Journal of Computational Physics (2025)

Contents lists available at ScienceDirect

Journal of Computational Physics

journal homepage: www.elsevier.com/locate/jcp

Reduced–Basis Deep Operator Learning for Parametric PDEs with Independently Varying Boundary and Source Data

ueqi Wang[a], Guang Lin[a,b,*]

Department of Mathematics, Purdue University, 610 Purdue Mall, West Lafayette, 47907, IN, USA
School of Mechanical Engineering, Purdue University, 610 Purdue Mall, West Lafayette, 47907, IN, USA

R T I C L E I N F O

rticle history:

eywords: reduced basis methods, perator learning, DeepONet, finite lement methods, parametric PDEs.

A B S T R A C T

Parametric PDEs power modern simulation, design, and digital-twin systems, yet their many-query workloads still hinge on repeatedly solving large finite-element systems. Existing operator-learning approaches accelerate this process but often rely on opaque learned trunks, require extensive labeled data, or break down when boundary and source data vary independently from physical parameters. We introduce RB–DeepONet, a hybrid operator-learning framework that fuses reduced-basis (RB) numerical structure with the branch–trunk architecture of DeepONet. The trunk is fixed to a rigorously constructed RB space generated offline via Greedy selection, granting physical interpretability, stability, and certified error control. The branch network predicts only RB coefficients and is trained label-free using a projected variational residual that targets the RB–Galerkin solution. For problems with independently varying loads or boundary conditions, we develop boundary and source modal encodings that compress exogenous data into low-dimensional coordinates while preserving accuracy. Combined with affine or empirical interpolation decompositions, RB–DeepONet achieves a strict offline–online split: all heavy lifting occurs offline, and online evaluation scales only with the RB dimension rather than the full mesh. We provide convergence guarantees separating RB approximation error from statistical learning error, and numerical experiments show that RB–DeepONet attains accuracy competitive with intrusive RB–Galerkin, POD–DeepONet, and FEONet while using dramatically fewer trainable parameters and achieving significant speedups. This establishes RB–DeepONet as an efficient, stable, and interpretable operator learner for large-scale parametric PDEs.

*Corresponding author: Guang Lin, E-Mail: Guanglin@purdue.edu

1. Introduction

Computational prediction for parametric PDEs underpins design, control, and digital-twin applications in science and engineering [1, 2, 3]. Classical discretizations, such as finite difference [4], finite volume [5], and especially finite element methods (FEM) [6], provide robust high-fidelity solvers, but their cost scales with the full spatial degrees of freedom and must be paid anew for each parameter query. Consequently, tasks that require repeated forward solves, such as parametric sweeps [2], PDE-constrained optimization [7], and uncertainty quantification [8], can become computationally prohibitive on fine meshes or in high-dimensional parameter spaces, even though the finite element method remains the predominant high-fidelity discretization in engineering analysis.

Projection-based model reduction techniques [9, 10] alleviate this bottleneck by constructing low-dimensional subspaces that approximate the solution manifold over the parameter domain. Among them, Reduced Basis (RB) method [11] offers a rigorously certified offline–online framework: The offline stage generates high-fidelity snapshots and constructs a reduced trial space, typically via Proper Orthogonal Decomposition (POD) [2, 11, 12] or a Greedy selection [13, 14, 15]. The online stage performs inexpensive Galerkin projection [16, 17] onto this reduced space. Under an affine or empirical interpolation decomposition, all parameter-independent operators can be preassembled, so the online complexity depends only on the reduced dimension N rather than the full-order dimension N_h. Rigorous *a posteriori* error estimators accompany each query, providing certified accuracy [2, 11, 18]. Nevertheless, classical RB methods remain intrusive and are not easily adaptable to heterogeneous or independently varying boundary and source data [19, 20, 21]. Nevertheless, classical RB methods remain intrusive and are not easily adaptable to heterogeneous or independently varying boundary and source data [20, 21, 19].

In parallel, machine learning has shown considerable promise for accelerating the numerical solutions of PDEs. Physics-Informed Neural Networks (PINNs) [22] incorporate PDE residuals and boundary conditions directly into the loss function, enabling solution approximation without labeled data. Building on this idea, a large body of work has proposed variants of PINNs to tackle a wide range of PDE and physics-based problems [23, 24, 25, 26, 27]. However, standard PINNs and their extensions are typically designed for a single boundary or forcing configuration and must be retrained when the input data changes. In addition, the trial space is only defined implicitly by the network, which hinders the exact enforcement of boundary conditions. These limitations make PINN-type methods difficult to apply to parametric PDEs or to problems where the input data vary across queries.

Operator-learning frameworks overcome these limitations by learning mappings between function spaces and are therefore well-suited to parametric PDEs. Fourier Neural Operators (FNOs) [28] parameterize the operator via Fourier convolutions, injecting a spectral inductive bias that is particularly effective on regular grids. DeepONet [29] approximates nonlinear operators via a branch–trunk architecture and is supported by the universal approximation theorem for operators [30]. Physics-Informed Neural Operators (PINO) [31] further combine operator learning with physics-informed residual losses to improve data efficiency. Most recently, hybrid operator learning methods integrate classical numerical structure into the trunk network of DeepONet, leading to several variants that differ primarily in the choice of basis. The Laplacian Eigenfunction-based Neural Operator (LE-NO) [32] uses Laplacian eigenfunctions as the trunk, enabling efficient approximation on nonlinear parabolic PDEs. POD–DeepONet [33] projects solutions onto a POD basis and learns only the corresponding coefficients, yielding a low-rank representation in a reduced output space. Finite Element Operator Network (FEONet) [34] fixes the trunk to finite element basis and enforces the FEM variational residual directly. However, these methods typically rely on learned trunks with limited physical interpretability, require large labeled datasets, or predict full-order fields with dimension N_h—hindering online efficiency. To handle parametric PDEs more efficiently, a natural idea is therefore to fix the trunk to a reduced basis that captures the solution manifold with as few modes as possible, thereby combining model reduction with operator learning.

