

$$\mathcal{L}u = \nabla^2 u, \quad u_{|\partial D(0,1)} = 0.$$

This experiment is a good benchmark for PDE learning techniques as the analytical expression of the Green's function in Cartesian coordinates can be expressed as [157]:

$$G_{\text{exact}}(x, y, \tilde{x}, \tilde{y}) = \frac{1}{4\pi} \ln \left(\frac{(x-\tilde{x})^2 + (y-\tilde{y})^2}{(x\tilde{y} - \tilde{x}y)^2 + (x\tilde{x} + y\tilde{y} - 1)^2} \right),$$

where $(x, y), (\tilde{x}, \tilde{y}) \in D(0,1)$.

The training dataset for this numerical example is created as follows. First, we generate $N = 100$ random forcing terms using the command `randnfundisk` of the Chebfun software [56, 66, 237] with a frequency parameter of $\lambda = 0.2$, and then solve the Poisson equation, with corresponding right-hand sides, using a spectral method. Then, the forcing terms and system responses (*i.e.* solutions) are sampled at the $N_u = N_f = 673$ nodes of a disk mesh, generated using the Gmsh software [70]. Moreover, the mesh structure ensures that the repartition of the sample points is approximately uniform in the disk (Figure 5.16(c)) and that the boundary is accurately captured.

The Green's function and homogeneous rational NNs have four hidden layers and width of 50 neurons, with 4 and 2 input nodes, respectively, as the Green's function is defined on $\Omega \times \Omega$. The two-dimensional integrals of the loss function (5.4) are discretized using uniform quadrature weights: $w_i = \pi/N_f$ for $1 \leq i \leq N_f$. In Figure 5.16(d)-(g), we visualize four two-dimensional slices of the learned Green's function together with two slices of the exact Green's function in panels (a) and (b). Because of the symmetry in the Green's function, due to the self-adjointness of $\mathcal{L}$ and the boundary constraints, the exact Green's function satisfies $G(x, y, 0, 0) = G(0, 0, x, y)$ for $(x, y) \in D(0,1)$. Therefore, we compare Figure 5.16(a) to Figure 5.16(d)-(e), and similarly for Figure 5.16(b) and Figure 5.16(f)-(g). We observe that the Green's function is accurately learned by the rational NN, which preserves low approximation errors near the singularity at $(x, y) = (\tilde{x}, \tilde{y})$, contrary to the neural operator technique [126]. The visual artifacts present in Figure 5.16(e)-(g) are likely due to the low spatial discretization of the training data. One could increase the number of spatial measurements or use a high-order quadrature rule.

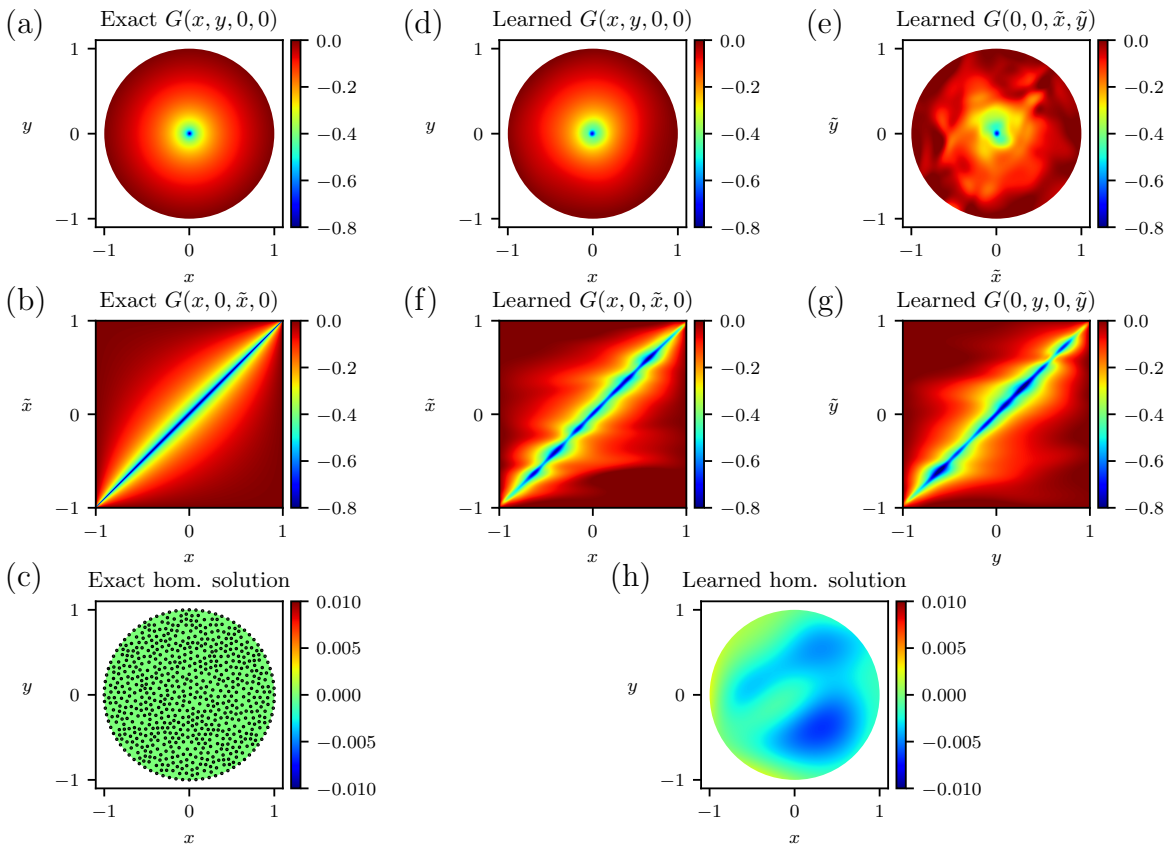


Figure 5.16: Poisson equation on the disk. Exact (a)-(b) and learned (d)-(f) Green's function of the Poisson operator on the unit disk, evaluated at two-dimensional slices. The colorbar is scaled to remove the singularity of the Green's function at $(x, y) = (\tilde{x}, \tilde{y})$. (c) Exact homogeneous solution with sample points for the training functions and (h) homogeneous solution learned by the rational NN.

5.6.2 System of differential equations

The method extends also naturally to systems of differential equations. Let $f = \begin{bmatrix} f^1 & \cdots & f^{n_f} \end{bmatrix}^\top : \Omega \to \mathbb{R}^{n_f}$ be a vector of n_f forcing terms and $u = \begin{bmatrix} u^1 & \cdots & u^{n_u} \end{bmatrix}^\top : \Omega \to \mathbb{R}^{n_u}$ be a vector of n_u system responses such that

$$\mathcal{L} \begin{bmatrix} u^1 \\ \vdots \\ u^{n_u} \end{bmatrix} = \begin{bmatrix} f^1 \\ \vdots \\ f^{n_f} \end{bmatrix}, \quad \mathcal{D} \left(\begin{bmatrix} u^1 \\ \vdots \\ u^{n_u} \end{bmatrix}, \Omega \right) = \begin{bmatrix} g^1 \\ \vdots \\ g^{n_u} \end{bmatrix}. \tag{5.12}$$

The solution to Equation (5.12) with $f = 0$ is called the homogeneous solution and denoted by $u_{\text{hom}} = \begin{bmatrix} u^1_{\text{hom}} & \cdots & u^{n_u}_{\text{hom}} \end{bmatrix}^\top$. Similarly to the scalar case, we can express the relation between the system's response and the forcing term using Green's functions and an integral formulation as

$$u^i(x) = \sum_{j=1}^{n_f} \int_\Omega G_{i,j}(x,y) f^j(y) \, \mathrm{d}y + u^i_{\text{hom}}(x), \quad x \in \Omega, \tag{5.13}$$

for $1 \leq i \leq n_u$. Here, $G_{i,j} : \Omega \times \Omega \to \mathbb{R} \cup \{\pm\infty\}$ is a component of the *Green's matrix* for $1 \leq i \leq n_u$ and $1 \leq j \leq n_f$, which consists of a $n_u \times n_f$ matrix of Green's functions:

$$G(x,y) = \begin{bmatrix} G_{1,1}(x,y) & \cdots & G_{1,n_f}(x,y) \\ \vdots & \ddots & \vdots \\ G_{n_u,1}(x,y) & \cdots & G_{n_u,n_f}(x,y) \end{bmatrix}, \quad x, y \in \Omega.$$

Following Equation (5.13), we remark that the differential equations decouple, and therefore we can learn each row of the Green's function matrix independently. That is, for each row $1 \leq i \leq n_u$, we train n_f NNs to approximate the components $G_{i,1}, \ldots, G_{i,n_f}$, and one NN to approximate the ith component of the homogeneous solution, u^i_{hom}.

As an example, we consider the following system of ordinary differential equations (ODEs) on $\Omega = [-1, 1]$:

$$\frac{d^2u}{dx^2} - v = f^1, \tag{5.14a}$$

$$\frac{-d^2v}{dx^2} + xu = f^2, \tag{5.14b}$$

with boundary conditions: $u(-1) = 1$, $u(1) = -1$, $v(-1) = v(1) = -2$. In Figure 5.17, we display the different components of the Green's matrix and the exact solution (computed by a spectral method), along with the learned homogeneous solutions. We find that the Green's function matrix provides insight on the coupling

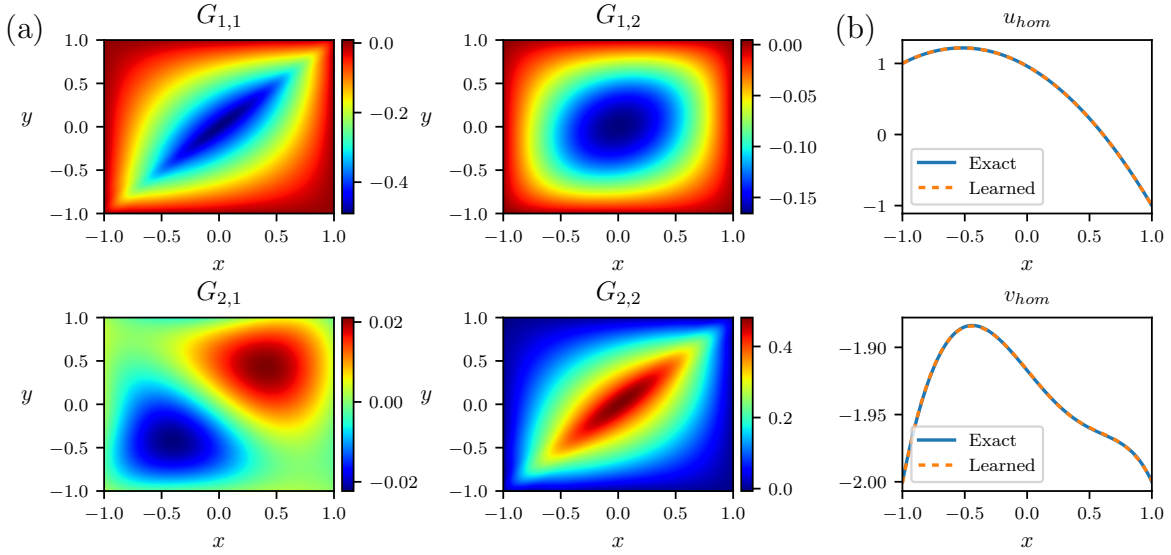


Figure 5.17: Green's matrix of system of ODEs. (a) Matrix of Green's function learned from the system of ordinary differential equations (5.14). (b) Homogeneous solutions associated with the system of ODEs.

between the two system variables, u and v, as shown by the diagonal components $G_{1,2}$ and $G_{2,1}$ of the Green's matrix in Figure 5.17(a). Similarly, the components $G_{1,1}$ and $G_{2,2}$ are characteristic of diffusion operators, which appear in Equation (5.14). In this case, the Green's matrix can be understood as a 2×2 block inverse [137] of the linear operator, $\mathcal{L}$.

