Principled interpolation of Green’s functions learned from data

Harshwardhan Praveen\textsuperscript{a,}\textsuperscript{*}, Nicolas Boullé\textsuperscript{b}, Christopher Earls\textsuperscript{a,c}

\textsuperscript{a}School of Civil and Environmental Engineering, Cornell University, Ithaca, NY, 14853, USA
\textsuperscript{b}Isaac Newton Institute for Mathematical Sciences, University of Cambridge, Cambridge, CB3 0EH, UK
\textsuperscript{c}Center for Applied Mathematics, Cornell University, Ithaca, NY, 14853, USA

Abstract

We present a data-driven approach to mathematically model physical systems whose governing partial differential equations are unknown, by learning their associated Green’s function. The subject systems are observed by collecting input-output pairs of system responses under excitations drawn from a Gaussian process. Two methods are proposed to learn the Green’s function. In the first method, we use the proper orthogonal decomposition (POD) modes of the system as a surrogate for the eigenvectors of the Green’s function, and subsequently fit the eigenvalues, using data. In the second, we employ a generalization of the randomized singular value decomposition (SVD) to operators, in order to construct a low-rank approximation to the Green’s function. Then, we propose a manifold interpolation scheme, for use in an offline-online setting, where offline excitation-response data, taken at specific model parameter instances, are compressed into empirical eigenmodes. These eigenmodes are subsequently used within a manifold interpolation scheme, to uncover other suitable eigenmodes at unseen model parameters. The approximation and interpolation numerical techniques are demonstrated on several examples in one and two dimensions.

Keywords: Green’s function, PDE learning, randomized SVD, POD, manifold interpolation

1. Introduction

It has been said that differential equations are the language of the universe \cite{1}. Indeed, most of our known physical laws are expressed mathematically, as rate forms \cite{2}; thus these types of equations are of great practical interest. Also of interest is the study of symmetries and invariant structures that occur within the solution operators accompanying such governing differential equations \cite{3}. These structures appear as patterns that are observable within the Green’s functions associated with some system of interest \cite{4}. 

\textsuperscript{*}Corresponding author

Email addresses: hp477@cornell.edu (Harshwardhan Praveen), nb690@cam.ac.uk (Nicolas Boullé),
earls@cornell.edu (Christopher Earls)
Though very useful in forming fast forward solvers, and for affording mechanistic insight, Green’s functions are not simple to find, in practice [5]. In response to this difficulty, the current paper proposes an approach for learning empirical Green’s functions (EFGs) from observational data emanating from the response of some system of interest.

Previous investigators have learned solution operators, mapping model coefficient functions directly into solutions to governing equations, using convolutional neural networks and graph neural networks [6, 7]. Some have even employed auto-encoder deep neural networks, as a kind of Koopman operator, uncovering latent spaces where response data from weakly nonlinear systems may be lifted, so as to become somewhat linear; and thus amenable for consideration with a learned Green’s function applied to this abstract, linear space [8]. The motivation for the foregoing important contributions to the literature appears to rest in either efficiently solving an inverse problem, or discovering an efficient to apply, forward model. In contrast to those contexts, others have employed deep rational neural networks [9] to uncover mechanistic insight from Green’s functions learned from data [4]. There have also been other works on solving and learning partial differential equations with data-driven approaches (including neural networks) [10, 11, 12, 13, 14, 15, 16, 17, 18], but our work is focused on learning a Green’s function.

In the present work, we propose a method for learning Green’s functions from data without resorting to machine learning. This idea, illustrated in Fig. 1 stems from earlier work aimed at learning Green’s function-like, Fredholm integral kernels using envi-
vironmental data, in order to formulate surrogate, reduced-order models exhibiting “just enough physics” to be useful in practical settings. The demonstration case for that earlier work was in predicting electro-magnetic ducting within the marine atmospheric boundary layer [19, 20]. The currently proposed method builds from these earlier ideas, so as to now learn Green’s function solution operators from observational data. While the method outlined herein is restricted to systems governed by self-adjoint differential operators, we note that many physical contexts are self-adjoint. Nonetheless, this is an important limitation to note, but we offer that our proposed approach for extending our method to weakly nonlinear (i.e., semi-linear) contexts [5] may offset concern over these limitations, by extending applicability into the nonlinear regime. We do this by interpolating our learned Green’s function models, in a structure preserving manner.

In the current work for uncovering meaningful descriptions of systems, in the form of solution operators to some self-adjoint physical context, we outline two data-driven approaches. The first one exploits the spectral decomposition of the kernel (Green’s function) within a first-kind Fredholm integral equation, in order to substitute its eigenfunctions with proper orthogonal decomposition modes (empirical eigenfunctions), learned from observed system response data under a well characterized forcing/source term. The second method uses the randomized singular value decomposition (SVD) to construct a low-rank approximation for the Green’s function, by using random Gaussian process with correlated entries, sampled from a multivariate normal distribution, to probe the unknown linear differential operator. We also propose a manifold interpolation scheme, for use in an offline-online setting, where offline excitation-response data, taken at specific model parameter instances, are compressed into empirical eigenmodes. These eigenmodes are subsequently used within a manifold interpolation scheme, to uncover other suitable eigenmodes, for an unseen model parameter instance; thus rendering an online, “just-in-time” EGF (obtained without the benefit of excitation-response data.) This interpolation approach is demonstrated on one and two-dimensional problems.

The present paper contains, in Section 2, a complete discussion on the proposed methods for learning and interpolating empirical Green’s Functions, along with details concerning considerations that are germane for the required data collection from the system of interest. Numerical examples of the method, applied to benchmark partial differential equations (PDEs), in one and two dimensions, with, and without noise appear in Section 3. Finally, concluding remarks are offered in Section 4.

2. Methodology

We consider a self-adjoint linear differential operator, \( \mathcal{L}_\theta \) depending on a set \( \theta = (\theta_1, \ldots, \theta_{N_{\text{param}}}) \in \mathbb{R}^{N_{\text{param}}} \) of modelling parameters, defined on a bounded and connected domain \( \Omega \subset \mathbb{R}^d \) in dimension \( 1 \leq d \leq 3 \), governing a physical system in the form of a boundary value problem:

\[
\mathcal{L}_\theta u = f, \quad \text{on } \Omega, \\
\mathcal{B}(u) = c, \quad \text{in } \partial \Omega, \tag{1}
\]

where \( \mathcal{B} \) is a linear differential operator specifying the boundary conditions of the problem, \( f \) is a forcing (or source) term, and \( u \) is the unknown system response. Under suitable conditions on the operator \( \mathcal{L}_\theta \) (e.g., uniformly elliptic), there exists a Green’s
function $G_\theta : \Omega \times \Omega \to \mathbb{R}$ associated with $L_\theta$ that is the impulse response of the linear differential operator and defined \[5\] as:

\[ L_\theta G_\theta(\vec{x}, \vec{s}) = \delta(\vec{s} - \vec{x}), \quad \vec{x}, \vec{s} \in \Omega, \]

where $L$ acts on the first variable and $\delta$ is the Dirac delta function. Then with homogeneous Dirichlet boundary conditions, $i.e.$, $B(u) = 0$ on the boundary of the domain, the solution to Eq. \[1\], for a forcing, $f$, can be expressed using the Green’s function within a Fredholm integral equation of the first-kind:

\[ u(\vec{x}) = \int_\Omega G_\theta(\vec{x}, \vec{s}) f(\vec{s}) d\vec{s}, \quad \vec{x} \in \Omega. \]

Moreover, according to the theory of first-kind Fredholm integral equations, the integral operator associated with $G_\theta(\vec{x}, \vec{s})$ has the following spectral decomposition \[21\]:

\[ G_\theta(\vec{x}, \vec{s}) = \sum_{k \geq 1} \frac{\Psi_k^*(\vec{x}; \theta) \Psi_k(\vec{s}; \theta)}{\omega_k(\theta)}, \quad \vec{x}, \vec{s} \in \Omega, \]

where $\omega_k(\theta) \in \mathbb{R}$ is the $k$th smallest eigenvalue (in absolute value) of $L_\theta$ associated with the eigenfunction, $\Psi_k$, and satisfying $L_\theta \Psi_k = \omega_k \Psi_k$ or, equivalently,

\[ \int_\Omega G_\theta(\vec{x}, \vec{s}) \Psi(\vec{s}; \theta) d\vec{s} = \frac{1}{\omega_k} \Psi_k(\vec{x}; \theta), \quad \vec{x} \in \Omega. \]

Then, the eigenfunctions of the Green’s function coincide with the eigenfunctions of the self-adjoint operator $L_\theta$. We will now discuss a way to replace these unknown operator eigenfunctions with empirical eigenfunctions; learned from the collected observational data from a system of interest. We begin with a description of the data (in our case synthetic data generated from simulations) which we use to learn the Green’s functions.