Actually, there is a growing line of work that couples model reduction techniques with learning for parametric PDEs. Hesthaven & Ubbiali [35] introduced a non-intrusive POD–ANN pipeline that learns a regression map from parameters to reduced coefficients. In [36], the PDE-aware deep neural network (PDE-DNN) uses a reduced basis solver as the activation function in the last layer, so that the network outputs parameters for a precomputed RB model. Chen et al. proposed physics-reinforced neural networks (PRNN) [37], which map parameters to RB coefficients using a loss that blends the RB–Galerkin residual with coefficient labels. Sentz et al. [38] trained a neural timestepping map that propagates RB coefficients for dynamical PDEs. O'Leary-Roseberry et al. proposed a reduced basis adaptive residual network [39] for learning high-dimensional parametric maps. For further combinations of reduced basis methods with neural networks, we refer to [35, 40, 41]. More recently, several works have combined model reduction with operator learning. In [42] and [43], reduced operators are identified directly from data via operator inference.

RO-NORM [44] projects spatio-temporal fields onto a POD basis and then learns the mapping between these reduced functions via the NORM neural operator on Riemannian manifolds. Reduced Basis Neural Operator (ReBaNO) [45] constructs a reduced basis from high-fidelity PINN solutions selected via a greedy procedure, and represents the solutions of parametric PDEs in this low-dimensional RB space.

Nevertheless, to the best of our knowledge, existing approaches do not provide a clear framework that operates entirely in a reduced basis space while simultaneously handling parametric dependence and independently varying external data (e.g., source terms and boundary conditions). To address this gap, we propose in this paper the Reduced-Basis DeepONet (RB–DeepONet), a hybrid operator-learning framework that combines the DeepONet architecture with Reduced Basis method.

Our key contributions are:

- **RB trunk**: We fix the DeepONet trunk to a rigorously constructed RB basis obtained via Greedy selection, yielding interpretability, stability, and certified error control.
- **Label-free residual training**: The branch network predicts only RB coefficients, trained by minimizing the RB-projected variational residual. This guarantees convergence to RB–Galerkin solutions on the training distribution without paired solution labels.
- **Independently varying data**: To handle boundary and source inputs that vary independently of the physical parameters, we introduce compact boundary and source modal encodings, producing a low-dimensional augmented input vector.
- **Strict offline–online split**: Under affine decomposition or empirical interpolation (EIM), all parameter-independent reduced operators are preassembled offline, making the online cost depend only on the RB dimension $N \ll N_h$.
- **Convergence theory**: We establish bounds that separate RB approximation error from statistical learning error, ensuring reliable generalization.

Numerical experiments show that RB–DeepONet achieves accuracy comparable to intrusive RB–Galerkin, POD–DeepONet [29], and FEONet [34], while requiring significantly fewer trainable parameters and providing substantial online speedups.

The remainder of the paper is organized as follows. Section 2 reviews the variational formulation, finite element discretization, and the lifted setting we adopt. Section 3 first develops RB–DeepONet for fully parameterized PDEs, including the construction of the RB trunk and the residual-minimization training objective. It then extends the framework to operator-parameterized problems with exogenous data by introducing boundary and source modes together with lifting and trace integration. Section 4 presents the convergence and generalization analysis. Section 5 reports numerical experiments, and Section 6 concludes with a discussion and perspectives.

2. Problem Setting

For clarity of exposition, we develop the method in the prototypical second-order elliptic setting (2.1). The proposed algorithm, however, extends to more general operators with only minor modifications, as summarized in Remark 3.4.

2.1. Parametric PDEs

Let $\Omega \subset \mathbb{R}^d$, $d \in \mathbb{N}$, be a bounded Lipschitz domain with boundary $\partial\Omega := \Gamma_D \cup \Gamma_N \cup \Gamma_R$, where the partition is pairwise disjoint corresponding to Dirichlet, Neumann and Robin parts, respectively. Let $\mathcal{D} \subset \mathbb{R}^p$, $p \geq 1$, be a compact parameter set and $\mathbf{k} \in \mathcal{D}$ a parameter vector. For a given parameter $\mathbf{k} \in \mathcal{D}$, we consider the family of elliptic boundary value problems

$$\begin{cases} -\nabla\cdot(A(\mathbf{x};\mathbf{k})\nabla u(\mathbf{x};\mathbf{k})) + c_0(\mathbf{x};\mathbf{k})\, u(\mathbf{x};\mathbf{k}) = f(\mathbf{x};\mathbf{k}), & \mathbf{x} \in \Omega, \\ \gamma_D u(\mathbf{x};\mathbf{k}) = g_D(\mathbf{x};\mathbf{k}), & \mathbf{x} \in \Gamma_D, \\ (A(\mathbf{x};\mathbf{k})\nabla u(\mathbf{x};\mathbf{k}))\cdot \boldsymbol{n} = h_N(\mathbf{x};\mathbf{k}), & \mathbf{x} \in \Gamma_N, \\ (A(\mathbf{x};\mathbf{k})\nabla u(\mathbf{x};\mathbf{k}))\cdot \boldsymbol{n} + \beta(\mathbf{x};\mathbf{k})\, u(\mathbf{x};\mathbf{k}) = r_R(\mathbf{x};\mathbf{k}), & \mathbf{x} \in \Gamma_R, \end{cases} \tag{2.1}$$

where $\gamma_D : V := H^1(\Omega) \to G_D := H^{1/2}(\Gamma_D)$ denotes the Dirichlet trace on Γ_D and $\boldsymbol{n}$ the unit outward normal on $\partial\Omega$. For each $\mathbf{k} \in \mathcal{D}$, we assume $A(\cdot;\mathbf{k}), c_0(\cdot;\mathbf{k}), \beta(\cdot;\mathbf{k})$ are in L^∞ spaces and the data are taken in the natural spaces

$$f(\cdot;\mathbf{k}) \in L^2(\Omega), \quad g_D(\cdot;\mathbf{k}) \in H^{1/2}(\Gamma_D), \quad h_N(\cdot;\mathbf{k}) \in H^{-1/2}(\Gamma_N), \quad r_R(\cdot;\mathbf{k}) \in H^{-1/2}(\Gamma_R),$$

which are uniformly bounded w.r.t. $\mathbf{k}$. The weak form of (2.1) reads: for given $\mathbf{k} \in \mathcal{D}$, find $u(\mathbf{k}) \in V$ with $\gamma_D u(\mathbf{k}) = g_D(\mathbf{k})$ such that

$$a(u(\mathbf{k}), v; \mathbf{k}) = \ell(v; \mathbf{k}) \qquad \forall v \in V_0. \tag{2.2}$$