5.7 Nonlinear and vector-valued equations

We can also discover Green's functions from forcing terms and concomitant solutions to nonlinear differential equations possessing semi-dominant linearity as well as Green's functions associated with vector-valued equations.

5.7.1 Linearized models of nonlinear operators

We demonstrate that our DL method can be used to linearize and extract Green's functions from nonlinear boundary value problems of the form

$$\mathcal{L}u + \epsilon\mathcal{N}(u) = f, \quad \mathcal{D}(u, \Omega) = g,$$

where $\mathcal{L}$ denotes a linear operator, $\mathcal{N}$ is a nonlinear operator, and $\epsilon < 1$ is a small parameter controlling the nonlinearity. We demonstrate this ability on the three

nonlinear boundary value problems, dominated by the linearity, used in [71]. In Figure 5.18(a)-(c), we visualize the Green's function NNs of three operators with cubic nonlinearity considered in [71].

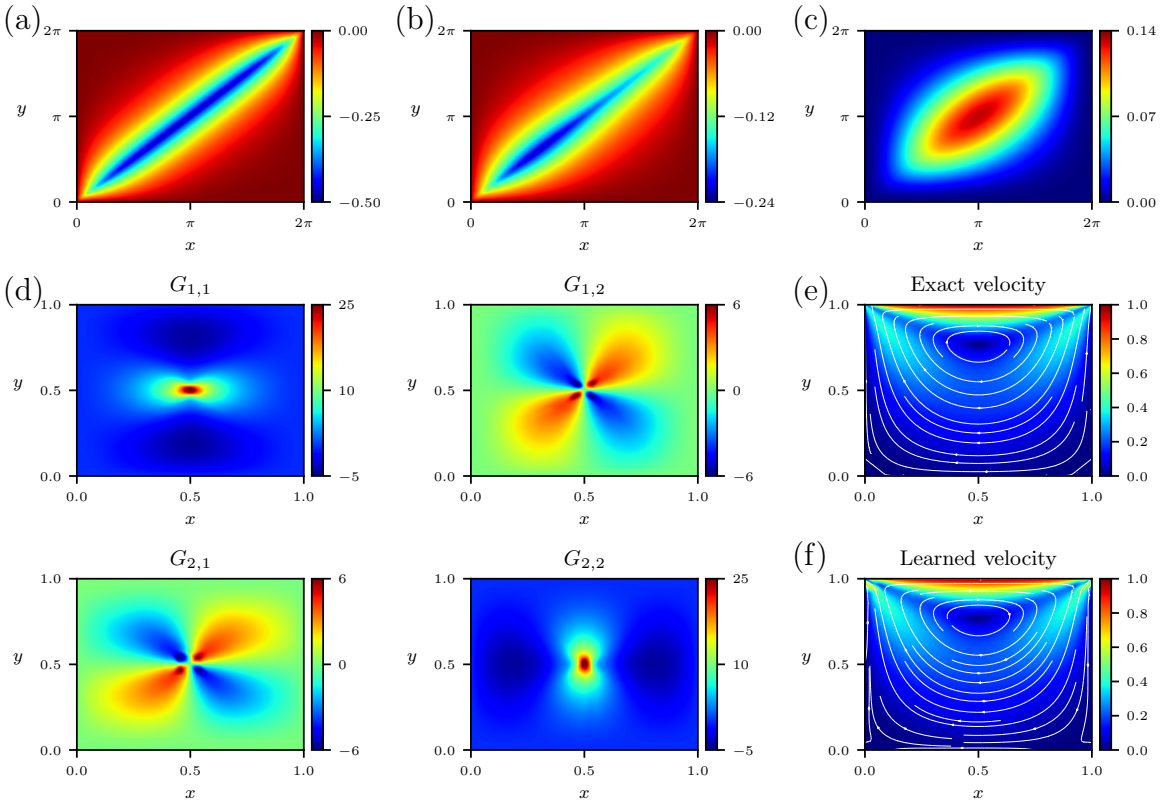


Figure 5.18: Linearized models and Stokes flow. (a)-(c) Green's functions of three differential operators: Helmholtz, Sturm–Liouville, and biharmonic, with cubic nonlinearity. (d) Matrix of Green's functions of a two-dimensional Stokes flow in a lid-driven cavity, evaluated at the two-dimensional slice $(x, y, 0.5, 0.5)$. Velocity magnitude and streamlines of the exact (e) and learned (f) homogeneous solution to the Stokes equations with zero applied body force.

First, Figure 5.18(a) illustrates the learned Green's function of a cubic Helmholtz system on $\Omega = [0, 2\pi]$ with homogeneous Dirichlet boundary conditions:

$$\frac{d^2u}{dx^2} + \alpha u + \epsilon u^3 = f(x),$$

where $\alpha = -1$ and $\epsilon = 0.4$. Next, in Figure 5.18(b), we consider a nonlinear Sturm–Liouville operator of the form:

$$[-p(x)u']' + q(x)(u + \epsilon u^3) = f(x), \quad u(0) = u(2\pi) = 0,$$

with $p(x) = 0.4\sin(x) - 3$, $q(x) = 0.6\sin(x) - 2$, and $\epsilon = 0.4$. The notation u' denotes the derivative with respect to x, du/dx. Finally, the example represented in Figure 5.18(c) is the learned Green's function of a nonlinear biharmonic operator:

$$[-p(x)u'']'' + q(u + \epsilon u^3) = f(x), \quad u(0) = u(2\pi) = 0,$$

where $p = -4$, $q = 2$, and $\epsilon = 0.4$.

The nonlinearity does not prevent our method from discovering a Green's function of an approximate linear model, from which one can understand features such as symmetry and boundary conditions. This property is crucial for tackling time-dependent problems, where the present technique may be extended and applied to uncover linear propagators.

5.7.2 Lid-driven cavity problem

Finally, we consider a classical benchmark in fluid dynamics consisting of Stokes flow in a two-dimensional lid-driven cavity problem [62]. We aim to discover the matrix of Green's functions of the Stokes flow [22], which is modelled by the following system of equations on the domain $\Omega = [0, 1]^2$,

$$\begin{aligned} \mu\nabla^2\mathbf{u} - \nabla p &= \mathbf{f}, \\ \nabla \cdot \mathbf{u} &= 0. \end{aligned}$$

Here, $\mathbf{u} = (u_x, u_y)$ is the fluid velocity, p is the pressure, $\mathbf{f} = (f_x, f_y)$ is an applied body force (*i.e.* a forcing term), and $\mu = 1/100$ is the dynamic viscosity. The fluid velocity satisfies no-slip boundary conditions on the walls, except on the top wall where $\mathbf{u} = (1, 0)$. We first generate one hundred forcing terms, $\mathbf{f}$, with two smooth random components, in the Chebfun software [56, 66] using the `randnfun2` command with wavelength parameter $\lambda = 0.1$. The Stokes equations are then discretized with Taylor–Hood finite elements [23, 212] for the velocity and pressure on a mesh with 96×96 square cells and subsequently solved using the Firedrake finite element library [188]. We illustrate in Figure 5.19 an example of applied body force and velocity solution obtained by solving the system of PDEs. We then create the training dataset for the NNs by sampling the applied body forces and corresponding velocity solutions, $\mathbf{u}$, on a regular 25×25 grid.

In this context, the relation between the system's responses and the forcing terms can be expressed using a Green's matrix, which consists of a two-by-two matrix of Green's functions and whose components reveal features of the underlying system

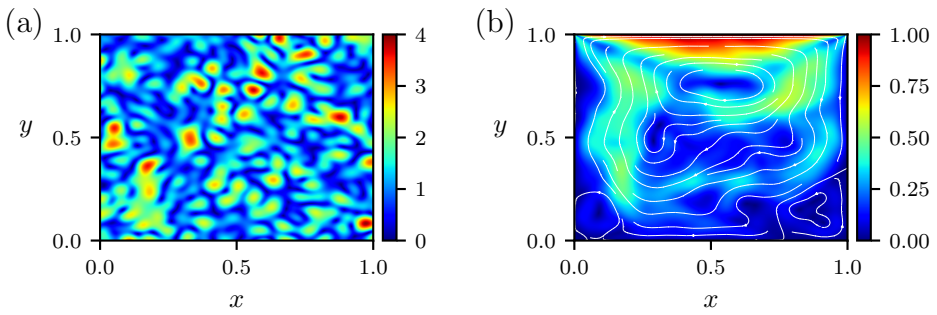


Figure 5.19: Training functions for Stokes flow. (a) Magnitude of a random applied body force used as a forcing term in the Stokes equations. (b) Velocity magnitude and streamlines of the system's response.

such as symmetry and coupling (Figure 5.18(d) and Section 5.6.2). The four Green's functions and two homogeneous NNs have the same architecture as the one described in Section 5.2.2, except that they have respectively four and two input nodes (instead of two and one) due to the current spatial dimension. Figure 5.18(e) and (f) illustrate that the homogeneous solution to the Stokes equation is accurately captured by the homogeneous rational NN, despite the corner singularities and coarse measurement grid. The four components of the Green's matrix for the Stokes flow are evaluated on the two-dimensional slice $(x, y, 0.5, 0.5)$, for $x, y \in [0, 1]$, and displayed in Figure 5.18(d). This figure allows us to visualize the system's response to a point force, $\mathbf{f} = (f_x, f_y)$, located at $(0.5, 0.5)$, with the system's response being denoted as $\mathbf{u} = (u_x, u_y)$, where

$$u_x(x, y) = G_{1,1}(x, y, 0.5, 0.5)f_x + G_{1,2}(x, y, 0.5, 0.5)f_y,$$
$$u_y(x, y) = G_{2,1}(x, y, 0.5, 0.5)f_x + G_{2,2}(x, y, 0.5, 0.5)f_y,$$

for $x, y \in [0, 1]$. The visualization of the $G_{2,2}$ component in Figure 5.18(d), corresponding to the system's response to a unitary vertical point force $\mathbf{f} = (0, 1)$ is reminiscent of [61, Figure 1].