2.1. Generation of the training dataset

The training datasets used to learn Green’s functions consist of $N_{\text{samples}} \geq 1$ forcing terms, $\{f_j\}_{j=1}^{N_{\text{samples}}}$, and associated system’s responses, $\{u_j\}_{j=1}^{N_{\text{samples}}}$, satisfying Eq. \[1\]. The selection of the forcing terms plays an important role in the accuracy of the algorithm that learns the Green’s function. One would ideally want the set of forcing terms to be sufficiently “diverse” so that it approximates the vector space spanned by the dominant right singular vectors of the Green’s functions \[22\]. For instance, if the forcing terms in the training dataset are orthogonal to the first right singular vector, then one cannot hope to approximate the largest singular value of the Green’s function. To mitigate this risk, we probe the system \[1\] with random functions, sampled from a Gaussian process (GP), $\mathcal{GP}(0, K)$, with a covariance kernel, $K$, as motivated by recent theoretical results \[22\] \[23\].

The forcing terms of the one-dimensional examples presented in Section \[3\] are drawn from a GP, with mean zero and squared-exponential covariance kernel, $K_{\text{SE}}$, defined as

\[ K_{\text{SE}}(x, y) = \exp\left(-\frac{|x - y|^2}{(2\sigma)^2}\right), \quad x, y \in \Omega, \]
The notation \( u_{\{\Omega} \). For simplicity, the sensor locations in this work, the forcing terms, \( \top \) at these points to assemble the following column vectors of dimension \( N \) at arbitrary locations as well. We sample both the forcing terms and system responses nodes of the mesh that discretize the domain, but one could also evaluate the responses where \( x \) can only measure the responses at a finite number of locations, \( \bar{x} \). multivariate normal distribution with covariance matrix \( K \) the forcing matrix \( F \) \( \theta \) instantiated with a collection of parameters, \( \theta \) manifold interpolation described in Section 2.3.2, can be extended to work for operators in the FEniCS software \([27]\). Note that the methods we present here, including the a finite element method with piecewise quadratic Lagrange polynomials implemented \( u \) associated system response \( \mathcal{L}_\theta \). Hence, the theoretical bounds for the randomized SVD with arbitrary covariance kernels show that one may obtain higher accuracy by incorporating knowledge of the leading singular vectors of the differential operator into the covariance kernel \([22]\). In the present work, we employ a generic squared-exponential kernel as one may not have prior information about the governing operator, \( \mathcal{L}_\theta \), in real applications.

For simplicity in notation, we now restrict to a scalar model parameter \( \theta \in \mathbb{R} \), and an associated system response \( u_j \), under a forcing term \( f_j \); obtained by solving Eq. (1) using a finite element method with piecewise quadratic Lagrange polynomials implemented in the FEniCS software \([27]\). Note that the methods we present here, including the manifold interpolation described in Section 2.3.2 can be extended to work for operators instantiated with a collection of parameters, \( \theta = \{ \theta_1, \theta_2, \ldots, \theta_{N_{\text{param}}} \} \in \mathbb{R}^{N_{\text{param}}} \), but we now restrict ourselves to cases where \( \theta \in \mathbb{R} \). We are interested in applications where one can only measure the responses at a finite number of locations, \( \bar{x} = \{ \bar{x}_1, \ldots, \bar{x}_{N_{\text{sensors}}} \} \), where \( x_i \in \Omega \) and \( N_{\text{sensors}} \geq 1 \) is the number of measurements taken within the domain, \( \Omega \). For simplicity, the sensor locations in this work, \( \{ \bar{x}_i \}_{i=1}^{N_{\text{sensors}}} \), correspond with the nodes of the mesh that discretize the domain, but one could also evaluate the responses at arbitrary locations as well. We sample both the forcing terms and system responses at these points to assemble the following column vectors of dimension \( N_{\text{sensors}} \):

\[
    f_i := [f_i(\bar{x}_1) \, \cdots \, f_i(\bar{x}_{N_{\text{sensors}}})]^\top, \quad u_i(\theta) := [u_i(\bar{x}_1) \, \cdots \, u_i(\bar{x}_{N_{\text{sensors}}})]^\top,
\]

The notation \( u_i(\theta) \) highlights the dependence of the responses on the parameter \( \theta \), and \( \top \) denotes the matrix transpose. These columns are then assembled to create sets of forcing terms, \( F(\bar{x}) \), and corresponding system responses, \( E(\bar{x}; \theta) \), respectively, as

\[
    F(\bar{x}) := [f_1 \, \cdots \, f_{N_{\text{samples}}}], \quad E(\bar{x}; \theta) := [u_1(\theta) \, \cdots \, u_{N_{\text{samples}}}(\theta)],
\]

where \( F(\bar{x}) \) and \( E(\bar{x}; \theta) \) are real matrices of size \( N_{\text{sensors}} \times N_{\text{samples}} \). The columns of the forcing matrix \( F \) are independent and identically distributed (i.i.d.) and follow a multivariate normal distribution with covariance matrix \( K = (K(\bar{x}_i, \bar{x}_j))_{1 \leq i, j \leq N_{\text{sensors}}} \).

2.2. Low-rank approximation of Green's functions

Let \( G_\theta = (G_\theta(\bar{x}_i, \bar{x}_j))_{1 \leq i, j \leq N_{\text{sensors}}} \) be the symmetric matrix of the Green’s function \( G_\theta \), associated with the self-adjoint operator \( \mathcal{L}_\theta \), and evaluated at the measurement...
locations $\bar{x} = \{\bar{x}_1, \ldots, \bar{x}_{\text{Nsensors}}\}$. Given the sets $F$ and $E$ of forcing terms and responses, we are interested in computing a low-rank approximation to the Green’s matrix $G_\theta$, as

$$G_\theta \approx \Phi \Sigma \Phi^\top, \quad \Phi = [\Phi_1 \cdots \Phi_K] \in \mathbb{R}^{\text{Nsensors} \times K}, \quad \Sigma = \text{diag}(\sigma_1, \ldots, \sigma_K), \quad (2)$$

where $K \geq 1$ is the target rank, and $|\sigma_1| \geq \cdots \geq |\sigma_K|$. We require our matrix of empirical eigenvectors $\Phi$, to be orthonormal with respect to the quadrature weight matrix $W = \text{diag}((w_i)_{i=1}^{\text{Nsensors}})$, associated with the finite element discretization of the forcing terms and solutions, i.e., $\Phi^\top W \Phi = I_K$, where $I_K$ is the $K \times K$ identity matrix. We aim that the $k^{th}$ empirical eigenvector approximates the $k^{th}$ eigenfunction $\psi_k$ of the operator $L_\theta$ in the following $L^2$-sense:

$$\lim_{\text{Nsensors} \to \infty} \sum_{i=1}^{\text{Nsensors}} w_i [\Psi_k(\bar{x}_i) - \Phi_k(i)]^2 = 0, \quad 1 \leq k \leq K.$$ 

This ensures the $L^2$-convergence of the finite element function represented by the vector $\Phi_k$ to $\Psi_k$, as the number of sensors increases. Once a low-rank approximant has been obtained, the discretized solution $u$ to Eq. (1), under a new forcing term $f$, can be efficiently computed as

$$u \approx \sum_{k=1}^{K} \sigma_k \Phi_k \Phi_k^\top W f, \quad f = [f(\bar{x}_1) \cdots f(\bar{x}_{\text{Nsensors}})]^\top. \quad (3)$$

### 2.2.1. Proper orthogonal decomposition and least-square fitting of the coefficients

This section presents an algorithm for computing a low-rank approximation to $G_\theta$, of the form of Eq. (2), from pairs of forcing terms $F$, and associate responses $E$. The left singular vectors of the output ensemble $E$ are used as the proper orthogonal decomposition (POD) modes that serve as our empirical eigenvectors $\Phi \in \mathbb{R}^{\text{Nsensors} \times K}$, where $K$ is the desired rank of the approximant. The first step of the method consists of computing a singular value decomposition of the matrix $E$, to obtain an orthonormal basis for the range of the integral operator associated with the Green’s function:

$$U, S, V^* = \text{svd}(W^{1/2} E), \quad \Phi = W^{-1/2} U_K,$$

where $U_K$ is the matrix obtained by truncating $U$ to the first $K$ columns, and $\Phi$ approximates the first $K$ eigenfunctions of the Green’s function. We note that the empirical eigenvectors are orthonormalized using the quadrature weight matrix $W$. Additionally, the matrix $S$ of singular values provides us with a principled way to select the number of relevant empirical eigenvectors for constructing the empirical Green’s function (EGF) using the POD method.