For brevity, we henceforth omit the explicit spatial variable and write $u(\mathbf{x};\mathbf{k}), g(\mathbf{x};\mathbf{k})$ for $u(\mathbf{k}), g(\mathbf{k})$. We set $V_0 := \{v \in H^1(\Omega) : \ \gamma_D v = 0\}$ and define the bilinear form

$$a(u, v; \mathbf{k}) := \int_\Omega A\nabla u \cdot \nabla v + \int_\Omega c_0\, uv + \int_{\Gamma_R} \beta\, \gamma_R u\, \gamma_R v,$$

and linear form

$$\ell(v; \mathbf{k}) := \int_\Omega f\, v + \int_{\Gamma_N} h_N\, \gamma_N v + \int_{\Gamma_R} r_R\, \gamma_R v.$$

Here, $\gamma_E : H^1(\Omega) \to H^{1/2}(\Gamma_E)$ is the trace operator restricted on Γ_E for $E \in \{N, R\}$. For each $\mathbf{k} \in \mathcal{D}$, we fix a reference parameter $\mathbf{k}_\star \in \mathcal{D}$ and use

$$(u, v)_V := a(u, v; \mathbf{k}_\star)$$

as the reference energy inner product in V and V_0, with associated norm $\|u\|_V := \sqrt{(u,u)_V}$.

We impose the following assumptions in what follows.

Assumption 2.1. *We assume that $a : V \times V \to \mathbb{R}$ is a continuous and symmetric bilinear form and that $\ell : V \to \mathbb{R}$ is a linear functional such that there exist constants $M, \alpha_{\mathrm{LB}}, C_\ell > 0$ independent of $\mathbf{k} \in \mathcal{D}$ with*

$$|a(u, v; \mathbf{k})| \le M\, \|u\|_V \|v\|_V, \qquad \forall u, v \in V, \tag{2.3}$$

$$a(v, v; \mathbf{k}) \ge \alpha_{\mathrm{LB}}\, \|v\|_V^2, \qquad \forall v \in V_0, \tag{2.4}$$

$$|\ell(v; \mathbf{k})| \le C_\ell \|v\|_V, \qquad \forall v \in V_0. \tag{2.5}$$

In addition, we assume that the Dirichlet trace $\gamma_D : V \to G_D$ admits a bounded right inverse $T : G_D \to V$, i.e.,

$$\gamma_D(Tg) = g, \quad \|Tg\|_V \le C_{\mathrm{tr}} \|g\|_{G_D}, \qquad \forall g \in G_D, \tag{2.6}$$

for some constant $C_{\mathrm{tr}} > 0$ independent of $\mathbf{k}$.

Under Assumption 2.1, the Lax–Milgram theorem [46] implies that, for each $\mathbf{k} \in \mathcal{D}$, there exists a unique $u(\mathbf{k}) \in V$ with $\gamma_D u(\mathbf{k}) = g_D(\mathbf{k})$ solving the weak problem (2.2). Moreover, there exists a unique bounded linear lifting operator $E_D : G_D \to V$ such that

$$\gamma_D(E_D g) = g, \qquad a(E_D g, v; \mathbf{k}_\star) = 0 \quad \forall v \in V_0. \tag{2.7}$$

This operator induces an inner product and norm on G_D via

$$\langle g_1, g_2 \rangle_{D,\star} := a(E_D g_1, E_D g_2; \mathbf{k}_\star), \qquad \|g\|_{D,\star} := \langle g, g \rangle_{D,\star}^{1/2}. \tag{2.8}$$

With respect to $(\cdot,\cdot)_V$, E_D is an isometry from $(G_D, \langle\cdot,\cdot\rangle_{D,\star})$ onto its range in $(V, (\cdot,\cdot)_V)$, i.e., $\|E_D g\|_V = \|g\|_{D,\star}$ for all $g \in G_D$. Furthermore, for any finite-dimensional subspace $E_r \subset G_D$, if $P_{E_r} : G_D \to E_r$ denotes the $\langle\cdot,\cdot\rangle_{D,\star}$-orthogonal projection and $P_{E_D(E_r)} : V \to E_D(E_r)$ the $(\cdot,\cdot)_V$-orthogonal projection, then

$$E_D(P_{E_r} g) \ = \ P_{E_D(E_r)}(E_D g), \qquad \forall g \in G_D.$$

Writing the solution of (2.1) as $u(\mathbf{k}) = w(\mathbf{k}) + E_D[g_D(\mathbf{k})]$ with $w(\mathbf{k}) \in V_0$, the weak problem is equivalent to: for given $\mathbf{k} \in \mathcal{D}$, find $w(\mathbf{k}) \in V_0$ such that

$$a(w(\mathbf{k}), v; \mathbf{k}) = \mathfrak{F}(\mathbf{k})[v] \qquad \forall v \in V_0, \tag{2.9}$$

where the aggregated load functional is

$$\mathfrak{F}(\mathbf{k})[v] := \ell(v; \mathbf{k}) - a(E_D[g_D(\mathbf{k})], v; \mathbf{k}), \qquad \forall v \in V_0. \tag{2.10}$$

By Assumption 2.1 and the construction of E_D as in (2.7), there exists $C_F > 0$ such that

$$|\mathfrak{F}(\mathbf{k})[v]| \le C_F \|v\|_V,$$

for all $\mathbf{k} \in \mathcal{D}$ and $v \in V_0$. In the sequel, we work with (2.9).