Finally, we evaluate the components of the Green's matrix at three other two-dimensional slices: $(x, 0.5, \tilde{x}, 0.5)$, $(0.5, y, 0.5, \tilde{y})$, $(0.5, 0.5, \tilde{x}, \tilde{y})$ and display them respectively in Figures 5.20 to 5.22. These figures illustrate the different symmetries of the Green's matrix, which are captured by the rational NNs. As an example, we see in Figures 5.20 and 5.21 that $G_{1,1}(x, 0.5, \tilde{x}, 0.5) = G_{2,2}(0.5, x, 0.5, \tilde{x})$ and $G_{2,2}(x, 0.5, \tilde{x}, 0.5) = G_{1,1}(0.5, x, 0.5, \tilde{x})$, for $x, \tilde{x} \in [0, 1]$. Similarly, we find in Figure 5.22 that $G_{1,1}(0.5, 0.5, \tilde{x}, \tilde{y}) = G_{1,1}(0.5, 0.5, \tilde{y}, \tilde{x})$ and $G_{1,2}(0.5, 0.5, \tilde{x}, \tilde{y}) = G_{2,1}(0.5, 0.5, \tilde{x}, \tilde{y})$, for $\tilde{x}, \tilde{y} \in [0, 1]$. The $G_{1,2}$ and $G_{2,1}$ components of the Green's

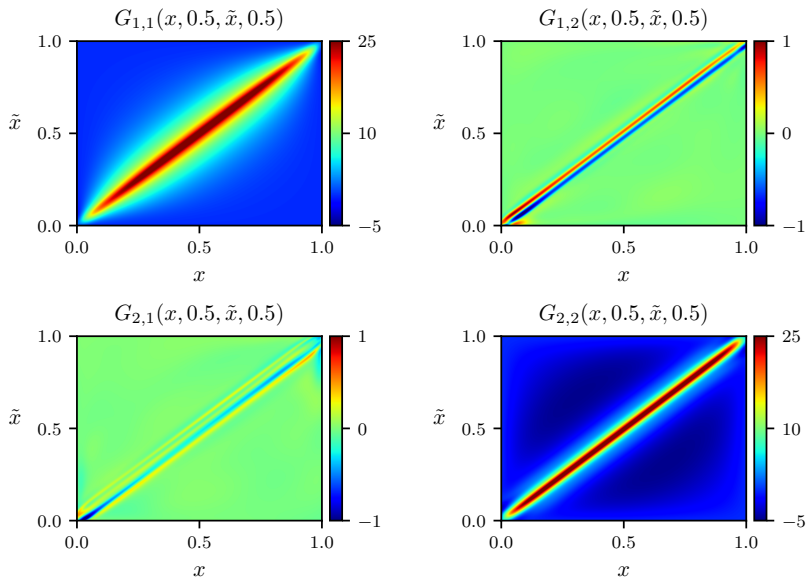


Figure 5.20: 2nd Green's matrix slice of Stokes flow. The four components of the Green's matrix learned by a rational neural network evaluated at the two-dimensional slice $(x, 0.5, \tilde{x}, 0.5)$.

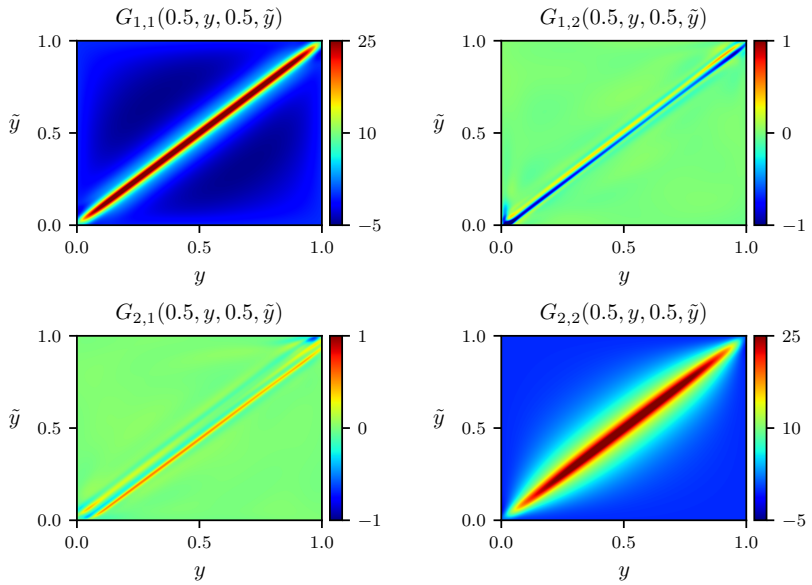


Figure 5.21: 3rd Green's matrix slice of Stokes flow. The four components of the Green's matrix learned by a rational neural network evaluated at the two-dimensional slice $(0.5, y, 0.5, \tilde{y})$.

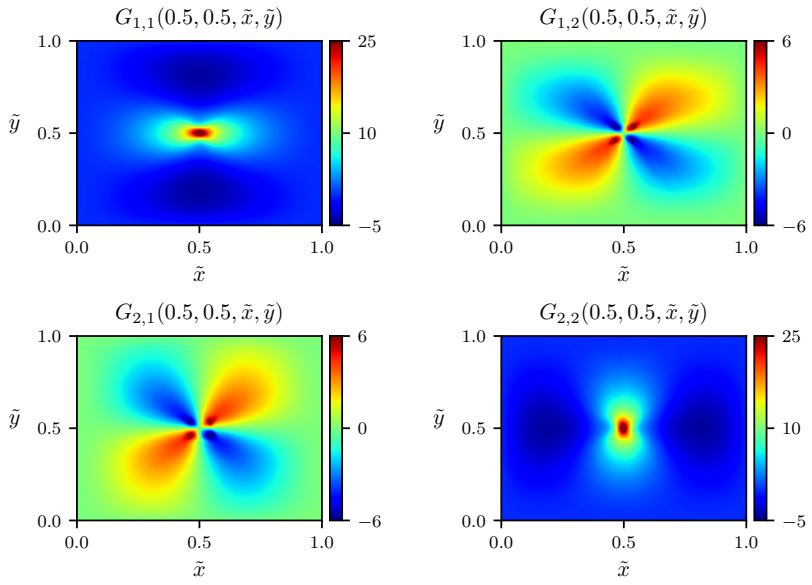


Figure 5.22: 4th Green's matrix slice of Stokes flow. The four components of the Green's matrix learned by a rational neural network evaluated at the two-dimensional slice $(0.5, 0.5, \tilde{x}, \tilde{y})$.

matrix in Figure 5.20 highlight a singularity along the diagonal $(x, 0.5, x, 0.5)$ for $x \in [0, 1]$. However, this singularity does not prevent the rational NNs from accurately learning the different components of the Green's matrix displayed in Figure 5.18(d) and Figures 5.20 to 5.22.

5.8 Time-dependent equations

In this section, we show that one can use a time-stepping scheme to discretize a time-dependent PDE and learn the Green's function associated with the time-propagator operator $\tau : u_n \to u_{n+1}$, where u_n is the solution of the PDE at time $t = n\Delta t$ for a fixed time step Δt. As an example, we consider the time-dependent Schrödinger equation with a harmonic trap potential $V(x) = x^2$ given by

$$\mathrm{i}\frac{\partial \psi(x,t)}{\partial t} = -\frac{1}{2}\frac{\partial^2 \psi(x,t)}{\partial x^2} + x^2\psi(x,t), \quad x \in [-3, 3], \tag{5.15}$$

with homogeneous Dirichlet boundary conditions. We use a Crank–Nicolson time-stepping scheme with time step $\Delta t = 2 \times 10^{-2}$ to discretize Equation (5.15) in time and obtain

$$\mathrm{i}\frac{\psi_{n+1} - \psi_n}{\Delta t} = \frac{1}{2}\left[-\frac{1}{2}\frac{d^2\psi_{n+1}}{dx^2} + x^2\psi_{n+1} - \frac{1}{2}\frac{d^2\psi_n}{dx^2} + x^2\psi_n\right].$$

Our training dataset consists of one hundred random initial forcing functions ψ_n at time t and associated response ψ_{n+1} at time $t + \Delta t$. The functions ψ_n have real and imaginary parts sampled from a Gaussian process with periodic kernel and length-scale parameter $\lambda = 0.5$ (see Section 5.2.1), and multiplied by the Gaussian damping function $g(x) = e^{-x^6/20}$ to ensure that the functions decay to zero before reaching the domain boundaries. We then train a rational neural network to learn the Green's function G associated with the time-propagator operator such that

$$\tau(\psi_n)(x) = \int_{-3}^{3} G(x, y)\psi_n(y)\,\mathrm{d}y = \psi_{n+1}(x), \quad x \in [-3, 3].$$

Note that since ψ takes complex values, we in fact split Equation (5.15) into a system of equations for the real and imaginary parts of ψ, and learn the Green's matrix associated with the system (see Section 5.6.2).