Once the empirical eigenvectors have been computed, one can follow Eq. (2) to obtain a rank $K$ empirical Green’s function $G_\theta$ for the specified sensor locations, as $G_\theta \approx \Phi \Sigma \Phi^\top$, where $\Sigma \in \mathbb{R}^{K \times K}$ is an unknown diagonal coefficient matrix. We approximate the coefficients of the diagonal matrix $\Sigma(\theta)$ by solving the following least squares problem:

$$\Sigma \approx \arg \min_{Z = \text{diag}(Z_k)_{k=1}^{K}} \sum_{i=1}^{N_{\text{samples}}} \left\| u_i - \Phi Z \Phi^\top W f_i \right\|_2^2,$$ 

where $u_i$ is the $i^{th}$ sample of the response, $\Phi$ is the matrix of empirical eigenvectors, $f_i$ is the $i^{th}$ forcing term, and $Z_k$ is the $k^{th}$ diagonal element of the coefficient matrix $\Sigma$. This problem can be solved using the least squares method, and the resulting coefficients can be used to construct the empirical Green’s function $G_\theta$ for new sensor locations and forcing terms.
where $\| \cdot \|_2$ denotes the matrix 2-norm. We construct the normal equations for the least square minimization and then solve the system of equations using the linear algebra library in NumPy [29]. The main advantage of this algorithm is that it only requires a single pass over the differential operator, and does not assume any distribution on the forcing terms. In the next section, we will introduce a second algorithm which allows us to obtain higher accuracy under the assumption that we have control over the forcing terms that act on the system.

### 2.2.2. Randomized singular value decomposition

The randomized singular value decomposition (SVD) is a popular algorithm for computing a low-rank approximant to a large matrix $G \in \mathbb{R}^{N_{\text{sensors}} \times N_{\text{sensors}}}$ using matrix-vector products with random vectors $\mathbf{f}_1, \ldots, \mathbf{f}_{N_{\text{samples}}} \in \mathbb{R}^{N_{\text{sensors}}}$ [30, 31]. The error analysis for the randomized SVD in [30] uses standard Gaussian random vectors, but other random embedding techniques have been considered such as random permutations [32], sparse sign matrices [33, 34, 35, 36], and subsampled randomized trigonometric transforms (SRTTs) [32, 37, 38, 39], to mitigate the computational cost of Gaussian vectors in large-scale numerical linear algebra applications. Here, we employ a generalization of the randomized SVD, which uses random Gaussian vectors with correlated entries, sampled from a multivariate normal distribution [22]. Hence, we are interested in probing an unknown linear differential operator using smooth random forcing terms sampled from a Gaussian process, which implies that the discretized forcing term $F$ has correlated rows but independent columns (see Section 2.1).

We mainly use the algorithm described in [30, Sec. 1.5] with a slight modification, to ensure that the approximated singular modes are orthonormal with respect to the weight matrix $W$ associated with the finite element discretization. The first step in the randomized SVD method consists of computing the economized QR decomposition of the system’s response matrix $E$, to obtain an orthonormal basis for the range of $G$:

$$Q, R = qr(W^{1/2}E), \quad \tilde{Q} = W^{-1/2}Q.$$  

The multiplication by the weight matrix ensures that the columns $\tilde{q}_1, \ldots, \tilde{q}_K$ of $\tilde{Q}$ are orthonormal in the $L^2$-norm associated with the finite element discretization, i.e.,

$$\sum_{j=1}^{N_{\text{sensors}}} w_j \tilde{q}_k(j) \tilde{q}_{k'}(j) = \delta_{kk'}, \quad 1 \leq k, k' \leq K,$$

where $\delta_{kk'}$ is the Kronecker delta symbol. The next step consists of forming the matrix $B = \tilde{Q}^* G = (G^* \tilde{Q})^* \in \mathbb{R}^{K \times N_{\text{sensors}}}$, which requires the solution of the adjoint problem to Eq. (1), under forcing that is prescribed by the columns of $\tilde{Q}$. In this work, we assume that the partial differential operator is self-adjoint, and solve Eq. (1) with the forcing terms $\tilde{q}_1, \ldots, \tilde{q}_K$. We then evaluate the corresponding solutions at the nodes of the mesh to form $B$. Finally, we compute the singular value decomposition of the matrix $B$ before computing the left and right singular vectors, $\tilde{U}$ and $\tilde{V}$, respectively, which approximate the singular vectors of $G$ in the $L^2$ norm, as

$$U, S, V^* = \text{svd}(BW^{1/2}), \quad \tilde{U} = \tilde{Q}U, \quad \tilde{V} = W^{-1/2}V.$$  

Since the partial differential operator is self-adjoint, we select the empirical eigenvectors to be the columns of $\tilde{U}$, i.e., $\Phi = \tilde{U}$. We then choose the empirical eigenvalue matrix
\[ \Sigma \in \mathbb{R}^{K \times K} \text{ such that } \tilde{U} \Sigma \tilde{V}^* = \tilde{V} \Sigma \tilde{V}^*, \text{ i.e., } \Sigma = \tilde{V}^* W \tilde{U} S, \] to account for the sign flip between the left and right singular vectors associated with the operator \( \mathcal{L}_\theta \).

The main difference between this algorithm and the POD method described in Section 2.2.1 is that the randomized SVD requires two passes to probe the system: a first pass with random functions sampled from a Gaussian process, and a second pass with the functions defined by the columns of \( \tilde{Q} \). While the randomized SVD has near-optimal theoretical guarantees and requires many fewer response samples than the POD algorithm, one may not be able to employ forcing terms associated with the columns of \( \tilde{Q} \) in practical applications. However, several single pass randomized algorithms [10, 11, 12] have been proposed to alleviate this issue and only require Gaussian input vectors, such as the generalized Nyström method [13, 14, 15].

We remark that the approach described in this section could be generalized to non-self-adjoint operators by automatically deriving the adjoint associated with the partial differential operator \( \mathcal{L}_\theta \) using the dolfin-adjoint software package [16], and then solving the adjoint equation to compute an approximant to the Green’s function. However, we are mainly motivated by applications where the operator is not known, and do not consider the non self-adjoint case in this paper. Finally, one may also exploit the hierarchical low-rank structure of partial differential operators [47, 48] to build a good approximation to the Green’s functions with fewer forcing terms [23, 49, 50, 51].

2.3. Interpolation of EGFs to unobserved model parameter instances

Since Green’s functions assume linearity of the underlying differential operator governing the system, we extend our method to “weakly nonlinear” or semi-linear [5] contexts by assuming that the system responses at some model parameter instance, \( \theta \), are locally linear, and thus suitable for forming an EGF that applies locally within an underlying nonlinear manifold. Again, we would like to emphasize that this interpolation can be extended to contexts where the models are instantiated by a set of parameters, \( \theta \). We navigate this space by moving from linear “patch” to linear “patch” using manifold interpolation. The interpolation method described in this section is inspired by [52], and extended due to [53]. We describe an algorithm (see Algorithm 1) which takes a set of \( N_{\text{models}} \) eigenbases and coefficients, \( \{ \Phi(\bar{x}; \theta_j), \Sigma(\theta_j), \theta_j \}_{j=1}^{N_{\text{models}}} \), and interpolates between them to approximate a model at the target parameter instance \( \theta_* \), in the form of interpolated eigenmodes, \( \Phi(\bar{x}; \theta_*) \), and corresponding coefficients, \( \Sigma(\theta_*) \).

2.3.1. Correcting eigenmodes sign and order

The approximated Green’s function (EGF), \( G_\theta(\bar{x}, \bar{s}) = \Phi(\bar{x}; \theta) \Sigma(\theta) \Phi(\bar{x}; \theta)^\top \), constructed in Section 2.2, is not affected by sign flips of its eigenmodes, as the normalized eigenvalue decomposition is unique up to a sign of the eigenvectors. Therefore, two Green’s functions approximated at close parameters, \( \theta_1 \) and \( \theta_2 \), may have learned eigenmodes with opposite signs. Before performing the interpolation of the EGFs, we account for any potential sign flips by matching the signs of the individual eigenmodes. We first consider the eigenvectors at the origin of the manifold, \( \Phi(\bar{x}; \theta) \), around which we form the tangent space. We then compute an inner product between the eigenvectors of the other interpolants \( \Phi_k(\bar{x}; \theta_j) \) and the corresponding eigenvectors at the origin, \( \Phi_k(\bar{x}; \theta) \),
Algorithm 1 Interpolation of EGFs to unseen modeled parameters

**Input:** A set of eigenbases and coefficients, \( \{ \Phi(\bar{x}; \theta_j), \Sigma(\theta_j), \theta_j \}_{j=1}^{N_{\text{model}}} \) to be interpolated for a new parameter instance, \( \theta_\ast \).

**Step 1:** From among the \( \theta_j \)'s, identify a \( \hat{\theta} \), as being closest to \( \theta_\ast \), and use the associated eigenbasis, \( \Phi(\bar{x}; \hat{\theta}) \), as the origin point for the interpolation.

**Step 2:** Correct sign flips and shuffling of the eigenmodes, following the procedure described in Section 2.3.1.

**Step 3:** Construct \( L^2 \)-orthonormal matrices of eigenmodes using the quadrature weight matrix: \( \Psi(\bar{x}; \theta_j) = W^{1/2} \Phi(\bar{x}; \theta_j) \).