2.2. Finite Element approximation

In the theory of FEM, let $\mathcal{T}_h$ be a shape-regular FE mesh and $V_h \subset V$ a conforming P_1 space. We define the finite-dimensional ansatz spaces $V_h := S_h \cap V$ and $V_{h,0} := S_h \cap V_0$. Let $\{\phi_i\}_{i=1}^{N_h}$ be the standard nodal basis of V_h associated with the mesh nodes $\{\mathbf{x}_i\}_{i=1}^{N_h}$. We split the index set of degrees of freedom as

$$\mathcal{I} := \{\, i : \mathbf{x}_i \notin \Gamma_D \,\}, \qquad \mathcal{B}_D := \{\, i : \mathbf{x}_i \in \Gamma_D \,\},$$

so that the functions $\{\phi_i\}_{i\in\mathcal{I}}$ form a basis of $V_{h,0}$. Denoting $N_0 := |\mathcal{I}|$ and, without loss of generality, relabeling the indices so that $\mathcal{I} = \{1, \dots, N_0\}$, we obtain that $\{\phi_i\}_{i=1}^{N_0}$ is a nodal basis of $V_{h,0}$. Define the assembled matrices and load by

$$[\mathbf{A}(\mathbf{k})]_{ij} := a(\phi_j, \phi_i; \mathbf{k}), \qquad [\mathbf{F}(\mathbf{k})]_i := \ell(\phi_i; \mathbf{k}), \qquad i, j = 1, \dots, N_h,$$

and write the block partition

$$\mathbf{A}(\mathbf{k}) = \begin{bmatrix} \mathbf{A}_{II}(\mathbf{k}) & \mathbf{A}_{IB_D}(\mathbf{k}) \\ \mathbf{A}_{B_DI}(\mathbf{k}) & \mathbf{A}_{B_DB_D}(\mathbf{k}) \end{bmatrix}, \qquad \mathbf{F}(\mathbf{k}) = \begin{bmatrix} \mathbf{F}_I(\mathbf{k}) \\ \mathbf{F}_{B_D}(\mathbf{k}) \end{bmatrix}.$$

The discrete lifting $\mathbf{E}_h : \mathbb{R}^{|\mathcal{B}_D|} \to \mathbb{R}^{N_h}$ corresponding to (2.7) is given by

$$\mathbf{E}_h \mathbf{g}_{\mathcal{B}_D} := \begin{bmatrix} \mathbf{L}_{\text{int}}\, \mathbf{g}_{\mathcal{B}_D} \\ \mathbf{g}_{\mathcal{B}_D} \end{bmatrix}, \quad \text{with} \quad \mathbf{L}_{\text{int}} := -(\mathbf{A}^\star_{II})^{-1} \mathbf{A}^\star_{IB_D}.$$

Here, $\mathbf{g}_{\mathcal{B}_D}$ is the Dirichlet data vector on $\mathcal{B}_D$ and $\mathbf{A}^\star_{II}$, $\mathbf{A}^\star_{IB_D}$ are blocks in $\mathbf{A}^\star := \mathbf{A}(\mathbf{k}_\star)$. The Dirichlet boundary metric induced by the reference energy is

$$\mathbf{W}_\Gamma \;:=\; \mathbf{E}_h^\top \mathbf{A}^\star \mathbf{E}_h,$$

which yields the discrete inner product $\langle \boldsymbol{\xi}, \boldsymbol{\zeta} \rangle_{D,\star} := \boldsymbol{\xi}^\top \mathbf{W}_\Gamma \boldsymbol{\zeta}$ on boundary vectors. The discrete form of (2.10) is given by

$$\widehat{\mathbf{F}}_I(\mathbf{k}) := \mathbf{F}_I(\mathbf{k}) - \mathbf{A}_{II}(\mathbf{k})\, \mathbf{L}_{\text{int}}\, \mathbf{g}_{\mathcal{B}_D}(\mathbf{k}) - \mathbf{A}_{IB_D}(\mathbf{k})\, \mathbf{g}_{\mathcal{B}_D}(\mathbf{k}).$$

The unknown $\mathbf{w}_I(\mathbf{k}) \in \mathbb{R}^{N_0}$ solves

$$\mathbf{A}_{II}(\mathbf{k})\, \mathbf{w}_I(\mathbf{k}) \;=\; \widehat{\mathbf{F}}_I(\mathbf{k}), \tag{2.11}$$

and

$$\mathbf{u}_h(\mathbf{k}) = \begin{bmatrix} \mathbf{w}_I(\mathbf{k}) \\ \mathbf{0} \end{bmatrix} + \mathbf{E}_h \mathbf{g}_{\mathcal{B}_D}(\mathbf{k}).$$

Since $V_{h,0} \subset V_0$, the continuity and coercivity constants of $a(\cdot, \cdot; \mathbf{k})$ carry over to V_h uniformly in h and $\mathbf{k}$. Hence, the discrete problem is well-posed. We assume quadratures of sufficient order such that numerical integration errors are negligible.