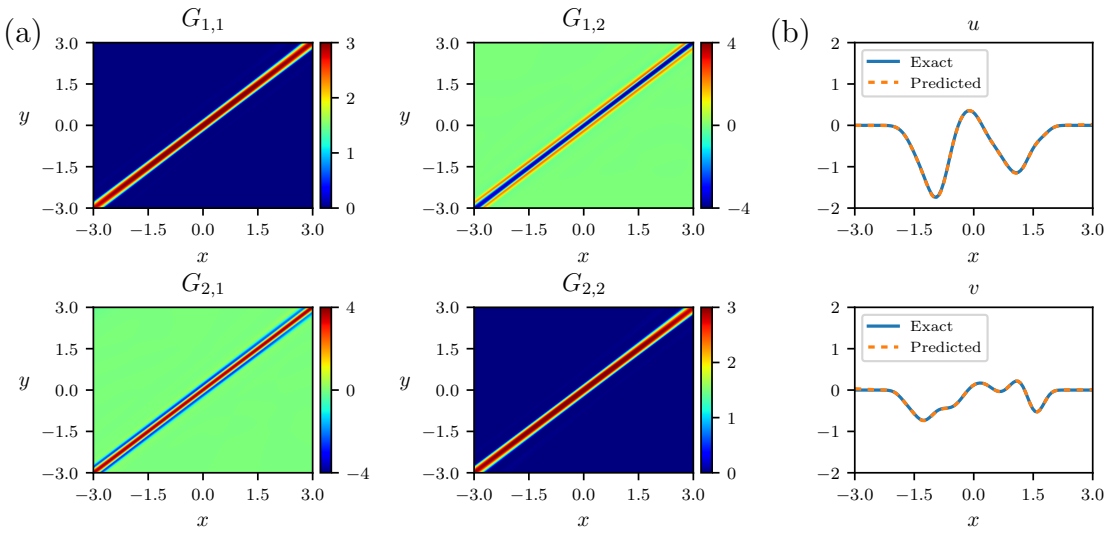


Figure 5.23: Green's matrix of the time-dependent Schrödinger equation. (a) The four components of the Green's matrix for the time propagator operator of the time-dependent Schrödinger equation discretized using a time-stepping scheme. (b) Real and imaginary components of the worst case prediction of the solution at the next time step.

We report the Green's matrix of the time-propagator operator for the Schrödinger equation in Figure 5.23(a) and observe that the four components are dominated by the diagonal, which is expected for a small time-step. Additionally, we evaluate the accuracy of the learned Green's functions by generating a testing dataset with one hundred initial functions ψ_n, sampled from the same distribution, and associated solution ψ_{n+1} at time $t + \Delta t$. We then compute the average (over the one hundred

test cases) relative error in the L^2 norm between the exact solution ψ_{n+1} and the one predicted using the learned Green's functions, ψ_{n+1}^{pred}, as

$$\text{relative error} = \|\psi_{n+1} - \psi_{n+1}^{\text{pred}}\|_{L^2([-3,3])} / \|\psi_{n+1}\|_{L^2([-3,3])},$$

where ψ_{n+1}^{pred} is defined as

$$\psi_{n+1}^{\text{pred}}(x) = \int_{-3}^{3} G(x,y)\psi_n(y)\,\mathrm{d}y, \quad x \in [-3,3].$$

Finally, we obtain an average relative error of 1.3% with standard deviation 0.2% across the 100 test cases, confirming the good accuracy of our method. We display the worst-case prediction of the solution ψ_{n+1} in Figure 5.23(b).

Conclusions

This thesis derived theoretical results and a practical deep learning algorithm for approximating Green's functions associated with linear partial differential equations (PDEs) from pairs of forcing terms and solutions to a PDE.

By generalizing the randomized singular value decomposition (SVD) to Hilbert–Schmidt (HS) operators in Chapter 2, we showed that one can rigorously learn the Green's function associated with an elliptic PDE in three dimensions. We derived a learning rate associated with elliptic partial differential operators in three dimensions and bounded the number of input-output training pairs required to recover a Green's function approximately with high probability. The random forcing functions are sampled from a Gaussian process (GP) with mean zero and are characterized by the associated covariance kernel. One practical outcome of this work is a measure for the quality of covariance kernels, which may be used to design efficient GP kernels for PDE learning tasks.

We then explored the practical extensions of the randomized SVD to Gaussian random vectors with correlated entries (*i.e.*, nonstandard covariance matrices) and HS operators in Chapter 3. This chapter motivates new computational and algorithmic approaches for constructing the covariance kernel based on prior information to compute a low-rank approximation of matrices and impose properties on the learned matrix and random functions from the GP. We performed numerical experiments to demonstrate that covariance matrices with prior knowledge can outperform the standard identity matrix used in the literature and lead to near-optimal approximation errors. In addition, we proposed a covariance kernel based on weighted Jacobi polynomials, which allows the control of the smoothness of the random functions generated and may find practical applications in PDE learning [27, 30] as it imposes prior knowledge of Dirichlet boundary conditions. The algorithm presented in this chapter is limited to matrices and HS operators and does not extend to unbounded operators such as differential operators. Additionally, the theoretical bounds only offer probabilistic guarantees for Gaussian inputs, while sub-Gaussian distributions [98]

of the inputs would be closer to realistic application settings.

Motivated by the theoretical results obtained in Chapters 2 and 3, we wanted to design an efficient deep learning architecture for learning Green's functions. In Chapter 4, we investigated rational neural networks, which are neural networks with smooth trainable activation functions based on rational functions. We proved theoretical statements quantifying the advantages of rational neural networks over ReLU networks. In particular, we remarked that a composition of low-degree rational functions has a good approximation power but a relatively small number of trainable parameters. Therefore, we showed that rational neural networks require fewer nodes and exponentially smaller depth than ReLU networks to approximate smooth functions to within a certain accuracy. This improved approximation power has practical consequences for large neural networks, given that a deep neural network is computationally expensive to train due to expensive gradient evaluations and slower convergence. The experiments conducted in the chapter demonstrate the potential applications of these rational networks for solving PDEs and generative adversarial networks. The practical implementation of rational networks is straightforward in the TensorFlow framework and consists of replacing the activation functions by trainable rational functions. The main benefits of rational NNs are their fast approximation power, the trainability of the activation parameters, and the smoothness of the activation function outside poles.

Our primary objective in Chapter 5 was to uncover mechanistic understanding from input-output data using a human-understandable representation of an underlying hidden differential operator. This representation took the form of a rational NN for the Green's function. We extensively described all the physical features of the operator that can be extracted and discovered from the learned Green's function and homogeneous solutions, such as linear conservation laws, symmetries, shock front and singularity locations, boundary conditions, and dominant modes. Our deep learning method for learning Green's functions and extracting human-understandable properties of partial differential equations benefits from the adaptivity of rational neural networks and its support for qualitative feature detection and interpretation. We successfully tested our approach with noisy and sparse measurements as training data in one and two dimensions. The design of our network architecture, and the covariance kernel used to generate the system forcings, are guided by rigorous theoretical statements, obtained in Chapters 2 to 4, that offer performance guarantees. This shows that our proposed deep learning method may be used to discover new mechanistic understanding with machine learning.

The deep learning method naturally extends to the case of three spatial dimensions but these systems are more challenging due to the GPU memory demands required to represent the six-dimensional inputs used to train the neural network representing the Green's function. However, alternative optimization algorithms than the one we used, such as mini-batch optimization [108, 125], may be employed to alleviate the computational expense of the training procedure. While our method is demonstrated on linear differential operators, it can be extended to nonlinear, time-dependent problems that can be linearized using an implicit-explicit time-stepping scheme [12, 171] or an iterative method [105]. This process should allow us to learn Green's functions of linear time propagators and understand physical behavior in time-dependent problems from input-output data such as the time-dependent Schrödinger equation. The numerical experiments conducted in Chapter 5 highlight that our approach can generalized to discover Green's functions of some linearization of a nonlinear differential operator.

There are many future research directions exploring the potential applications of rational networks beyond PDE learning, in fields such as image classification, time series forecasting, and generative adversarial networks. These applications already employ nonstandard activation functions to overcome various drawbacks of ReLU. Another exciting and promising field is the numerical solution and data-driven discovery of partial differential equations with deep learning. We believe that popular techniques such as physics-informed neural networks [184] could benefit from rational NNs to improve the robustness and performance of PDE solvers, both from a theoretical and practical viewpoint.

Finally, while the ideas present in Chapter 2 have been recently applied to derive a learning rate for Green's functions associated with parabolic PDEs [28], obtaining theoretical results for more general classes of PDEs, such as hyperbolic, fractional, or stochastic PDEs, remain highly challenging. Such studies are essential to understand which mathematical models can be learned from data, obtain performance guarantees, and, more generally, deepen our knowledge of PDE learning techniques.

Bibliography

[1] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, et al., *TensorFlow: A System for Large-Scale Machine Learning*, in 12th USENIX Conference on Operating Systems Design and Implementation, 2016, pp. 265–283.

[2] H. Abdi and L. J. Williams, *Principal component analysis*, Wiley Interdiscip. Rev. Comput. Stat., 2 (2010), pp. 433–459.

[3] N. I. Achieser, *Theory of Approximation*, Courier Corporation, 2013.

[4] N. Ailon and B. Chazelle, *Approximate nearest neighbors and the fast Johnson–Lindenstrauss transform*, in Proceedings of the Thirty-Eighth Annual ACM Symposium on Theory of Computing, 2006, pp. 557–563.

[5] ——, *The fast Johnson–Lindenstrauss transform and approximate nearest neighbors*, SIAM J. Comput., 39 (2009), pp. 302–322.

[6] R. Alexander and D. Giannakis, *Operator-theoretic framework for forecasting nonlinear time series with kernel analog techniques*, Physica D, 409 (2020), p. 132520.

[7] O. Alter, P. O. Brown, and D. Botstein, *Singular value decomposition for genome-wide expression data processing and modeling*, Proc. Natl. Acad. Sci. USA, 97 (2000), pp. 10101–10106.

[8] M. Anthony and P. Bartlett, *Neural network learning: Theoretical foundations*, Cambridge University Press, 1999.

[9] G. Arfken, H. Weber, and F. E. Harris, *Mathematical Methods for Physicists*, Academic Press, 7th ed., 2012.

[10] M. Arjovsky, S. Chintala, and L. Bottou, *Wasserstein Generative Adversarial Networks*, in Proc. 34th International Conference on Machine Learning (ICML), 2017, pp. 214–223.

[11] S. Arridge, P. Maass, O. Öktem, and C.-B. Schönlieb, *Solving inverse problems using data-driven models*, Acta Numer., 28 (2019), pp. 1–174.

[12] U. M. Ascher, S. J. Ruuth, and R. J. Spiteri, *Implicit-explicit Runge-Kutta methods for time-dependent partial differential equations*, Appl. Numer. Math., 25 (1997), pp. 151–167.