**Step 4:** Perform the interpolation:
1. Lift to the tangent space \( T_{\Psi} G_{n,r} \) (bijective to the horizontal space of \( T_{\Psi} ST_{n,r} \)) by using the following map [53],
   \[
   \Gamma_j = \Psi(\bar{x}; \theta_j) - \Psi(\bar{x}; \hat{\theta}) \text{ sym} \left( \Psi^\top(\bar{x}; \hat{\theta}) \Psi(\bar{x}; \theta_j) \right), \quad \text{sym} (Y) := (Y + Y^\top)/2.
   \]
2. Using Lagrange polynomials, compute the interpolated tangent vector, \( \Gamma_\ast \), using \( \{ \Gamma_j \}_{j=1}^{N_{\text{model}}} \). Interpolate the coefficients, \( \{ \tilde{\Sigma}(\theta_j) \}_{j=1}^{N_{\text{model}}} \), with the same scheme, to obtain \( \Sigma(\theta_\ast) \).
3. Compute the interpolated eigenvectors, \( \Psi(\bar{x}; \theta_\ast) \), by mapping \( \Gamma_\ast \) back to \( ST_{n,r} \) using the exponential map [53],
   \[
   \Psi(\bar{x}; \theta_\ast) = \text{qf}(\Psi(\bar{x}; \hat{\theta}) + \Gamma_\ast),
   \]
   where \( \text{qf}(A) \) denotes the \( Q \) factor of the QR decomposition of \( A \in \mathbb{R}^{n \times r} \).

**Step 5:** \( L^2 \)-orthonormalize \( \Psi \), using \( W : \Phi(\bar{x}; \theta_\ast) = W^{-1/2} \Phi(\bar{x}; \theta_\ast) \).

**Step 6:** Order the eigenbasis and coefficients to match the eigenbasis at the initial parameter instance, \( \hat{\theta} \).

**Output:** Interpolated eigenmodes \( \Phi(\bar{x}; \theta_\ast) \) and coefficients \( \Sigma(\theta_\ast) \) at parameter \( \theta_\ast \).
in order to update the signs of the interpolants’ eigenmodes, as follows:

\[ \Phi_k(\bar{x}; \theta_j) = \begin{cases} -\Phi_k(\bar{x}; \theta_j), & \text{if } \langle \Phi_k(\bar{x}; \theta_j), \Phi_k(\bar{x}; \hat{\theta}) \rangle < 0, \\ \Phi_k(\bar{x}; \theta_j), & \text{otherwise}, \end{cases} \]

where \( \langle \cdot, \cdot \rangle \) denotes the inner product with respect to the quadrature weight matrix, \( W \).

Figure 2: (A) Magnitude of the first two eigenvalues of the Green’s function associated with the one-dimensional Helmholtz equation. The first two corresponding eigenmodes are swapped when the frequency, \( \theta \), increases beyond the critical value of \( \theta_{\text{crit}} = \sqrt{5/2\pi} \approx 5 \), as displayed in (B) and (C).

Another potential issue that might arise during the manifold interpolation is the “shuffling” of eigenmodes. By construction, the eigenmodes approximated by the algorithms described in Section 2.2 are ordered according to the magnitude of the eigenvalues, but swapping of the eigenmodes may occur as the parameter \( \theta \) varies. As an example, consider the one-dimensional Helmholtz equation with frequency \( \theta \geq 0 \) and homogeneous boundary conditions:

\[ \frac{d^2 u}{dx^2} + \theta^2 u = f, \quad x \in [0, 1]. \]

The first two eigenvalues of the associated Green’s functions are given by \( \sigma_1 = 1/(\theta^2 - \pi^2) \) and \( \sigma_2 = 1/(\theta^2 - 4\pi^2) \), and satisfy \( |\sigma_1| > |\sigma_2| \), when \( \theta < \sqrt{5/2\pi} \), and \( |\sigma_1| \leq |\sigma_2| \) otherwise. This implies that the first two eigenmodes are swapped when \( \theta > \sqrt{5/2\pi} \), as illustrated by Fig. 2, and that the manifold interpolation technique will perform poorly in such cases. Note that the same phenomena occurs for the higher eigenmodes at larger values of \( \theta \). We resolve this issue by reordering the eigenmodes of the discovered basis at the given interpolation parameter to match the ones at the origin point \( \hat{\theta} \), where we are linearizing our Green’s function. For a given parameter, \( \theta_j \), and mode number, \( 1 \leq k \leq K \), we select the \( k \)th eigenmode at \( \theta_j \) to be the one with minimal angle with the \( k \)-th eigenmode, at parameter \( \theta \), i.e.,

\[ \Phi_k(\bar{x}; \theta_j) = \Phi_{\ell'}(\bar{x}; \theta_j), \quad \text{where } \ell' = \text{match}(k), \]

match(k) = \[ \arg \max_{1 \leq \ell \leq K, \ell \neq \text{match}(k')} |\langle \Phi_\ell(\bar{x}; \theta_j), \Phi_k(\bar{x}; \hat{\theta}) \rangle|, \quad 1 \leq k \leq K. \]

Note that the reordering is done by matching the eigenmodes, starting from the eigenmode with the highest singular value, without replacement. When the order of the
eigenmodes has been changed by this procedure, we also re-order the corresponding eigenvalues to preserve the value of the Green’s function.

2.3.2. Manifold interpolation

We begin by summarizing some useful notions and results from differential geometry; the interested reader may refer to [53] for a more exhaustive treatment. The Grassmann manifold \( \mathcal{G}_{n,r} \) is defined as the set of all \( r \)-dimensional subspaces in \( \mathbb{R}^n \), and a particular \( r \)-dimensional subspace from \( \mathbb{R}^n \) is represented with an element from \( \mathcal{G}_{n,r} \). Such an element can also be non-uniquely represented by some particular matrix, having orthonormal columns, \( \Psi \in \mathbb{R}^{n \times r} \): itself an element from within the equivalence class of matrices spanning the \( r \)-dimensional subspace in question. Such a matrix is an element from the \emph{compact Stiefel manifold}: the set of all such orthonormal matrices within \( \mathbb{R}^{n \times r} \) [53]. In our case, we construct each orthonormal matrix \( \Psi \) from data using a set of empirical eigenmodes \( \Phi \). Then, \( \Psi \) is an element within the compact Stiefel manifold \( \mathcal{S}_{n,r} \). Our aim is to use pre-existing offline observations to create a collection of \( \Phi \)'s that are suitable for an online interpolation, at a point where we have no observational data. It is not possible to directly interpolate the empirical eigenmodes modes within the compact Stiefel manifold since it is not a vector space. However, at each manifold point there exists a tangent space, which is a vector space, with its origin occurring at the point of tangency. We use the appropriate tangent space to interpolate the offline empirical eigenmodes, to create a “just-in-time,” online mode set, for use in constructing a suitable Green’s function at some desired model parameter instance \( \theta^* \).

The manifold interpolation scheme we employ exploits the bijective relation between the tangent space to the Grassmann manifold, \( T_{\Psi} \mathcal{G}_{n,r} \), and the horizontal space, within the tangent space to the compact Stiefel manifold, \( T_{\Psi} \mathcal{S}_{n,r} \). Along with the preserved metric structure between these two spaces [53], the use of the horizontal space has more practical benefits: it ensures uniqueness within the computational framework underpinning the interpolation, by allowing a particular matrix representation, from within the previously mentioned equivalence class, to be used in the computations [54]. Fig. 3 offers a schematic overview of the manifold interpolation, while Algorithm 1 describes...
the QR-decomposition based variant of the interpolation algorithm implemented in this work \cite{[19, 55]}. The scheme used in the present work relies only on a single QR decomposition. Thus, it is computationally cheaper than previous approaches \cite{[52]}, which require multiple singular value decompositions and matrix inversions. The process of returning from the horizontal space of $T_{\Psi}S_{n,r}$, back to $S_{n,r}$, with schemes that employ singular value decomposition might lead to a disruption within the ordering of the columns within the resulting matrix. For the problems that we consider, the resulting interpolated eigenbasis, using the proposed QR-based maps, are free from this mode "shuffling"; though we still check for this for the reasons mentioned in Section 2.3.1.

We remark that the matrix of empirical eigenmodes $\Phi(\bar{x}; \theta)$ constructed in Sections 2.2.1 and 2.2.2 is orthonormal with respect to the quadrature weights, as $\Psi(\bar{x}; \theta) = W^{1/2} \Phi(\bar{x}; \theta)$. We also need to specify the point on $T_{\Psi}S_{n,r}$, where we construct the required tangent space, within whose horizontal space we perform the interpolation. For this, we select the mode set with the parametric value, $\theta_j$, closest to the target parameter point $\theta_*$, i.e.,

$$\hat{\theta} = \arg \min_{1 \leq j \leq N_{\text{models}}} |\theta_j - \theta_*|,$$

where $\hat{\theta}$ denotes the parametric value at the origin. Once the origin has been identified, we use linear Lagrange polynomials to interpolate the tangent vectors $\{\Gamma_j\}_{j=1}^{N_{\text{models}}}$ within the horizontal space, as described in Algorithm 1. Since the associated coefficient matrices $\Sigma(\theta_*)$ occur within a Euclidean space, manifold interpolation is not required. Therefore, we interpolate these directly with linear Lagrange polynomials.