3. RB–DeepONet framework

Throughout the paper, we work with the homogeneous, lifted formulation (2.9): for given $\mathbf{k} \in \mathcal{D}$, find $w(\mathbf{k}) \in V_0$ such that

$$a(w(\mathbf{k}), v; \mathbf{k}) = \mathfrak{F}(\mathbf{k})[v] \qquad \forall v \in V_0,$$

with $\mathfrak{F}(\mathbf{k})$ defined in (2.10). The full solution is $u(\mathbf{k}) = w(\mathbf{k}) + E_D[g_D(\mathbf{k})]$. We present a unified RB–DeepONet framework that fixes an RB trunk space and learns the branch coefficients via residual minimization with an offline/online split. Within this framework, we focus on two specific problem cases:

- **Case I: fully parameterized operators and data.** There exist known maps

$$\mathbf{k} \mapsto \{A(\cdot;\mathbf{k}), c_0(\cdot;\mathbf{k}), \beta(\cdot;\mathbf{k}), f(\cdot;\mathbf{k}),\ g_D(\cdot;\mathbf{k}),\ h_N(\cdot;\mathbf{k}),\ r_R(\cdot;\mathbf{k})\}.$$

  Hence, the learning target is the solution operator

$$\mathbf{k} \ \longmapsto\ w(\cdot;\mathbf{k}). \tag{3.1}$$

- **Case II: parametric operators with independently varying data.** Here, we consider a more complicated case, where only the coefficients depend on $\mathbf{k}$, while the data may vary independently. We therefore allow the map

$$\mathbf{k} \mapsto \{A(\cdot;\mathbf{k}), c_0(\cdot;\mathbf{k}), \beta(\cdot;\mathbf{k})\}.$$

  To avoid handling full-order fields online, we compress exogenous data offline into low-dimensional coordinates:

  – source coefficients $\mathbf{a} \in \mathbb{R}^{r_f}$ obtained by projecting the aggregated load onto *source modes*;
  – boundary coefficients $\mathbf{b} \in \mathbb{R}^{r_g}$ obtained by projecting g_D onto *boundary modes*.

  The online learning target is then

$$\mathbf{k}_{\text{aug}} := \left[\mathbf{k}^\top,\ \mathbf{a}^\top,\ \mathbf{b}^\top\right]^\top \ \longmapsto\ w(\cdot;\mathbf{k}_{\text{aug}}). \tag{3.2}$$

Since the FE dimension N_h is typically large, solving the full system (2.11) repeatedly for many $\mathbf{k} \in \mathcal{D}$ becomes prohibitively expensive. Both (3.1) and (3.2) are instances of operator learning, which motivates us to consider using DeepONet to solve them. A naive DeepONet would treat the trunk as an unconstrained black box: the trunk provides learned basis functions, the branch produces coefficients, and the prediction for $w(\mathbf{k})$ is their inner product, i.e., a linear combination of the trunk functions with branch coefficients. While flexible, this representation does not encode any of the underlying PDE or variational structure, and offers little *a priori* control over stability or approximation quality. A more structured idea is therefore to use basis functions that already embed information about the solution manifold as the trunk. At one extreme, one could take the full finite element basis $\{\phi_i\}_{i=1}^{N_0}$ as the trunk [34], but this would require predicting N_0 coefficients, which is both computationally prohibitive and unnecessary when the solution manifold is low-dimensional. POD–DeepONet [33] implements a related idea by replacing the FE basis with a POD trunk, but it still relies on a large ensemble of high-fidelity snapshots to construct that trunk space.

RB spaces are rigorously validated in numerical analysis, which capture parametric solution manifolds with very few modes, and admit both *a priori* and *a posteriori* error estimates. Motivated by these properties, we adopt a reduced-order operator-learning pipeline and propose the RB–DeepONet framework, which couples the DeepONet architecture with RB methodology. Concretely, we construct a low-dimensional reduced space

$$V_{\text{rb}} := \text{span}\{\psi_1, \ldots, \psi_N\} \subset V_0, \qquad N \ll N_h,$$

via Greedy selection. In RB–DeepONet, V_{rb} acts as a fixed, interpretable trunk and the branch network predicts RB coefficients $\mathbf{c}_\theta(\mathbf{k})$, i.e.,

$$\cdot \ \longmapsto\ \mathbf{c}_\theta(\cdot), \tag{3.3}$$

where $\cdot = \mathbf{k}$ in Case I and $\cdot = \mathbf{k}_{\text{aug}}$ in Case II. In practice, the branch network can be implemented using standard architectures such as CNNs, MLPs, or ResNets. In our experiments, the specific architecture is described in Section 5.

Combining the fixed trunk and the output of branch network, the RB–DeepONet prediction is expressed as

$$w_\theta(\mathbf{x};\cdot) := \sum_{i=1}^{N} c_{\theta,i}(\cdot)\,\psi_i(\mathbf{x}) = \Psi(\mathbf{x})\mathbf{c}_\theta(\cdot). \tag{3.4}$$

where $\Psi(\mathbf{x}) = [\psi_1(\mathbf{x}), \ldots, \psi_N(\mathbf{x})]$ denotes the fixed trunk basis, and $\mathbf{c}_\theta = [c_{\theta,1}, \cdots, c_{\theta,N}]^\top \in \mathbb{R}^N$ is the branch network output. This yields an offline–online split whose inference cost depends on the RB dimension N but not on N_h, thereby avoiding repeated full-order solves while preserving quantitative accuracy.

For both cases, the RB-trunk construction and the loss function are identical. The only differences arise in the inputs of the branch and in the assembly of the aggregated load (2.10) during the offline stage. Accordingly, we present the shared material once (under Case I) in Sections 3.1-3.3, and introduce the specific treatment for Case II in Section 3.4.

3.1. RB trunk construction

Our goal is to build a low-dimensional space $V_{\mathrm{rb}} \subset V_0$ from a parameter sample set $\mathcal{S} := \{\mathbf{k}_i\}_{i=1}^{N_k} \subset \mathcal{D}$ such that the Galerkin projection on V_{rb} provides accurate surrogates uniformly over $\mathcal{S}$. We adopt a Greedy selection as the default strategy: the space is grown iteratively, one truth snapshot per iteration, until a certified error indicator falls below a prescribed tolerance. Thus, it requires only N high-fidelity solves to build an N-dimensional RB space. For comparison, we also summarize the POD construction, which needs all N_k snapshots but enjoys an optimality identity. Further details can be found in [2].