[13] J. Ballani and D. Kressner, *Matrices with hierarchical low-rank structures*, in Exploiting Hidden Structure in Matrix Computations: Algorithms and Applications, Springer, 2016, pp. 161–209.

[14] A. G. Baydin, B. A. Pearlmutter, A. A. Radul, and J. M. Siskind, *Automatic differentiation in machine learning: a survey*, J. Mach. Learn. Res., 18 (2018), pp. 1–43.

[15] M. Bebendorf, *Hierarchical Matrices: A Means to Efficiently Solve Elliptic Boundary Value Problems*, Springer-Verlag, 2008.

[16] M. Bebendorf and W. Hackbusch, *Existence of $\mathscr{H}$-matrix approximants to the inverse FE-matrix of elliptic operators with L^∞-coefficients*, Numer. Math., 95 (2003), pp. 1–28.

[17] B. Beckermann and A. Townsend, *On the Singular Values of Matrices with Displacement Structure*, SIAM J. Matrix Anal. A., 38 (2017), pp. 1227–1248.

[18] Y. Bengio, *Practical recommendations for gradient-based training of deep architectures*, in Neural networks: Tricks of the trade, Springer, 2012, pp. 437–478.

[19] Y. Bengio, P. Simard, and P. Frasconi, *Learning Long-Term Dependencies with Gradient Descent is Difficult*, IEEE T. Neural Netw., 5 (1994), pp. 157–166.

[20] W. T. Beyene, *Pole-clustering and rational-interpolation techniques for simplifying distributed systems*, IEEE T. Circuits-I, 46 (1999), pp. 1468–1472.

[21] K. Binder, D. M. Ceperley, J.-P. Hansen, M. Kalos, D. Landau, D. Levesque, H. Mueller-Krumbhaar, D. Stauffer, and J.-J. Weis, *Monte Carlo Methods in Statistical Physics*, Springer Science & Business Media, 2012.

[22] J. R. Blake, *A note on the image system for a Stokeslet in a no-slip boundary*, Math. Proc. Camb. Philos. Soc., 70 (1971), pp. 303–310.

[23] D. Boffi, F. Brezzi, and M. Fortin, *Mixed Finite Element Methods and Applications*, Springer, 2013.

[24] A. Bonito, A. Cohen, R. DeVore, G. Petrova, and G. Welper, *Diffusion coefficients estimation for elliptic partial differential equations*, SIAM J. Math. Anal., 49 (2017), pp. 1570–1592.

[25] W. Boukaram, G. Turkiyyah, and D. Keyes, *Randomized GPU algorithms for the construction of hierarchical matrices from matrix-vector operations*, SIAM J. Sci. Comput., 41 (2019), pp. C339–C366.

[26] N. Boullé, *NBoulle/GreenLearning - Software and datasets (version v1.0). Zenodo.* `https://doi.org/10.5281/zenodo.4656020`, 2021.

[27] N. Boullé, C. J. Earls, and A. Townsend, *Data-driven discovery of Green's functions with human-understandable deep learning*, Sci. Rep., 12 (2022).

[28] N. Boullé, S. Kim, T. Shi, and A. Townsend, *Learning Green's functions associated with time-dependent partial differential equations*, J. Mach. Learn. Res., 23 (2022), pp. 1–34.

[29] N. Boullé, Y. Nakatsukasa, and A. Townsend, *GitHub repository.* `https://github.com/NBoulle/RationalNets/`, 2020.

[30] N. Boullé, Y. Nakatsukasa, and A. Townsend, *Rational neural networks*, in Advances in Neural Information Processing Systems (NeurIPS), vol. 33, 2020, pp. 14243–14253.

[31] N. Boullé and A. Townsend, *A generalization of the randomized singular value decomposition*, in International Conference on Learning Representations (ICLR), 2022.

[32] N. Boullé and A. Townsend, *Learning elliptic partial differential equations with randomized linear algebra*, Found. Comput. Math., (2022).

[33] H. Brezis, *Functional Analysis, Sobolev Spaces and Partial Differential Equations*, Springer Science & Business Media, 2010.

[34] S. L. Brunton, M. Budišić, E. Kaiser, and J. N. Kutz, *Modern Koopman theory for dynamical systems*, arXiv preprint arXiv:2102.12086, (2021).

[35] S. L. Brunton, B. R. Noack, and P. Koumoutsakos, *Machine Learning for Fluid Mechanics*, Annu. Rev. Fluid Mech., 52 (2020), pp. 477–508.

[36] S. L. Brunton, J. L. Proctor, and J. N. Kutz, *Discovering governing equations from data by sparse identification of nonlinear dynamical systems*, Proc. Natl. Acad. Sci. USA, 113 (2016).

[37] R. H. Byrd, P. Lu, J. Nocedal, and C. Zhu, *A limited memory algorithm for bound constrained optimization*, SIAM J. Sci. Comput., 16 (1995), pp. 1190–1208.

[38] T. Chen and H. Chen, *Universal approximation to nonlinear operators by neural networks with arbitrary activation functions and its application to dynamical systems*, IEEE Trans. Neur. Netw., 6 (1995), pp. 911–917.

[39] Z. Chen, F. Chen, R. Lai, X. Zhang, and C.-T. Lu, *Rational Neural Networks for Approximating Graph Convolution Operator on Jump Discontinuities*, in IEEE International Conference on Data Mining (ICDM), 2018, pp. 59–68.

[40] X. Cheng, B. Khomtchouk, N. Matloff, and P. Mohanty, *Polynomial Regression As an Alternative to Neural Nets*, arXiv preprint arXiv:1806.06850, (2018).

[41] H. Chernoff, *A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations*, Ann. Math. Stat., (1952), pp. 493–507.

[42] S. Cho, H. Dong, and S. Kim, *Global estimates for Green's matrix of second order parabolic systems with application to elliptic systems in two dimensional domains*, Potential Anal., 36 (2012), pp. 339–372.

[43] F. Chollet et al., *Keras*. `https://keras.io`, 2015.

[44] K. L. Clarkson and D. P. Woodruff, *Low-rank approximation and regression in input sparsity time*, J. ACM, 63 (2017), pp. 1–45.

[45] D.-A. Clevert, T. Unterthiner, and S. Hochreiter, *Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs)*, arXiv preprint arXiv:1511.07289, (2015).

[46] G. Cybenko, *Approximation by superpositions of a sigmoidal function*, Math. Control Signals Syst., 2 (1989), pp. 303–314.

[47] J. Daws Jr. and C. G. Webster, *A Polynomial-Based Approach for Architectural Design and Learning with Deep Neural Networks*, arXiv preprint arXiv:1905.10457, (2019).

[48] C. de Boor, *An alternative approach to (the teaching of) rank, basis, and dimension*, Lin. Alg. Appl., 146 (1991), pp. 221–229.

[49] M. V. de Hoop, N. B. Kovachki, N. H. Nelsen, and A. M. Stuart, *Convergence rates for learning linear operators from noisy data*, arXiv preprint arXiv:2108.12515, (2021).

[50] P. Deheuvels and G. V. Martynov, *A Karhunen–Loeve decomposition of a Gaussian process generated by independent pairs of exponential random variables*, J. Funct. Anal., 255 (2008), pp. 2363–2394.

[51] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, *Imagenet: A large-scale hierarchical image database*, in IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2009, pp. 248–255.

[52] R. A. DeVore, R. Howard, and C. Micchelli, *Optimal nonlinear approximation*, Manuscripta Math., 63 (1989), pp. 469–478.

[53] H. Dong and S. Kim, *Green's matrices of second order elliptic systems with measurable coefficients in two dimensional domains*, Trans. Am. Math. Soc., 361 (2009), pp. 3303–3323.

[54] ——, *Green's function for nondivergence elliptic operators in two dimensions*, SIAM J. Math. Anal., 53 (2021), pp. 4637–4656.

[55] M. F. Driscoll, *The reproducing kernel Hilbert space structure of the sample paths of a Gaussian process*, Zeit. Wahrscheinlichkeitstheorie Verwandte Geb., 26 (1973), pp. 309–316.

[56] T. A. Driscoll, N. Hale, and L. N. Trefethen, *Chebfun Guide*, Pafnuty Publications, 2014.

[57] R. Durrett, *Probability: Theory and Examples*, Cambridge University Press, 5th ed., 2019.

[58] C. Eckart and G. Young, *The approximation of one matrix by another of lower rank*, Psychometrika, 1 (1936), pp. 211–218.

[59] D. E. Edmunds and W. D. Evans, *Spectral theory and differential operators*, Oxford University Press, 2018.

[60] D. E. Edmunds, V. M. Kokilashvili, and A. Meskhi, *Bounded and compact integral operators*, Springer Science & Business Media, 2013.

[61] M. Ekiel-Jeżewska, R. Boniecki, M. Bukowicki, and M. Gruca, *Stokes velocity generated by a point force in various geometries*, Eur. Phys. J. E, 41 (2018), pp. 1–7.

[62] H. C. Elman, D. J. Silvester, and A. J. Wathen, *Finite Elements and Fast Iterative Solvers: With Applications in Incompressible Fluid Dynamics*, Oxford University Press, 2nd ed., 2014.

[63] B. Engquist and H. Zhao, *Approximate separability of the Green's function of the Helmholtz equation in the high frequency limit*, Comm. Pure Appl. Math., 71 (2018), pp. 2220–2274.

[64] L. C. Evans, *Partial Differential Equations*, AMS, Providence, RI, 2nd ed., 2010.

[65] J. Feliu-Faba, Y. Fan, and L. Ying, *Meta-learning pseudo-differential operators with deep neural networks*, J. Comput. Phys., 408 (2020).

[66] S. Filip, A. Javeed, and L. N. Trefethen, *Smooth random functions, random ODEs, and Gaussian processes*, SIAM Rev., 61 (2019), pp. 185–205.

[67] S.-I. Filip, Y. Nakatsukasa, L. N. Trefethen, and B. Beckermann, *Rational Minimax Approximation via Adaptive Barycentric Representations*, SIAM J. Sci. Comput., 40 (2018), pp. A2427–A2455.

[68] J. Foster, T. Lyons, and H. Oberhauser, *An optimal polynomial approximation of Brownian motion*, SIAM J. Numer. Anal., 58 (2020), pp. 1393–1421.