As a closing remark, we note that in some cases it is possible to interpolate the matrices $\Phi$, which correspond to the eigenmodes in our Green’s function model, directly with simple linear interpolation (e.g., with splines). This considerably simplifies the procedure outlined in Algorithm 1. However, such linear interpolation, unlike the manifold-based interpolation described in this work, is not structure preserving \cite{[56]}, and its accuracy is very problem dependent.

3. Numerical results

In this section, we evaluate the algorithms described in Section 2 for approximating Green’s functions (as EGFs) from input-output pairs, and also interpolating to unseen parameter points, $\theta$. A number of synthetic problems, in one and two dimensions, are considered. Unless specified otherwise, we generate the different training data sets presented in this section using the parameters $N_{\text{sensors}} = 2000$, $\sigma = 5 \times 10^{-3}$ (i.e., length-scale for the GP squared exponential covariance kernel), $N_{\text{samples}} = 2000$, and apply the low-rank approximation algorithms with a target rank of $K = 100$. Additionally, we only use 100 training pairs in the randomized SVD case to illustrate its superiority for approximating Green’s function with smaller training sets (at the cost of using a two pass procedure). For problems where the closed-form Green’s function is not known, we estimate the relative error of the EGF empirically by generating a testing data set of $N_{\text{test}} = 100$ input-output pairs of the form $\{(f_j, u_j)\}_{j=1}^{N_{\text{test}}}$. Then, we define the testing error as

$$\epsilon_{\text{test}} = \frac{1}{N_{\text{test}}} \sum_{j=1}^{N_{\text{test}}} \frac{\|u_j - \tilde{u}_j\|_{L^2(\Omega)}}{\|u_j\|_{L^2(\Omega)}}, \quad \tilde{u}_j = \int_{\Omega} G(\bar{x}, \bar{s}) f_j(\bar{s}) \, d\bar{s},$$

12
where the integrals are approximated by a quadrature rule that uses the finite element quadrature weights. Here, \( \tilde{u} \) denotes the reconstructed solution from the learned Green’s function. Note that we often express the relative error as a percentage by multiplying it by one hundred.

3.1. Approximation of Green’s functions

We first evaluate the algorithms for approximating Green’s functions from a training dataset of forcing terms and solutions on one and two-dimensional problems.

3.1.1. One dimensional Poisson equation

We begin with a one-dimensional Poisson equation with homogeneous boundary conditions:

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

The Green’s function associated with this problem is available in closed-form as

\[ G_{\text{exact}}(x,y) = \begin{cases} 
  x(1-s), & \text{if } x \leq s, \\
  s(1-x), & \text{if } s > x,
\end{cases} \]

where \( x, s \in [0, 1] \).

In Fig. 4, we display the exact Green’s function (panel E) along with its approximations constructed using the POD method (A) and the randomized SVD (C), introduced in Sections 2.2.1 and 2.2.2. We first compute the relative error \( \epsilon \) between the learned EGFs \( G \), and the exact Green’s function \( G_{\text{exact}} = G_{\text{exact}}(\bar{x}, \bar{s}) \) (evaluated at the sensor locations used to learn the EGF), as

\[ \epsilon = \frac{\|G - G_{\text{exact}}\|_{L^2([0,1] \times [0,1])}}{\|G_{\text{exact}}\|_{L^2([0,1] \times [0,1])}} \times 100, \]

and find that the POD method achieves a relative error of 1.1%, while the randomized SVD yields an error of 0.09%. We note that the performance of the randomized SVD is consistent with the theory [22, 30], which predicts a relative error of the order of 0.01% for a target rank of \( K = 100 \) and Green’s functions singular values, \( \sigma_k = 1/(\pi^2 k^2) \). We observe that the randomized SVD achieves a lower relative error than the POD method, while using fewer input-output training pairs. We also present the error contours for the approximated Green’s function where we plot the relative error,

\[ \epsilon_G = \frac{|G - G_{\text{exact}}|}{\|G_{\text{exact}}\|_{L^2([0,1] \times [0,1])}}, \]

for the POD method in Fig. 4(B) and the randomized SVD (D). Lastly, panel (F) compares the learned singular values with the first hundred exact singular values associated with the Green’s functions. While the randomized SVD yields a lower relative error on the Green’s function, the POD method is surprisingly able to recover the higher order singular values more accurately, while also capturing the algebraic decay rate. Finally, we evaluate the relative error of the learned Green’s function on the testing dataset, and obtain a testing error of 1.6% for the POD method and 0.1% for the randomized SVD.
Figure 4: EGF models for the 1D Poisson problem. (A) Green’s function learned using the POD method. (B) Error contour for the POD method. (C) Green’s function learned using the randomized SVD. (D) Error contour for the randomized SVD. (E) Exact Green’s function. (F) Comparison of coefficients learned by both methods with the exact singular values.
3.1.2. Noisy Poisson equation

We evaluate the effect of noise on the accuracy of the learned EGFs by perturbing the training solutions to the Poisson equation, generated in Section 3.1.1 with 10% Gaussian white noise. We assume that our system only has access to forcing terms \( \{ f_j \}_{j=1}^{N_{\text{samples}}} \) and associated noisy solutions \( \{ u_{\text{noisy}}^j \}_{j=1}^{N_{\text{samples}}} \) satisfying

\[
    u_{\text{noisy}}^j(\vec{x}_i) = u_j(\vec{x}_i) + 0.1c_{i,j}|\bar{u}_j|, \quad 1 \leq i \leq N_{\text{sensors}},
\]

where \( |\bar{u}_j| \) is the average of the \( j \)th system’s response and the \( c_{i,j} \sim \mathcal{N}(0, 1) \) are i.i.d.

The approximation error of the EGF over the domain is \( \epsilon = 3.5\% \) for the POD method, and \( \epsilon = 2.7\% \) for the randomized SVD method. The associated test error is \( \epsilon_{\text{test}} = 8.5\% \) for the POD method and 8.9\% for the randomized SVD. The error does increase, as we increase the noise in the data set; though, the POD method still learns a reasonable Green’s function approximation from the noisy dataset. This suggests that the POD model is not extremely sensitive to noise, in this case. We find that the randomized SVD has similar noise sensitivity to the POD technique, despite the fact that the response are perturbed twice: first when solving the PDE for random forcing terms and secondly for solution under the orthonormal function excitations.

3.1.3. Effect of different hyperparameters on the POD method

There are three dominant parameters influencing the fidelity of our empirical Green’s functions in the case of the POD method: the number of input-output pairs used for learning the empirical Green’s function, the length-scale parameter of the squared-exponential covariance kernel of the Gaussian Process, and finally the number of modes employed within the EGF. The number of sensors in the domain is chosen so that the forcing functions are sufficiently resolved. We illustrate the effects of these parameters on the EGF’s fidelity for the 1D Poisson problem. We report the relative error between the learned and exact Green’s function as a function of the parameters in Fig. 5. We discretize the domain with quadratic Lagrange (\( L_2 \)) finite elements. When the parameters are not varied for the study, they are fixed at \( N_{\text{samples}} = 2000, \sigma = 0.0025, K = 100, \) and \( N_{\text{sensors}} = 2000. \) For every set of parameters, we average the results over 10 simulations with different random forcing functions.

Fig. 5(A) shows that the error decreases exponentially fast as the number of training pairs in the dataset increases up to \( N_{\text{samples}} = 500, \) where it reaches a slower regime. Additionally, as we decrease the length scale parameter, the forcing functions sampled from the GP become more oscillatory. This allows the forcing functions to more effectively probe the domain of the EGF solution operator, and generate a diverse output ensemble, which increases the quality of the empirical eigenfunctions \[23\]. Learning empirical eigenvectors with higher fidelity to the eigenfunctions of the exact Green’s function leads to more accurate EGF models.

When we plotting the error as a function of the target rank \( K \), we find that the error decreases while \( K \) increased up to \( K = 50. \) As one allows more POD modes to be used in the representation of the EGF, the EGF accuracy improves up to that point. Since \( K \) is effectively the rank at which we are truncating the singular value decomposition, one can choose a value of \( K \) based on the decay in singular values. The singular value decay sheds light on where the plateau will form in the panel C of the plot in Fig. 5.
Figure 5: Relative error for the Green’s function associated with the 1D Poisson problem as a function of the following parameters: $N_{\text{samples}}$ (A), $\sigma$ (B), $K$ (C), and $N_{\text{sensors}}$ (D).
Finally, in Fig. 5(D), we plot the relative error as a function of the discretization size. We observe that error stays constant as we increase $N_{\text{sensors}}$ from a level which allows for sufficient resolution of the forcing functions on the domain. This demonstrates that the empirical Green’s function which the method learns is independent of the level of discretization of the domain.