First, we choose an initial sample $\mathbf{k}_1 \in \mathcal{S}$, compute the high-fidelity solution $w(\mathbf{k}_1)$, normalized in the V-norm, and initialize $V_{\mathrm{rb}} = \mathrm{span}\{w(\mathbf{k}_1)\}$. Given the current space V_{rb}, for each $\mathbf{k} \in \mathcal{S}$, we compute the reduced approximation $w_{\mathrm{rb}}(\mathbf{k}) \in V_{\mathrm{rb}}$ as the solution of

$$a(w_{\mathrm{rb}}(\mathbf{k}), v; \mathbf{k}) \;=\; \mathfrak{F}(\mathbf{k})[v] \qquad \forall v \in V_{\mathrm{rb}}. \tag{3.5}$$

Define the residual functional and its dual norm by

$$r(v; \mathbf{k}) := \mathfrak{F}(\mathbf{k})[v] - a(w_{\mathrm{rb}}(\mathbf{k}), v; \mathbf{k}), \qquad \|r(\cdot; \mathbf{k})\|_{V_0'} := \sup_{0 \neq v \in V_0} \frac{r(v; \mathbf{k})}{\|v\|_V}. \tag{3.6}$$

With the coercivity lower bound $\alpha_{\mathrm{LB}} > 0$, the *a posteriori* estimator

$$\eta(\mathbf{k}) := \alpha_{\mathrm{LB}}^{-1} \|r(\cdot; \mathbf{k})\|_{V_0'} \tag{3.7}$$

controls the energy error [2]:

$$\|w(\mathbf{k}) - w_{\mathrm{rb}}(\mathbf{k})\|_V \;\leq\; \eta(\mathbf{k}) \qquad \forall\, \mathbf{k} \in \mathcal{D}. \tag{3.8}$$

Thus, we use $\eta(\mathbf{k})$ to do Greedy selection as summarized in Algorithm 1 below.

Algorithm 1: Greedy construction of V_{rb}

Input: parameter sample set $\mathcal{S}$, tolerance $\epsilon_{\mathrm{greedy}} > 0$
Output: RB basis $\{\psi_i\}_{i=1}^{N}$

Set $n = 1$.
Pick $\mathbf{k}_1 \in \mathcal{S}$; compute the truth $w(\mathbf{k}_1)$, normalize in $\|\cdot\|_V$, and set $V_{\mathrm{rb}} = \mathrm{span}\{w(\mathbf{k}_1)\}$.
while $\max_{\mathbf{k}\in\mathcal{S}} \eta(\mathbf{k}) > \epsilon_{\mathrm{greedy}}$ **do**
  For each $\mathbf{k} \in \mathcal{S}$, solve (3.5) in V_{rb} and evaluate $\eta(\mathbf{k})$.
  Set $\mathbf{k}_{n+1} = \arg\max_{\mathbf{k}\in\mathcal{S}} \eta(\mathbf{k})$.
  Compute the truth $w(\mathbf{k}_{n+1})$; V-orthonormalize and enrich $V_{\mathrm{rb}} \leftarrow V_{\mathrm{rb}} \oplus \mathrm{span}\{w(\mathbf{k}_{n+1})\}$.
  Set $n = n + 1$.

As summarized in Algorithm 1, the offline cost is proportional to N truth solves. Under an affine decomposition or an empirical interpolation surrogate, all parameter-independent contributions to (3.5) and (3.7) can be preassembled, so the online evaluation of $\eta(\mathbf{k})$ and the reduced solve scales only with $\dim V_{\mathrm{rb}} = N$, not with the full FE dimension. Moreover, the Greedy algorithm uses the estimator (3.7) as its selection criterion, so the maximal indicator over the sample set decreases monotonically with each enrichment step, providing a built-in measure of convergence of the RB space.

Another way to construct V_{rb} is POD. Let $w^i := w(\mathbf{k}_i) \in V_0$ be the snapshots and $V_{\mathcal{S}} = \mathrm{span}\{w^i : i = 1, \ldots, N_k\}$. Define the correlation operator

$$C(v) := \tfrac{1}{N_k} \sum_{i=1}^{N_k} (v, w^i)_V \, w^i, \qquad v \in V_0, \tag{3.9}$$

and let $\{(\lambda_i, \psi_i)\}_{i=1}^{N_k}$ be its eigenpairs in $V_{\mathcal{S}}$, ordered $\lambda_1 \geq \cdots \geq \lambda_{N_k} \geq 0$ with $\|\psi_i\|_V = 1$. The POD space is $V_{\mathrm{rb}} = \mathrm{span}\{\psi_1, \ldots, \psi_N\}$, and it satisfies the optimality identity

$$\frac{1}{N_k} \sum_{i=1}^{N_k} \|w^i - P_N w^i\|_V^2 \;=\; \sum_{i=N+1}^{N_k} \lambda_i, \tag{3.10}$$

where P_N is the V-orthogonal projector onto V_{rb}. In practice, N is chosen as the smallest integer such that

$$1 - \frac{\sum_{i=1}^{N} \lambda_i}{\sum_{i=1}^{N_k} \lambda_i} \;\le\; \epsilon_{\mathrm{POD}}^2, \tag{3.11}$$

for some tolerance $\epsilon_{\mathrm{POD}} > 0$.

Note that POD requires all N_k snapshots but delivers the best mean-square projection in (3.10). In contrast, the Greedy procedure attains comparable accuracy with significantly fewer truth solves and, with (3.8), provides certified selection on $\mathcal{S}$.