[69] A. P. George and W. B. Powell, *Adaptive stepsizes for recursive estimation with applications in approximate dynamic programming*, Mach. Learn., 65 (2006), pp. 167–198.

[70] C. Geuzaine and J.-F. Remacle, *Gmsh: A 3-D finite element mesh generator with built-in pre-and post-processing facilities*, Int. J. Numer. Methods Eng., 79 (2009), pp. 1309–1331.

[71] C. R. Gin, D. E. Shea, S. L. Brunton, and J. N. Kutz, *DeepGreen: deep learning of Green's functions for nonlinear boundary value problems*, Sci. Rep., 11 (2021), pp. 1–14.

[72] X. Glorot and Y. Bengio, *Understanding the difficulty of training deep feedforward neural networks*, in Proc. 13th International Conference on Artificial Intelligence and Statistics (AISTATS), 2010, pp. 249–256.

[73] X. Glorot, A. Bordes, and Y. Bengio, *Deep Sparse Rectifier Neural Networks*, in Proc. 14th International Conference on Artificial Intelligence and Statistics (AISTATS), 2011, pp. 315–323.

[74] G. H. Golub and C. F. Van Loan, *Matrix Computations*, JHU Press, 4th ed., 2013.

[75] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, MIT Press, 2016.

[76] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, *Generative Adversarial Nets*, in Advances in Neural Information Processing Systems (NeurIPS), 2014, pp. 2672–2680.

[77] M. Goyal, R. Goyal, and B. Lall, *Improved polynomial neural networks with normalised activations*, in International Joint Conference on Neural Networks (IJCNN), 2020, pp. 1–8.

[78] A. Graves and N. Jaitly, *Towards end-to-end speech recognition with recurrent neural networks*, in International Conference on Machine Learning (ICML), PMLR, 2014, pp. 1764–1772.

[79] G. Green, *An essay on the application of mathematical analysis to the theories of electricity and magnetism*, J. für die Reine und Angew. Math., 47 (1854), pp. 161–221.

[80] M. Grüter and K.-O. Widman, *The Green function for uniformly elliptic equations*, Manuscripta Math., 37 (1982), pp. 303–342.

[81] S. Guarnieri, F. Piazza, and A. Uncini, *Multilayer feedforward networks with adaptive spline activation function*, IEEE Trans. Neural Networ., 10 (1999), pp. 672–683.

[82] I. Gühring, G. Kutyniok, and P. Petersen, *Error bounds for approximations with deep ReLU neural networks in $W^{s,p}$ norms*, Anal. Appl., 18 (2020), pp. 803–859.

[83] K. Habermann, *A semicircle law and decorrelation phenomena for iterated Kolmogorov loops*, J. London Math. Soc., (2019).

[84] W. Hackbusch, *Hierarchical Matrices: Algorithms and Analysis*, Springer, 2015.

[85] E. Haghighat, M. Raissi, A. Moure, H. Gomez, and R. Juanes, *A physics-informed deep learning framework for inversion and surrogate modeling in solid mechanics*, Comput. Methods Appl. Mech. Eng., 379 (2021).

[86] N. Halko, P.-G. Martinsson, and J. A. Tropp, *Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions*, SIAM Rev., 53 (2011), pp. 217–288.

[87] K. He, X. Zhang, S. Ren, and J. Sun, *Delving deep into rectifiers: Surpassing human-level performance on imagenet classification*, in Proc. IEEE International Conference on Computer Vision (ICCV), 2015, pp. 1026–1034.

[88] C. F. Higham and D. J. Higham, *Deep learning: An introduction for applied mathematicians*, SIAM Rev., 61 (2019), pp. 860–891.

[89] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, et al., *Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups*, IEEE Signal Process. Mag., 29 (2012), pp. 82–97.

[90] S. Hochreiter and J. Schmidhuber, *Long short-term memory*, Neural Comput., 9 (1997), pp. 1735–1780.

[91] S. Hofmann and S. Kim, *The Green function estimates for strongly elliptic systems of second order*, Manuscripta Math., 124 (2007), pp. 139–172.

[92] H. Hotelling, *Analysis of a complex of statistical variables into principal components.*, J. Educ. Psychol., 24 (1933), p. 417.

[93] T. Hsing and R. Eubank, *Theoretical foundations of functional data analysis, with an introduction to linear operators*, John Wiley & Sons, 2015.

[94] S. Hwang and S. Kim, *Green's function for second order elliptic equations in non-divergence form*, Potential Anal., 52 (2020), pp. 27–39.

[95] S. Ioffe and C. Szegedy, *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift*, in Proc. 32nd International Conference on Machine Learning (ICML), 2015, pp. 448–456.

[96] A. D. Jagtap, K. Kawaguchi, and G. E. Karniadakis, *Adaptive activation functions accelerate convergence in deep and physics-informed neural networks*, J. Comput. Phys., 404 (2020).

[97] K. Jarrett, K. Kavukcuoglu, M. Ranzato, and Y. LeCun, *What is the best multi-stage architecture for object recognition?*, in Proc. IEEE International Conference on Computer Vision (ICCV), 2009, pp. 2146–2153.

[98] J.-P. Kahane, *Propriétés locales des fonctions à séries de Fourier aléatoires*, Stud. Math., 19 (1960), pp. 1–25.

[99] M. Kanagawa, P. Hennig, D. Sejdinovic, and B. K. Sriperumbudur, *Gaussian processes and kernel methods: A review on connections and equivalences*, arXiv preprint arXiv:1807.02582, (2018).

[100] K. Kang and S. Kim, *Global pointwise estimates for Green's matrix of second order elliptic systems*, J. Differ. Equ., 249 (2010), pp. 2643–2662.

[101] K. Karhunen, *Über lineare methoden in der wahrscheinlichkeitsrechnung*, Ann. Acad. Science Fenn., Ser. A. I., 37 (1946), pp. 3–79.

[102] G. E. Karniadakis, I. G. Kevrekidis, L. Lu, P. Perdikaris, S. Wang, and L. Yang, *Physics-informed machine learning*, Nat. Rev. Phys., 3 (2021), pp. 422–440.

[103] T. Karras, S. Laine, and T. Aila, *A style-based generator architecture for generative adversarial networks*, in Conference on Computer Vision and Pattern Recognition (CVPR), 2019, pp. 4401–4410.

[104] T. Kato, *Perturbation Theory for Linear Operators*, Springer Science & Business Media, 2013.

[105] C. T. Kelley, *Iterative Methods for Linear and Nonlinear Equations*, SIAM, 1995.

[106] S. Kim and G. Sakellaris, *Green's function for second order elliptic equations with singular lower order coefficients*, Commun. Partial. Differ. Equ., 44 (2019), pp. 228–270.

[107] S. Kim and L. Xu, *Green's function for second order parabolic equations with singular lower order coefficients*, Commun. Pure Appl. Anal., 21 (2022), pp. 1–21.

[108] D. P. Kingma and J. Ba, *Adam: A method for stochastic optimization*, in International Conference on Learning Representations (ICLR), 2015.

[109] G. Klambauer, T. Unterthiner, A. Mayr, and S. Hochreiter, *Self-Normalizing Neural Networks*, in Advances in Neural Information Processing Systems (NeurIPS), 2017, pp. 971–980.

[110] B. O. Koopman, *Hamiltonian systems and transformation in Hilbert space*, Proc. Natl. Acad. Sci.s, 17 (1931), pp. 315–318.

[111] N. Kovachki, S. Lanthaler, and S. Mishra, *On universal approximation and error bounds for Fourier Neural Operators*, J. Mach. Learn. Res., 22 (2021), pp. 1–76.

[112] N. Kovachki, Z. Li, B. Liu, K. Azizzadenesheli, K. Bhattacharya, A. Stuart, and A. Anandkumar, *Neural operator: Learning maps between function spaces*, arXiv preprint arXiv:2108.08481, (2021).

[113] E. Kreyszig, *Introductory Functional Analysis with Applications*, Wiley, 1978.

[114] A. Krizhevsky, I. Sutskever, and G. E. Hinton, *ImageNet Classification with Deep Convolutional Neural Networks*, in Advances in Neural Information Processing Systems (NeurIPS), 2012, pp. 1097–1105.

[115] H. W. Kuhn and A. W. Tucker, *Nonlinear Programming*, in Proc. Second Berkeley Symp. on Math. Statist. and Prob., Univ. of Calif. Press, 1951, pp. 481–492.

[116] S. Lanthaler, S. Mishra, and G. E. Karniadakis, *Error estimates for DeepONets: A deep learning framework in infinite dimensions*, Trans. Math. Appl., 6 (2022).

[117] V. I. Lebedev, *On a Zolotarev problem in the method of alternating directions*, USSR Comp. Math. Math+, 17 (1977), pp. 58–76.

[118] Y. LeCun, Y. Bengio, and G. Hinton, *Deep learning*, Nature, 521 (2015), pp. 436–444.

[119] Y. LeCun, B. Boser, J. Denker, D. Henderson, R. Howard, W. Hubbard, and L. Jackel, *Handwritten digit recognition with a back-propagation network*, Advances in Neural Information Processing Systems (NeurIPS), 2 (1989).

[120] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, *Gradient-based learning applied to document recognition*, Proc. IEEE, 86 (1998), pp. 2278–2324.

[121] M. Ledoux, *The concentration of measure phenomenon*, Math. Surveys. Monog. 89, AMS, Providence, RI, 2001.

[122] J.-Y. Lee and L. Greengard, *A fast adaptive numerical method for stiff two-point boundary value problems*, SIAM J. Sci. Comput., 18 (1997), pp. 403–429.

[123] F. C. Leone, L. S. Nelson, and R. B. Nottingham, *The folded normal distribution*, Technometrics, 3 (1961), pp. 543–550.

[124] D. Li, K. Xu, J. M. Harris, and E. Darve, *Coupled time-lapse full-waveform inversion for subsurface flow problems using intrusive automatic differentiation*, Water Resour. Res., 56 (2020).

[125] M. Li, T. Zhang, Y. Chen, and A. J. Smola, *Efficient mini-batch training for stochastic optimization*, in Proc. 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2014, pp. 661–670.

[126] Z. Li, N. Kovachki, K. Azizzadenesheli, B. Liu, K. Bhattacharya, A. Stuart, and A. Anandkumar, *Neural operator: Graph kernel network for partial differential equations*, arXiv preprint arXiv:2003.03485, (2020).

[127] ——, *Fourier neural operator for parametric partial differential equations*, in International Conference on Learning Representations (ICLR), 2021.