3.1.4. “Multi-Physics” context

We now demonstrate our method on a problem in which the behavior of the physical context changes within the domain. The differential equation governing the system on the domain $\Omega = [0, 1]$ is given by

$$\frac{1}{2} \left( \frac{d^2 u}{dx^2} + \theta^2 u \right) = f \quad \text{on} \quad (0, 1/4), \quad -\frac{d^2 u}{dx^2} = f \quad \text{on} \quad (1/4, 1),$$

$$u(0) = u(1/4) = u(1) = 0. \tag{5}$$

As a result of this, the system is governed by a Helmholtz equation with parameter $\theta = 15$ on the left side of the domain, while it behaves as a Poisson equation for $x > 1/4$. We display the learned EGF obtained from the POD method and the randomized SVD method in the left and right panel of Fig. 6, respectively. The relative error on the testing data set is given by $\epsilon_{\text{test}} = 5.8\%$ for POD method and $\epsilon_{\text{test}} = 0.3\%$ for the randomized SVD method.

![Figure 6: Green’s function associated with Eq. (5) approximated by the POD method (A) and the randomized SVD method.](image)

3.1.5. Summary of errors for 1D problems

The errors for all of the 1D problems are summarized in Table 1. For all problems, there is a version with clean data and another version in which the system responses are contaminated with 10% additive white Gaussian noise (similar to the 1D Poisson problem in Section 3.1.2).

Looking at Table 1, we observe relatively low test errors (and relative errors), in the noise-free and also noise-contaminated cases. In general, the randomized SVD achieves better performance than the POD method. One possible explanation for this is that the minimization problem of the POD method (see Eq. (4)) may exhibit multiple local minima [57]. As a result, the POD method might fail to identify the optimal set of
Table 1: Summary of the test error with and without noise for the 1D Poisson, Helmholtz, Airy, and “multi-physics” problems. The relative error is mentioned in parenthesis for the cases where the exact Green’s function is known.

| Algorithm      | Poisson | Helmholtz | Airy  | Multi-Phys. |
|----------------|---------|-----------|-------|-------------|
| Clean POD      | 1.6 (1.1) | 4.0 (1.9) | 2.4   | 5.9         |
| Randomized SVD | 0.1 (0.09) | 0.2 (0.06) | 0.4   | 0.3         |
| Noisy POD      | 8.5 (3.5) | 4.6 (1.9) | 6.5   | 8.6         |
| Randomized SVD | 8.9 (2.7) | 2.4 (0.3) | 7.1   | 1.2         |

Singular values associated with the high frequency modes, whereas the randomized SVD does a fairly good job at approximating them, even in the presence of noise. However, there are a few examples (see the Poisson and Airy problems in Table 1), where POD performs marginally better than the randomized SVD when the inputs are contaminated with noise. This could be attributed to the fact that the responses are perturbed twice with the randomized SVD method (during the sketching and projection steps), whereas the responses are only perturbed once in the POD method.

3.1.6. Two dimensional Poisson equation

Finally, the two algorithms for approximating Green’s functions are compared on a Poisson problem in two dimensions. The governing equation with homogeneous boundary conditions on the unit disk reads:

\[
\nabla \cdot (\nabla u) = f, \quad \text{in } \Omega = D(0, 1),
\]

\[
u = 0, \quad \text{on } \partial \Omega.
\]

In this case, the closed-form expression for the associated Green’s function is known and given by [58]:

\[
G_{\text{exact}}(\vec{x}, \vec{s}) = \frac{1}{4\pi} \ln \left( \frac{(x_1 - s_1)^2 + (x_2 - s_2)^2}{(x_1 s_2 - x_2 s_1)^2 + (x_1 s_1 + x_2 s_2 - 1)^2} \right), \quad \vec{x} \neq \vec{s} \in \Omega,
\]

where \( \vec{x} = (x_1, x_2) \) and \( \vec{s} = (s_1, s_2) \). Following Section 2.1, the forcing functions \( \{f_j\}_{j=1}^{N_{\text{samples}}} \) are generated using the \texttt{randnfundisk} function of the Chebfun software system, with a length-scale parameter of \( \sigma = 0.2 \). Eq. (6) is discretized with quadratic Lagrange finite elements on a mesh of \( N_{\text{sensors}} \approx 5000 \) nodes and solved with the FEniCS software. We select a target rank of \( K = 500 \) and choose \( N_{\text{samples}} = 2000 \) for the POD method, while the randomized SVD method uses \( N_{\text{samples}} = 500 \) input-output pairs. After computing an approximation to the Green’s function using both methods, we visualize the two-dimensional slices at \( s_1 = s_2 = 0 \) and \( x_2 = s_2 = 0 \) of the learned EGF and exact Green’s function in Fig. 7. We subsequently observe a very good agreement between the low-rank EGF and the exact Green’s function, as confirmed by the low relative test errors of \( \epsilon_{\text{test}} = 4.7\% \) for the POD method and \( \epsilon_{\text{test}} = 0.4\% \) for the randomized SVD technique. We remark that the presence of a pole at \( x = s \) leads to a slow decay in the singular values associated with the Green’s function, since the function is not smooth and requires a larger number of training pairs to obtain an accurate approximation.
Figure 7: Green’s function associated with the two-dimensional Poisson equation approximated by the POD method (A,D) and randomized SVD method (B,E), along with the exact Green’s function (C,F). The top row displays the slice \( G(x_1, x_2, 0, 0) \) while the bottom row shows the Green’s function at \( x_2 = s_2 = 0 \), i.e. \( G(x_1, 0, s_1, 0) \).
3.2. Interpolation and extrapolation of Green’s functions

We now evaluate the interpolation algorithm described in Section 2.3.

3.2.1. One dimensional Airy problem

We begin with a parameterized Airy’s equation in one spatial dimension with homogeneous Dirichlet boundary conditions:

$$\frac{d^2 u}{dx^2} - \theta^2 x u = f, \quad x \in [0, 1],$$  \hspace{1cm} (7)

where $\theta \in \mathbb{R}$ is a parameter of the model. We first compute approximations to the Green’s functions at parameter values $\theta_1 = 1$, $\theta_2 = 5$, and $\theta_3 = 10$, using the randomized SVD. Then, we aim to interpolate the Green’s function at $\theta^* = 7$, where there is no training data. Since we do not have access to an analytical expression for the Green’s function associated with Eq. (7), we compare the interpolated model against the approximation (target Green’s function) computed by the randomized SVD at the target parameter in Fig. 8, i.e., using a dataset generated at $\theta_*=\theta^*$.

We obtain a relative error of $\epsilon = 2.7\%$ between the target and interpolated Green’s functions, while the relative error on the testing data set at $\theta^* = 7$ is equal to $\epsilon_{\text{test}} = 2.6\%$. While the error is six times larger than the one reported in Section 3.1.5 for the randomized SVD method, we note that the error is on the order of what the POD method reached. This is a relatively low error given the very small number of interpolation points used, i.e., just three points.

3.2.2. Extrapolation on 1D Airy problem

We note that our method also generalizes to extrapolation, out of the neighborhood of a set of learned empirical eigenmodes. To demonstrate this we compute approximations to the Green’s functions at parameter values $\theta_1 = 6.0$, $\theta_2 = 7.0$, and $\theta_3 = 8.0$ using the randomized SVD, and then subsequently extrapolate the EGF model to $\theta^* = 9.0$. We compare the extrapolated EGF against a randomized SVD based EGF model, learned (using ground truth, system response data) at the target parameter $\theta^*$ in Fig. 9.

We obtain a relative error of $\epsilon = 1.3\%$ between the target and extrapolated EGFs and testing error for the extrapolated EGF of $\epsilon_{\text{test}} = 1.1\%$. These errors are similar to the interpolation problem. While extrapolation is in general less reliable than interpolation, our method does appear to perform reasonably well in this case.

3.2.3. 2D Helmholtz problem

We now demonstrate our interpolation method on a 2D Helmholtz problem. The governing equations are given by

$$\nabla \cdot (\nabla u) + \theta^2 u = f, \quad \text{in } \Omega = D(0,1),$$  

$$u = 0, \quad \text{on } \partial \Omega.$$  \hspace{1cm} (8)

We employ the randomized SVD with a length-scale parameter of $\sigma = 0.2$, $N_{\text{samples}} = 100$, and the number of empirical eigenmodes used in the EGF model fixed to $K = 100$. We solve Eq. (8) with piecewise quadratic finite elements using FEniCS. The nodes of the mesh define the sensor locations $x \in \Omega$. The number of sensors is then, $N_{\text{sensors}} \sim 10000$. We learn empirical eigenmodes at parameter values of $\theta_1 = 4.8$, $\theta_2 = 4.9$, and $\theta_3 = 5.1$.  

20
Figure 8: (A-C) Approximation of the Green’s functions associated with the Airy problem at $\theta = 1, 5, 10$ used by the interpolation scheme. (D) Interpolated Green’s function at $\theta^* = 7$. (E) The target Green’s function approximated by randomized SVD.