Remark 3.1. *In addition to Greedy selection, we also include POD, since it is the standard mechanism for prescribing a fixed trunk in POD–DeepONet [33]. In that setting, one first computes a set of high-fidelity snapshots, forms the POD modes, and then fixes the trunk to the leading modes while the branch network predicts their coefficients. In our numerical experiments in Section 5, the POD trunk used for POD–DeepONet is constructed exactly in this way.*

In RB–DeepONet, POD is only one possible choice for constructing an interpretable trunk. A certified Greedy procedure can generate a basis of the same size by adaptively sampling parameters using a posteriori *error estimators, thereby reducing the offline cost while retaining rigorous error control [2]. Our numerical results indicate that, for a fixed reduced dimension N, POD- and Greedy-generated trunks yield comparable prediction errors. We therefore report both, using the Greedy basis as the default for offline economy and the POD basis as a convenient baseline.*

3.2. Offline reduced operators

Let $\{\psi_i\}_{i=1}^{N} \subset V_0$ be the reduced basis built by Algorithm 1. Each ψ_i is represented in the FE trial space $V_{h,0}$ with basis $\{\phi_j\}_{j=1}^{N_0}$. We denote this FE representation by $\psi_{h,i} \in V_{h,0}$. Thus there exist coefficient vectors $\mathbf{v}_i \in \mathbb{R}^{N_0}$ such that

$$\psi_{h,i}(\mathbf{x}) = \sum_{j=1}^{N_0} (\mathbf{v}_i)_j \, \phi_j(\mathbf{x}), \qquad i = 1, \dots, N. \tag{3.12}$$

Collecting the columns $\mathbf{v}_i$ yields the reduced basis matrix

$$\Psi := [\, \mathbf{v}_1, \dots, \mathbf{v}_N \,] \in \mathbb{R}^{N_0 \times N}, \qquad N \ll N_0. \tag{3.13}$$

Recall that the full FE solution coefficients $\mathbf{w}_I(\mathbf{k}) \in \mathbb{R}^{N_0}$ satisfy (2.11). The RB–Galerkin approximation seeks $\mathbf{c}_N(\mathbf{k}) = [c_{N,1}(\mathbf{k}), \dots, c_{N,N}(\mathbf{k})]^\top \in \mathbb{R}^N$ such that $\Psi\, \mathbf{c}_N(\mathbf{k}) \approx \mathbf{w}_I(\mathbf{k})$. Galerkin projection on $V_{\mathrm{rb}} = \mathrm{span}\{\psi_i\}_{i=1}^{N}$ gives the reduced system

$$\mathbf{A}_{\mathrm{rb}}(\mathbf{k})\, \mathbf{c}_N(\mathbf{k}) = \mathbf{F}_{\mathrm{rb}}(\mathbf{k}), \tag{3.14}$$

with

$$\mathbf{A}_{\mathrm{rb}}(\mathbf{k}) = \Psi^\top \mathbf{A}_{II}(\mathbf{k}) \Psi, \qquad \mathbf{F}_{\mathrm{rb}}(\mathbf{k}) = \Psi^\top \widehat{\mathbf{F}}_I(\mathbf{k}). \tag{3.15}$$

Under the uniform well-posedness assumptions of the full model, the RB–Galerkin approximation (3.14) is well-defined and admits a unique solution $\mathbf{c}_N(\mathbf{k})$, which is the coefficient vector that our branch network aims to predict.

To make the online cost independent of N_0, we employ an affine decomposition of the FE operators:

$$\mathbf{A}_{II}(\mathbf{k}) = \sum_{p=1}^{Q_a} \Theta_p^a(\mathbf{k})\, \mathbf{A}_p, \qquad \widehat{\mathbf{F}}_I(\mathbf{k}) = \sum_{q=1}^{Q_f} \Theta_q^f(\mathbf{k})\, \mathbf{F}_q, \tag{3.16}$$

where Θ_p^a, Θ_q^f are parameter-dependent scalars and $\mathbf{A}_p, \mathbf{F}_q$ are parameter-independent FE quantities. In the offline stage, we precompute and store the reduced components

$$\mathbf{A}_p^N := \Psi^\top \mathbf{A}_p \Psi, \qquad \mathbf{F}_q^N := \Psi^\top \mathbf{F}_q, \qquad p = 1, \dots, Q_a,\ q = 1, \dots, Q_f. \tag{3.17}$$

Then, at query time, the reduced system (3.14) is assembled as

$$\mathbf{A}_{\mathrm{rb}}(\mathbf{k}) = \sum_{p=1}^{Q_a} \Theta_p^a(\mathbf{k})\, \mathbf{A}_p^N, \qquad \mathbf{F}_{\mathrm{rb}}(\mathbf{k}) = \sum_{q=1}^{Q_f} \Theta_q^f(\mathbf{k})\, \mathbf{F}_q^N, \tag{3.18}$$

at cost $O(Q_a N^2 + Q_f N)$, independent of N_0. If an exact affine form is not available, empirical interpolation (EIM/DEIM) can be used to approximate (3.16) [2, 47, 48]. This non-affine setting is instantiated in our numerical experiments (Example 5.3), where EIM is employed to approximate the parameter dependence, thereby demonstrating the broad applicability of the proposed offline–online framework.

3.3. Training and inference

In the online stage, we feed the branch network with $\mathbf{k}$ and obtain the RB coefficients $\mathbf{c}_\theta(\mathbf{k}) \in \mathbb{R}^N$. With the fixed trunk $\Psi(\mathbf{x})$, the RB–DeepONet reconstructions are given in (3.4), and the prediction reads

$$u_\theta(\mathbf{x};\mathbf{k}) = w_\theta(\mathbf{x};\mathbf{k}) + E_D(g_D(\mathbf{x};\mathbf{k})).$$

Rather than data-driven training, we employ the variational formulation and enforce physics through the residual projected onto the RB space. Using the RB stiffness $\mathbf{A}_{\mathrm{rb}}(\mathbf{k})$ and load $\mathbf{F}_{\mathrm{rb}}(\mathbf{k})$ defined in (3.18), we collect the RB residual tested against $\{\psi_i\}_{i=1}^N$ in the vector

$$\mathbf{r}(\mathbf{k}) := \mathbf{F}_{\mathrm{rb}}(\mathbf{k}) - \mathbf{A}_{\mathrm{rb}}(\mathbf{k})\mathbf{c}_\theta(\mathbf{k}) \in \mathbb{R}^N. \tag{3.19}$$

Given a parameter set $\mathcal{B}_s = \{\mathbf{k}_i\}_{i=1}^{N_s}$, the loss function is then defined as

$$\mathcal{L}(\theta) = \frac{1}{N_s}\sum_{i=1}^{N_s} \|\mathbf{A}_{\mathrm{rb}}(\mathbf{k}_i)^{-1/2}\mathbf{r}(\mathbf{k}_i)\|_2^2, \tag{3.20}$$

This is exactly the intrusive Galerkin equilibrium condition projected onto the reduced basis. Thus, minimizing (3.20) drives the RB-projected residual to zero. In particular, if $\mathcal{L}(\theta) = 0$, then $\mathbf{r}(\mathbf{k}_i) = \mathbf{0}$, and $\mathbf{c}_\theta(\mathbf{k}_i)$ coincides with the RB–Galerkin coefficients $c_N(\mathbf{k}_i)$ defined in (3.14), for all $i = 1, \cdots, N_s$.