[128] Z. Li, N. Kovachki, K. Azizzadenesheli, B. Liu, A. Stuart, K. Bhattacharya, and A. Anandkumar, *Multipole graph neural operator for parametric partial differential equations*, in Advances in Neural Information Processing Systems (NeurIPS), vol. 33, 2020, pp. 6755–6766.

[129] S. Liang and R. Srikant, *Why Deep Neural Networks for Function Approximation?*, in International Conference on Learning Representations (ICLR), 2017.

[130] L. Lin, J. Lu, and L. Ying, *Fast construction of hierarchical matrix representation from matrix–vector multiplication*, J. Comput. Phys., 230 (2011), pp. 4071–4087.

[131] F. Lindgren, H. Rue, and J. Lindström, *An explicit link between Gaussian fields and Gaussian Markov random fields: the stochastic partial differential equation approach*, J. R. Stat. Soc. B, 73 (2011), pp. 423–498.

[132] D. C. Liu and J. Nocedal, *On the limited memory BFGS method for large scale optimization*, Math. Program., 45 (1989), pp. 503–528.

[133] M. Loève, *Fonctions aleatoire de second ordre*, Rev. Sci., 84 (1946), pp. 195–206.

[134] Z. Long, Y. Lu, X. Ma, and B. Dong, *PDE-NET: Learning PDEs from data*, in International Conference on Machine Learning (ICML), 2018, pp. 3208–3216.

[135] L. Lu, P. Jin, G. Pang, Z. Zhang, and G. E. Karniadakis, *Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators*, Nat. Mach. Intell., 3 (2021), pp. 218–229.

[136] L. Lu, X. Meng, Z. Mao, and G. E. Karniadakis, *DeepXDE: A deep learning library for solving differential equations*, SIAM Rev., 63 (2021), pp. 208–228.

[137] T.-T. Lu and S.-H. Shiou, *Inverses of 2× 2 block matrices*, Comput. Math. Appl., 43 (2002), pp. 119–129.

[138] J. Ma, R. P. Sheridan, A. Liaw, G. E. Dahl, and V. Svetnik, *Deep neural nets as a method for quantitative structure–activity relationships*, J. Chem. Inf. Model., 55 (2015), pp. 263–274.

[139] L. Ma and K. Khorasani, *Constructive feedforward neural networks using Hermite polynomial activation functions*, IEEE Trans. Neural Networ., 16 (2005), pp. 821–833.

[140] A. L. Maas, A. Y. Hannun, and A. Y. Ng, *Rectifier Nonlinearities Improve Neural Network Acoustic Models*, in International Conference on Machine Learning (ICML), 2013.

[141] S. Maddu, B. L. Cheeseman, I. F. Sbalzarini, and C. L. Müller, *Stability selection enables robust learning of differential equations from limited noisy data*, Proc. R. Soc. A, 478 (2022).

[142] A. A. Markov, *On a question by D. I. Mendeleev*, Zapiski Imp. Akad. Nauk, 62 (1889), pp. 1–24.

[143] P.-G. Martinsson, *A fast randomized algorithm for computing a hierarchically semiseparable representation of a matrix*, SIAM J. Matrix Anal. Appl., 32 (2011), pp. 1251–1274.

[144] ——, *Compressing rank-structured matrices via randomized sampling*, SIAM J. Sci. Comput., 38 (2016), pp. A1959–A1986.

[145] P.-G. Martinsson and J. A. Tropp, *Randomized numerical linear algebra: Foundations and algorithms*, Acta Numer., 29 (2020), p. 403–572.

[146] X. Meng, Z. Li, D. Zhang, and G. E. Karniadakis, *PPINN: Parareal physics-informed neural network for time-dependent PDEs*, Comput. Methods Appl. Mech. Eng., 370 (2020).

[147] X. Meng and M. W. Mahoney, *Low-distortion subspace embeddings in input-sparsity time and applications to robust linear regression*, in Proceedings of the Forty-Fifth Annual ACM Symposium on Theory of Computing, 2013, pp. 91–100.

[148] J. Mercer, *Functions of positive and negative type, and their connection the theory of integral equations*, Philos. T. R. Soc. A, 209 (1909), pp. 415–446.

[149] L. Metz, B. Poole, D. Pfau, and J. Sohl-Dickstein, *Unrolled Generative Adversarial Networks*, in International Conference on Learning Representations (ICLR), 2017.

[150] H. N. Mhaskar, *Neural Networks for Optimal Approximation of Smooth and Analytic Functions* , Neural Comput., 8 (1996), pp. 164–177.

[151] L. Mirsky, *Symmetric gauge functions and unitarily invariant norms*, Q. J. Math., 11 (1960), pp. 50–59.

[152] ——, *A trace inequality of John von Neumann*, Monatsh. Math., 79 (1975), pp. 303–306.

[153] A. Molina, P. Schramowski, and K. Kersting, *Padé Activation Units: End-to-end Learning of Flexible Activation Functions in Deep Networks*, in International Conference on Learning Representations (ICLR), 2019.

[154] H. Montanelli, H. Yang, and Q. Du, *Deep ReLU networks overcome the curse of dimensionality for generalized bandlimited functions*, J. Comput. Math., 39 (2021), pp. 801–815.

[155] A. M. Mood, F. A. Graybill, and D. C. Boes, *Introduction to the Theory of Statistics*, McGraw-Hill, 3rd ed., 1974.

[156] R. J. Muirhead, *Aspects of multivariate statistical theory*, John Wiley & Sons, 2009.

[157] T. Myint-U and L. Debnath, *Linear Partial Differential Equations for Scientists and Engineers*, Birkhäuser Basel, 2007.

[158] V. Nair and G. E. Hinton, *Rectified Linear Units Improve Restricted Boltzmann Machines*, in Proc. 27th International Conference on Machine Learning (ICML), 2010, pp. 807–814.

[159] Y. Nakatsukasa, *Fast and stable randomized low-rank matrix approximation*, arXiv preprint arXiv:2009.11392, (2020).

[160] Y. Nakatsukasa and R. W. Freund, *Computing fundamental matrix decompositions accurately via the matrix sign function in two iterations: The power of Zolotarev's functions*, SIAM Rev., 58 (2016), pp. 461–493.

[161] N. H. Nelsen and A. M. Stuart, *The random feature model for input-output maps between Banach spaces*, SIAM J. Sci. Comput., 43 (2021), pp. A3212–A3243.

[162] J. Nelson and H. L. Nguyên, *OSNAP: Faster Numerical Linear Algebra Algorithms via Sparser Subspace Embeddings*, in IEEE 54th Annual Symposium on Foundations of Computer Science, 2013, pp. 117–126.

[163] D. J. Newman, *Rational approximation to* $|x|$, Mich. Math. J., 11 (1964), pp. 11–14.

[164] E. J. Nyström, *Über die praktische Auflösung von integralgleichungen mit Anwendungen auf randwertaufgaben*, Acta Math., 54 (1930), pp. 185–204.

[165] A. Odena, C. Olah, and J. Shlens, *Conditional image synthesis with auxiliary classifier GANs*, in International Conference on Machine Learning (ICML), vol. 70, 2017, pp. 2642–2651.

[166] F. W. J. Olver, D. W. Lozier, R. F. Boisvert, and C. W. Clark, *NIST Handbook of Mathematical Functions* , Cambridge University Press, 2010.

[167] P. J. Olver, *Applications of Lie groups to differential equations*, Springer-Verlag, 2nd ed., 1993.

[168] R. Pachón and L. N. Trefethen, *Barycentric-Remez algorithms for best polynomial approximation in the chebfun system*, BIT, 49 (2009), pp. 721–741.

[169] G. Pang, L. Lu, and G. E. Karniadakis, *fPINNs: Fractional physics-informed neural networks*, SIAM J. Sci. Comput., 41 (2019), pp. A2603–A2626.

[170] G. Pang, L. Yang, and G. E. Karniadakis, *Neural-net-induced Gaussian process regression for function approximation and PDE solution*, J. Comput. Phys., 384 (2019), pp. 270–288.

[171] L. Pareschi and G. Russo, *Implicit–explicit runge–kutta schemes and applications to hyperbolic systems with relaxation*, J. Sci. Comput., 25 (2005), pp. 129–155.

[172] D. S. Parker, *Random Butterfly Transformations with Applications in Computational Linear Algebra*, Tech. Rep. CSD-950023, UCLA, 1995.

[173] A. Paterek, *Improving regularized singular value decomposition for collaborative filtering*, in Proc. KDD cup and workshop, 2007, pp. 5–8.

[174] K. Pearson, *On lines and planes of closest fit to systems of points in space*, Lond .Edinb. Phil. Mag., 2 (1901), pp. 559–572.

[175] P. Petersen and F. Voigtlaender, *Optimal approximation of piecewise smooth functions using deep ReLU neural networks*, Neural Netw., 108 (2018), pp. 296–330.

[176] P. P. Petrushev and V. A. Popov, *Rational Approximation of Real Functions*, Cambridge University Press, 2011.

[177] E. Qian, I.-G. Farcas, and K. Willcox, *Reduced operator inference for nonlinear partial differential equations*, SIAM J. Sci. Comput., 44 (2022), pp. A1934–A1959.

[178] E. Qian, B. Kramer, B. Peherstorfer, and K. Willcox, *Lift & learn: Physics-informed machine learning for large-scale nonlinear dynamical systems*, Phys. D, 406 (2020).

[179] A. Radford, L. Metz, and S. Chintala, *Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks*, in International Conference on Learning Representations (ICLR), 2016.

[180] M. Raissi, *Deep hidden physics models: Deep learning of nonlinear partial differential equations*, J. Mach. Learn. Res., 19 (2018), pp. 932–955.

[181] ——, *GitHub repository.* `https://github.com/maziarraissi/DeepHPMs/`, 2020.

[182] M. Raissi and G. E. Karniadakis, *Hidden physics models: Machine learning of nonlinear partial differential equations*, J. Comput. Phys., 357 (2018), pp. 125–141.

[183] M. Raissi, P. Perdikaris, and G. E. Karniadakis, *Multistep neural networks for data-driven discovery of nonlinear dynamical systems*, arXiv preprint arXiv:1801.01236, (2018).

[184] M. Raissi, P. Perdikaris, and G. E. Karniadakis, *Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations*, J. Comput. Phys., 378 (2019), pp. 686–707.