Figure 9: EGFs for 1D Airy problem at $\theta_* = 9.0$ obtained using extrapolation on a manifold (A) and learned from data at the target parameter (B).
Using these empirical eigenmodes, we subsequently apply our interpolation method and obtain an EGF at $\theta^* = 5.0$. A comparison between the interpolated Green’s function (left) and a Green’s function learned using a randomized SVD based EGF model (i.e., learned using ground truth system response data at the target parameter $\theta^*$) is shown in Fig. 10. For this problem, the largest singular values change rapidly near $\theta_{\text{crit}} \approx 5.14$. While we are only using three points to find an interpolated Green’s function at $\theta^* = 5.0$, resulting in relatively high errors for the largest interpolated singular values, the remaining singular values are interpolated with high accuracy, as shown in Fig. 11.

Figure 10: Green’s function associated with the two-dimensional Helmholtz problem at $\theta^* = 5.0$, obtained by manifold interpolation (A, C), and learned at the target parameter from data (B, D). The top row display the slice $G(x_1, x_2, 0, 0)$, while the bottom row shows the slice $G(x_1, 0, x_3, 0)$.

3.2.4. One dimensional fractional laplacian problem

In this section, we evaluate the interpolation method on a one-dimensional fractional Laplacian operator with periodic boundary conditions:

$$(-\Delta)^\theta u = f, \quad u(-1) = u(1), \quad x \in [-1, 1], \quad (9)$$

where $0 < \theta < 1$ is the fractional order. We solve Eq. (9) using a Fourier spectral collocation method and use the randomized SVD to approximate the Green’s function at parameters $\theta_1 = 0.6$, $\theta_2 = 0.7$, and $\theta_3 = 0.8$. Then, we interpolate the Green’s functions at a new parameter $\theta^* = 0.75$ for which there is no training data available. We compare the interpolated model against the approximant (target Green’s function) computed by the randomized SVD at the target parameter $\theta^*$ in Fig. 12, i.e., using a dataset generated at the target parameter. We obtain a relative error of $\epsilon = 0.1\%$ between the target and interpolated Green’s function, while the relative error on the testing dataset at $\theta^* = 0.75$ is equal to $\epsilon_{\text{test}} = 0.8\%$.  

22
Figure 11: First hundred largest singular values of the Green’s function with the two-dimensional Helmholtz problem approximated by the randomized SVD method at $\theta = 4.9$ (A) and $\theta = 5.1$ (C). Panel (B) displays the first hundred interpolated singular values along with the exact ones at $\theta^* = 5.0$.

Figure 12: (A) Approximation of the Green’s function associated with the fractional Laplacian problem at $\theta^* = 0.75$ used by the interpolation scheme. (B) Interpolated Green’s function at $\theta^* = 0.75$. (C) Relative error between the interpolated and target model.
3.2.5. 2D Poisson on a Pac-Man-shaped domain

In this last example, we consider a more challenging parametrization, where the shape of the domain depends on the parameter $\theta$. We consider a two-dimensional Poisson problem defined on a Pac-Man-shaped domain $P(\theta) \subset \mathbb{R}^2$, with a variable-size inscribed angle $\theta \in [0, 2\pi)$. The governing equations are given by:

$$\nabla \cdot (\nabla u) = f, \quad \text{in } \Omega = P(\theta),$$
$$u = 0, \quad \text{on } \partial \Omega.$$  \hfill (10)

We approximate Green’s functions associated with Eq. (10) using the randomized SVD at parameters $\theta_1 = 0.4\pi$, $\theta_2 = 0.5\pi$, and $\theta_3 = 0.6\pi$, and apply our interpolation algorithm to recover the Green’s function at $\theta_\star = 0.55\pi$. To ensure that the empirical eigenmodes are defined on the same finite element basis, we first generate a mesh of $P(\theta)$ at $\theta_\star$ using the Gmsh meshing software [59]. Then, we apply a polar transformation of the coordinates of the mesh to generate meshes at $\theta_1$, $\theta_2$, and $\theta_3$ (see Fig. 13). If $(r_\star, \alpha_\star)$ represents the polar coordinates associated the mesh $P(\theta_\star)$ and $(r_1, \alpha_1)$ represents the same for the mesh $P(\theta_1)$, then the coordinate transform for changing a mesh $P(\theta_\star)$ to $P(\theta_1)$ is defined as

$$r_1 = r_\star, \quad \alpha_1 = \begin{cases} a\alpha_\star + b, & \text{if } r\sin(\alpha_\star) \geq 0, \\ a\alpha_\star + b, & \text{if } r\sin(\alpha_\star) < 0, \end{cases}$$

where $a = \frac{\pi - \alpha_\star}{\pi - \alpha_1}$ and $b = \pi - a\alpha_\star$. We solve Eq. (10) using the Firedrake finite element software [63] and employ the randomized SVD with the parameters $\sigma = 0.2$, $N_{\text{samples}} = 200$, $N_{\text{sensors}} \sim 5000$, $K = 200$, to approximate the Green’s functions at parameters $\theta_1$, $\theta_2$, and $\theta_3$.

![Figure 13: Meshes of the Pac-man shaped domain $P(\theta)$ with an inscribed angle $\theta$.](image)

In this case, the eigenmodes for the empirical Green’s functions learned at different parameters are defined on different domains. We then map the different eigenmodes to the target domain $P(\theta_\star)$ using the coordinate transform defined above. Here, we exploit Firedrake’s capabilities to compute the Jacobian associated with the coordinate transformation. We multiply the eigenmodes ($\Psi(x; \theta_j)$) for the interpolant with the determinant of the inverse of the Jacobian, to preserve the orthonormality of modes with respect to the mesh at $\theta_\star$. Using these empirical eigenmode sets, we subsequently apply our interpolation method and obtain an EGF at $\theta_\star = 0.55\pi$.  

24
Figure 14: Comparison between the modes of the Green’s function associated with the two-dimensional Poisson problem on a parameterized Pac-man shaped domain, approximated by interpolated empirical Green’s functions at $\theta_\ast = 0.55\pi$ (A,B,C,D) and the randomized SVD (E,F,G,H).

Figure 15: Two hundred largest singular values of the interpolated Green’s function associated with the two-dimensional Poisson problem on a parameterized Pac-man shaped domain, compared against a target Green’s function approximated using the randomized SVD at $\theta_\ast = 0.55\pi$. 
A comparison between the modes of the interpolated Green’s function (first row) and an approximation (target Green’s function) computed by the randomized SVD method (second row) at the target parameter (i.e., using a dataset generated at \( \theta^\ast \)), is shown in Fig. 14. A comparison between the coefficients learned using a randomized SVD based empirical Green’s function (i.e., learned using ground truth, system response data at the target parameter) and the interpolated empirical Green’s function is shown in Fig. 15. Note that we reorder the eigenmodes for the interpolated Green’s function to match the eigenmodes of the target empirical Green’s function, as described in Section 2.3.1.

4. Conclusions

In this work, we considered two methods based on the proper orthogonal decomposition and the randomized SVD for learning empirical Green’s functions from training data consisting of excitation-response pairs. The first method is more appropriate in cases where one has little control over the forcing functions, while the latter is more accurate, but requires more control over the forcing terms and two passes through the PDE. Both methods are observed to perform well in the one and two dimensional numerical experiments. Then, we proposed the use of a manifold interpolation scheme in an offline-online setting, where offline excitation-response data, taken at specific model parameter instances, are compressed into empirical eigenmodes. These eigenmodes are subsequently used within a manifold interpolation scheme, to uncover other suitable eigenmodes, for an unseen model parameter instance; thus rendering an online, “just-in-time” EGF (obtained without the benefit of excitation-response data) This interpolation approach is demonstrated in 1D and 2D contexts; yielding promising results.

Data availability

The code used to produce the numerical results is publicly available on GitHub at https://github.com/hsharsh/EmpiricalGreensFunctions for reproducibility purposes.

Acknowledgements

H.P. and C.J.E. were supported by the Army Research Office (ARO) Biomathematics Program grant W911NF-18-1-0351. N.B. was supported by the EPSRC Centre for Doctoral Training in Industrially Focused Mathematical Modelling through grant EP/L015803/1 in collaboration with Simula Research Laboratory and an INI-Simons Postdoctoral Research Fellowship. The authors also thank Max Jenquin for initial help in implementing this idea, as well as Maria Oprea, for insights regarding the manifold interpolation. We are grateful to Alex Townsend for his insights and many helpful suggestions.