[185] M. Raissi, A. Yazdani, and G. E. Karniadakis, *Hidden fluid mechanics: Learning velocity and pressure fields from flow visualizations*, Science, 367 (2020), pp. 1026–1030.

[186] P. Ramachandran, B. Zoph, and Q. V. Le, *Searching for Activation Functions*, arXiv preprint arXiv:1710.05941, (2017).

[187] C. E. Rasmussen and C. Williams, *Gaussian processes for machine learning*, MIT Press, 2006.

[188] F. Rathgeber, D. A. Ham, L. Mitchell, M. Lange, F. Luporini, A. T. McRae, G.-T. Bercea, G. R. Markall, and P. H. Kelly, *Firedrake: automating the finite element method by composing abstractions*, ACM Trans. Math. Softw., 43 (2016), pp. 1–27.

[189] J. Rissanen, *A universal prior for integers and estimation by minimum description length*, Ann. Stat., 11 (1983), pp. 416–431.

[190] G. F. Roach, *Green's Functions*, Cambridge University Press, 2nd ed., 1982.

[191] V. Rokhlin, A. Szlam, and M. Tygert, *A randomized algorithm for principal component analysis*, SIAM J. Matrix Anal. Appl., 31 (2010), pp. 1100–1124.

[192] S. Roweis, *EM algorithms for PCA and SPCA*, Advances in Neural Information Processing Systems (NeurIPS), 10 (1997), pp. 626–632.

[193] W. Rudin, *Principles of mathematical analysis*, International series in pure and applied mathematics, McGraw-Hill, 3rd ed., 1976.

[194] ——, *Real and complex analysis*, McGraw-Hill, 3rd ed., 1986.

[195] S. H. Rudy, S. L. Brunton, J. L. Proctor, and J. N. Kutz, *Data-driven discovery of partial differential equations*, Sci. Adv., 3 (2017).

[196] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, *Learning representations by back-propagating errors*, Nature, 323 (1986), pp. 533–536.

[197] H. Schaeffer, *Learning partial differential equations via data discovery and sparse optimization*, Proc. Math. Phys. Eng. Sci., 473 (2017).

[198] F. Schäfer and H. Owhadi, *Sparse recovery of elliptic solvers from matrix-vector products*, arXiv preprint arXiv:2110.05351, (2021).

[199] F. Schäfer, T. J. Sullivan, and H. Owhadi, *Compression, inversion, and approximate PCA of dense kernel matrices at near-linear computational complexity*, Multiscale Model. Sim., 19 (2021), pp. 688–730.

[200] E. Schmidt, *Zur Theorie der linearen und nicht linearen Integralgleichungen Zweite Abhandlung*, Math. Ann, 64 (1907), pp. 161–174.

[201] M. Schmidt and H. Lipson, *Distilling free-form natural laws from experimental data*, Science, 324 (2009), pp. 81–85.

[202] K. Shukla, P. C. Di Leoni, J. Blackshire, D. Sparkman, and G. E. Karniadakis, *Physics-informed neural network for ultrasound nondestructive quantification of surface breaking cracks*, J. Nondestruct. Eval., 39 (2020), pp. 1–20.

[203] J. Sirignano and K. Spiliopoulos, *DGM: A deep learning algorithm for solving partial differential equations*, J. Comput. Phys., 375 (2018), pp. 1339–1364.

[204] L. N. Smith, *Cyclical learning rates for training neural networks*, in IEEE Winter Conference on Applications of Computer Vision, 2017, pp. 464–472.

[205] H. Stahl, *Best uniform rational approximation of $|x|$ on $[-1,1]$*, Mat. Sb., 183 (1992), pp. 85–118.

[206] ——, *Best uniform rational approximation of x^α on $[0,1]$*, Bull. Amer. Math. Soc., 28 (1993), pp. 116–122.

[207] I. Stakgold and M. J. Holst, *Green's Functions and Boundary Value Problems*, John Wiley & Sons, 2011.

[208] G. W. Stewart, *Matrix Algorithms: Volume 1: Basic Decompositions*, SIAM, 1998.

[209] A. M. Stuart, *Inverse problems: a Bayesian perspective*, Acta Numer., 19 (2010), pp. 451–559.

[210] E. Süli and D. F. Mayers, *An Introduction to Numerical Analysis*, Cambridge University Press, 2003.

[211] G. Szego, *Orthogonal polynomials*, AMS, Providence, RI, 4th ed., 1939.

[212] C. Taylor and P. Hood, *A numerical solution of the Navier-Stokes equations using the finite element technique*, Comput. Fluids, 1 (1973), pp. 73–100.

[213] M. Telgarsky, *Benefits of depth in neural networks*, in Conference on Learning Theory (COLT), 2016, pp. 1517–1539.

[214] ——, *Neural networks and rational functions*, in International Conference on Machine Learning (ICML), vol. 70, 2017, pp. 3387–3393.

[215] A. Townsend, *Pretty functions approximated by Chebfun2.* `https://www.chebfun.org/examples/approx2/PrettyFunctions.html`, 2013.

[216] ——, *Computing with functions in two dimensions*, PhD thesis, University of Oxford, 2014.

[217] A. Townsend and L. N. Trefethen, *An extension of Chebfun to two dimensions*, SIAM J. Sci. Comput., 35 (2013), pp. C495–C518.

[218] ——, *Continuous analogues of matrix factorizations*, P. Roy. Soc. A, 471 (2015).

[219] L. N. Trefethen, *Spectral Methods in MATLAB*, SIAM, 2000.

[220] ——, *Approximation Theory and Approximation Practice, Extended Edition*, SIAM, 2019.

[221] L. N. Trefethen and D. Bau III, *Numerical linear algebra*, SIAM, 1997.

[222] L. N. Trefethen, A. Birkisson, and T. A. Driscoll, *Exploring ODEs*, SIAM, 2017.

[223] L. N. Trefethen, Y. Nakatsukasa, and J. Weideman, *Exponential node clustering at singularities for rational approximation, quadrature, and PDEs*, Numer. Math., 147 (2021), pp. 227–254.

[224] M.-L. Udrescu and M. Tegmark, *AI Feynman: A physics-inspired method for symbolic regression*, Sci. Adv., 6 (2020).

[225] S.-M. Udrescu, A. Tan, J. Feng, O. Neto, T. Wu, and M. Tegmark, *AI Feynman 2.0: Pareto-optimal symbolic regression exploiting graph modularity*, in Advances in Neural Information Processing Systems (NeurIPS), vol. 33, 2020, pp. 4860–4871.

[226] Y. Urano, *A fast randomized algorithm for linear least-squares regression via sparse transforms*, Master's thesis, New York University, 2013.

[227] A.-J. Van Der Veen, E. F. Deprettere, and A. L. Swindlehurst, *Subspace-based signal analysis using singular value decomposition*, Proc. IEEE, 81 (1993), pp. 1277–1308.

[228] L. Vecci, F. Piazza, and A. Uncini, *Learning and approximation capabilities of adaptive spline activation function neural networks*, Neural Netw., 11 (1998), pp. 259–270.

[229] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, et al., *SciPy 1.0: fundamental algorithms for scientific computing in Python*, Nat. Methods, 17 (2020), pp. 261–272.

[230] J. von Neumann, *Some matrix-inequalities and metrization of matrix-space*, Tomsk Univ. Rev., 1 (1937), pp. 286–300.

[231] H. U. Voss, J. Timmer, and J. Kurths, *Nonlinear dynamical system identification from uncertain and indirect measurements*, Int. J. Bifurc. Chaos Appl. Sci. Eng., 14 (2004), pp. 1905–1933.

[232] N. S. Vyacheslavov, *On the uniform approximation of $|x|$ by rational functions.*, Sov. Math. Dokl., 16 (1975), pp. 100–104.

[233] S. Wang, H. Wang, and P. Perdikaris, *Learning the solution operator of parametric partial differential equations with physics-informed DeepONets*, Sci. Adv., 7 (2021).

[234] Z. Wang, X. Huan, and K. Garikipati, *Variational system identification of the partial differential equations governing the physics of pattern-formation: inference under varying fidelity and noise*, Comput. Methods Appl. Mech. Eng., 356 (2019), pp. 44–74.

[235] E. Wegert, *Visual Complex Functions: An Introduction with Phase Portraits*, Springer Science & Business Media, 2012.

[236] C. L. Wight and J. Zhao, *Solving Allen-Cahn and Cahn-Hilliard Equations Using the Adaptive Physics Informed Neural Networks*, Commun. Comput. Phys., 29 (2021), pp. 930–954.

[237] H. Wilber, A. Townsend, and G. B. Wright, *Computing with functions in spherical and polar geometries II. The disk*, SIAM J. Sci. Comput., 39 (2017), pp. C238–C262.

[238] C. Williams and M. Seeger, *Using the Nyström method to speed up kernel machines*, in Advances in Neural Information Processing Systems (NeurIPS), vol. 14, 2001, pp. 682–688.

[239] J. Wishart, *The generalised product moment distribution in samples from a normal multivariate population*, Biometrika, (1928), pp. 32–52.

[240] S. Wold, K. Esbensen, and P. Geladi, *Principal component analysis*, Chemometr. Intell. Lab., 2 (1987), pp. 37–52.

[241] F. Woolfe, E. Liberty, V. Rokhlin, and M. Tygert, *A fast randomized algorithm for the approximation of matrices*, Appl. Comput. Harmon. Anal., 25 (2008), pp. 335–366.

[242] D. Yarotsky, *Error bounds for approximations with deep ReLU networks*, Neural Netw., 94 (2017), pp. 103–114.

[243] A. Yazdani, L. Lu, M. Raissi, and G. E. Karniadakis, *Systems biology informed deep learning for inferring parameters and hidden dynamics*, PLoS Comput. Biol., 16 (2020).

[244] D. Zhang, L. Guo, and G. E. Karniadakis, *Learning in modal space: Solving time-dependent stochastic PDEs using physics-informed neural networks*, SIAM J. Sci. Comput., 42 (2020), pp. A639–A665.

[245] J. Zhang and W. Ma, *Data-driven discovery of governing equations for fluid dynamics based on molecular simulation*, J. Fluid Mech., 892 (2020).

[246] S. Zhang and G. Lin, *Robust data-driven discovery of governing physical laws with error bars*, Proc. R. Soc. A, 474 (2018).

[247] H. Zhao, B. D. Storey, R. D. Braatz, and M. Z. Bazant, *Learning the physics of pattern formation from images*, Phys. Rev. Lett., 124 (2020).