References

[1] S. H. Strogatz, Infinite Powers: How Calculus Reveals the Secrets of the Universe, Mariner Books, 2019.
[2] R. P. Feynman, The Character of Physical Laws, M.I.T. Press, 1967.
[3] P. J. Olver, Application of Lie Groups to Differential Equations, 2nd Edition, Springer, 1993.
[4] N. Boullé, C. J. Earls, A. Townsend, Data-driven discovery of Green's functions with human-understandable deep learning, Sci. Rep 12 (1) (2022) 1–9.
[5] L. Evans, Partial differential equations, 2nd Edition, American Mathematical Society, 2010.
[6] J. Feliu-Fabà, Y. Fan, L. Ying, Meta-learning pseudo-differential operators with deep neural networks, J. Comput. Phys. 408 (2020) 109309.
[7] N. Kovachki, Z. Li, B. Liu, K. Azizzadenesheli, K. Bhattacharya, A. Stuart, A. Anandkumar, Neural Operator: Learning Maps Between Function Spaces (2021). arXiv:2108.08481.
[8] C. R. Gin, D. E. Shea, S. L. Brunton, J. N. Kutz, DeepGreen: Deep learning of Green's functions for nonlinear boundary value problems, Sci. Rep. 11 (1) (2021) 1–14.
[9] N. Boullé, Y. Nakatsukasa, A. Townsend, Rational neural networks, in: Advances in Neural Information Processing Systems (NeurIPS), Vol. 33, 2020, pp. 14243–14253.
[10] M. Raisi, Deep hidden physics models: Deep learning of nonlinear partial differential equations, J. Mach. Learn. Res. 19 (1) (2018) 932–955.
[11] M. Raisi, G. E. Karniadakis, Hidden physics models: Machine learning of nonlinear partial differential equations, J. Comput. Phys. 357 (2018) 125–141.
[12] M. Raisi, P. Perdikaris, 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) 686–707.
[13] S. H. Rudy, S. L. Brunton, J. L. Proctor, J. N. Kutz, Data-driven discovery of partial differential equations, Sci. Adv. 3 (4) (2017) e1602614.
[14] H. Schaeffer, Learning partial differential equations via data discovery and sparse optimization, Proc. R. Soc. A 473 (2197) (2017) 20160446.
[15] J. Berg, K. Nyström, Neural network augmented inverse problems for PDEs, arXiv preprint arXiv:1712.09685 (2017).
[16] S. L. Brunton, J. L. Proctor, J. N. Kutz, Discovering governing equations from data by sparse identification of nonlinear dynamical systems, Proc. Natl. Acad. Sci. U.S.A. 113 (15) (2016) 3932–3937.
[17] R. Stephany, C. Earls, PDE-READ: Human-readable partial differential equation discovery using deep learning, Neural Netw. 154 (2022) 360–382.
[18] C. Bonneville, C. Earls, Bayesian deep learning for partial differential equation parameter discovery with sparse and noisy data, J. Comput. Phys.: X 16 (2022) 100115.
[19] V. Fountoulakis, C. Earls, Inverting for maritime environments using proper orthogonal bases from sparsely sampled electromagnetic propagation data, IEEE Trans. Geosci. Remote Sens. 54 (12) (2016) 7166–7176.
[20] V. Fountoulakis, C. J. Earls, Duct heights inferred from radar sea clutter using proper orthogonal bases, Radio Sci. 51 (10) (2016) 1614–1626.
[21] T. Hsing, R. Eubank, Theoretical foundations of functional data analysis, with an introduction to linear operators, John Wiley & Sons, 2015.
[22] N. Boullé, A. Townsend, A generalization of the randomized singular value decomposition, in: International Conference on Learning Representations (ICLR), 2022.
[23] N. Boullé, A. Townsend, Learning elliptic partial differential equations with randomized linear algebra, Found. Comput. Math. (2022) 1–31.
[24] T. A. Driscoll, N. Hale, L. N. Trefethen, Chebfun Guide, Pafnuty Publications, 2014.
[25] S. Filip, A. Javeed, L. N. Trefethen, Smooth random functions, random ODEs, and Gaussian processes, SIAM Rev. 61 (1) (2019) 185–205.
[26] Z. Li, N. Kovachki, K. Azizzadenesheli, B. Liu, K. Bhattacharya, A. Stuart, A. Anandkumar, Fourier neural operator for parametric partial differential equations, in: International Conference on Learning Representations (ICLR), 2021.
[27] A. Logg, K.-A. Mardal, G. Wells, Automated solution of differential equations by the finite element method: The FEniCS book, Springer Science & Business Media, 2012.
[28] Y. Liang, H. Lee, S. Lim, W. Lin, K. Lee, C. Wu, Proper orthogonal decomposition and its applications—Part I: Theory, J. Sound Vib. 252 (3) (2002) 527–544.
[29] C. R. Harris, K. J. Millman, S. J. Van Der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, et al., Array programming with NumPy, Nature 585 (7825) (2020) 357–362.
[30] N. Hallo, P.-G. Martinsson, J. A. Tropp, Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions, SIAM Rev. 53 (2) (2011) 217–288.
[31] P.-G. Martinsson, J. A. Tropp, Randomized numerical linear algebra: Foundations and algorithms, Acta Numer. 29 (2020) 403–572.
[32] N. Ailon, B. Chazelle, The fast Johnson–Lindenstrauss transform and approximate nearest neighbors, SIAM J. Comput. 39 (1) (2009) 302–322.
[33] K. L. Clarkson, D. P. Woodruff, Low-rank approximation and regression in input sparsity time, J. ACM 63 (6) (2017) 1–45.
[34] X. Meng, 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.
[35] J. Nelson, 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.
[36] Y. Urano, A fast randomized algorithm for linear least-squares regression via sparse transforms, Master’s thesis, New York University (2013).
[37] N. Ailon, 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.
[38] D. S. Parker, Random Butterfly Transformations with Applications in Computational Linear Algebra, Rep. CSD-950023, UCLA (1995).
[39] F. Woolfe, E. Liberty, V. Rokhlin, M. Tygert, A fast randomized algorithm for the approximation of matrices, Appl. Comput. Harmon. Anal. 25 (3) (2008) 335–366.
[40] J. A. Tropp, A. Yurtsever, M. Udell, V. Cevher, Practical sketching algorithms for low-rank matrix approximation, SIAM J. Matrix Anal. Appl. 38 (4) (2017) 1454–1485.
[41] J. A. Tropp, A. Yurtsever, M. Udell, V. Cevher, Streaming low-rank matrix approximation with an application to scientific simulation, SIAM J. Sci. Comput. 41 (4) (2019) A2430–A2463.
[42] J. Upadhyay, Fast and space-optimal low-rank factorization in the streaming model with application in differential privacy, arXiv preprint arXiv:1604.01429 (2016).
[43] Y. Nakatsukasa, Fast and stable randomized low-rank matrix approximation, arXiv preprint arXiv:2009.11392 (2020).
[44] E. J. Nyström, Über die praktische Auflösung von Integralgleichungen mit Anwendungen auf Randwertaufgaben, Acta Math. 54 (1930) 185–204.
[45] C. Williams, M. Seeger, Using the Nyström method to speed up kernel machines, in: Advances in Neural Information Processing Systems (NeurIPS), Vol. 13, 2000.
[46] P. E. Farrell, D. A. Ham, S. W. Funke, M. E. Rognes, Automated derivation of the adjoint of high-level transient finite element programs, SIAM J. Sci. Comput. 35 (4) (2013) C369–C393.
[47] M. Bebendorf, W. Hackbusch, Existence of H-matrix approximants to the inverse FE-matrix of elliptic operators with $L^\infty$-coefficients, Numer. Math. 95 (1) (2003) 1–28.
[48] M. Bebendorf, Hierarchical matrices, Springer, 2008.
[49] N. Boullé, S. Kim, T. Shi, A. Townsend, Learning Green’s functions associated with time-dependent partial differential equations, J. Mach. Learn. Res. 23 (218) (2022) 1–34.
[50] L. Lin, J. Lu, L. Ying, Fast construction of hierarchical matrix representation from matrix–vector multiplication, J. Comput. Phys. 230 (10) (2011) 4071–4087.
[51] P.-G. Martinsson, Fast direct solvers for elliptic PDEs, SIAM, 2019.
[52] D. Amsallem, C. Farhat, Interpolation method for adapting reduced-order models and application to aeroelasticity, AIAA J. 46 (7) (2008) 1803–1813.
[53] P.-A. Absil, R. Mahony, R. Sepulchre, Optimization algorithms on matrix manifolds, Princeton University Press, 2008.
[54] A. Edelman, T. A. Arias, S. T. Smith, The geometry of algorithms with orthogonality constraints, SIAM J. Matrix Anal. Appl. 20 (2) (1998) 303–353.
[55] R. Sternfels, C. J. Earls, Reduced-order model tracking and interpolation to solve PDE-based Bayesian inverse problems, Inverse Probl. 29 (7) (2013) 075014.
[56] J. Degroote, J. Vierendeels, K. Willcox, Interpolation among reduced-order matrices to obtain parameterized models for design, optimization and probabilistic analysis, Int. J. Numer. Methods Fluids 63 (2010) 207–230.
[57] M. A. Gilles, C. Earls, D. Bindel, A subspace pursuit method to infer refractivity in the marine atmospheric boundary layer, IEEE Trans. Geosci. Remote Sens. 57 (8) (2019) 5606–5617.
[58] T. Myint-U, L. Debnath, Linear partial differential equations for scientists and engineers, Springer Science & Business Media, 2007.
[59] C. Geuzaine, 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 (11) (2009) 1309–1331.

[60] F. Rathgeber, D. A. Ham, L. Mitchell, M. Lange, F. Luporini, A. T. T. McRae, et al., Firedrake: automating the finite element method by composing abstractions, ACM Trans. Math. Softw. 43 (3) (2016) 1–24.