The overall procedure for RB–DeepONet is summarized in Algorithm 2 below.

Algorithm 2: RB–DeepONet for parametric PDEs (fixed trunk, learned branch)

Input: Parameter samples $\mathcal{S}$; epochs T; batch size N_s; Greedy tolerance $\epsilon_{\mathrm{greedy}}$.
Output: Trained surrogate $u_\theta(\mathbf{x};\mathbf{k})$.

Offline (RB-trunk and reduced operators).
Construct RB basis $\{\psi_i\}_{i=1}^N$ by Algorithm 1, and form $\Psi(\mathbf{x}) := [\psi_1(\mathbf{x}), \ldots, \psi_N(\mathbf{x})]$.
Preassemble all parameter-independent reduced operators

Training (operator learning).
Fix $\Psi(\mathbf{x})$ as trunk; initialize branch net $\mathbf{c}_\theta$.
for $t = 1$ **to** T **do**
    Shuffle $\mathcal{S}$ and split it into batches $\{\mathcal{B}_s\}_{s=1}^S$ of size N_s.
    for $s = 1$ **to** S **do**
        Update the branch network by minimizing $\mathcal{L}(\theta)$ defined in (3.20).

Online (prediction).
Given new $\mathbf{k} \in \mathcal{D}$, output $w_\theta(\mathbf{x};\mathbf{k}) = \Psi(\mathbf{x})\mathbf{c}_\theta(\mathbf{k})$.
Formulate $u_\theta(\mathbf{x};\mathbf{k}) = w_\theta(\mathbf{x};\mathbf{k}) + E_D[g_D(\mathbf{x};\mathbf{k})]$.

3.4. Extension to independent boundary and source data

Recall that in Case II we assume only the operator coefficients $A(\cdot;\mathbf{k}), c_0(\cdot;\mathbf{k}), \beta(\cdot;\mathbf{k})$ depend on $\mathbf{k}$, whereas the data (f, g_D, h_N, r_R) are allowed to vary inpedently. The construction of trunk basis has the same procedure as we introduced in Section 3.1 and the training procedure is identical to Algorithm 2. The only changes are the branch input and the assembly of the reduced right–hand side $\mathbf{F}_{\mathrm{rb}}(\mathbf{k})$.

3.4.1. Boundary and source modes construction

To avoid carrying full-order fields online, we construct, in the offline stage, two compact data-driven spaces from snapshot ensembles: a boundary space $E_{r_g} \subset G_D$ for Dirichlet traces and a source space $W_{r_f} \subset V_0$ for the aggregated load functional. At inference time, we work only with the modal coordinates $\mathbf{b} \in \mathbb{R}^{r_g}$ and $\mathbf{a} \in \mathbb{R}^{r_f}$, obtained by projecting the data onto E_{r_g} and W_{r_f}. We adopt a certified Greedy construction. As in Algorithm 1, we start from a rank–one initial space by a randomly chosen function in the corresponding function space. At each iteration, we select modes by maximizing certified error indicators

Let $\mathcal{G} = \{g^{(m)}\}_{m=1}^{N_g} \subset G_D$ be a training set of boundary traces. Given the current boundary subspace $E_r \subset G_D$ and its orthogonal projector P_{E_r} in $\langle\cdot,\cdot\rangle_{D,\star}$, we define the *a posteriori* estimator

$$\eta_g(m) := \left\| g^{(m)} - P_{E_r} g^{(m)} \right\|_{D,\star}. \tag{3.21}$$

The Greedy update selects $m^\star = \arg\max_m \eta_g(m)$. The corresponding function $g^{(m^\star)}$ is orthonormalized in $\langle\cdot,\cdot\rangle_{D,\star}$ and appended to the boundary space E_{r+1}. The procedure is repeated until $\max_m \eta_g(m) \le \epsilon_g$ for a prescribed tolerance ϵ_g. The resulting *boundary modes* are denoted by $\{\eta_n\}_{n=1}^{r_g}$, which span the space E_{r_g}. Their corresponding liftings are defined as $\mathcal{E}_n := E_D[\eta_n]$.

For any parameter $\mathbf{k}$, let $u(\mathbf{k})$ be the solution with datum $g^{(m)}$, and let $u^{(g_r)}(\mathbf{k})$ be the solution with $g_r := P_{E_r} g^{(m)}$. By the lifting isometry and the coercivity of $a(\cdot,\cdot;\mathbf{k})$, there exists $C_g := 1 + M\alpha_{\mathrm{LB}}^{-1}$ such that

$$\left\| u(\mathbf{k}) - u^{(g_r)}(\mathbf{k}) \right\|_V \;\le\; C_g\, \eta_g(m) \qquad \forall\, \mathbf{k} \in \mathcal{D},\ \forall\, m \in \{1, \cdots, N_g\}. \tag{3.22}$$

To obtain *source modes*, let $\mathcal{F} = \{\mathfrak{F}^{(n)}\}_{n=1}^{N_f} \subset V_0'$ be a training set of aggregated load functionals, which is built from representative f, g_D, h_N, r_R. We define the Riesz map $\mathcal{R}_\star : V_0 \to V_0'$ by

$$(\mathcal{R}_\star q)[v] := a(q, v; \mathbf{k}_\star) = (q, v)_V.$$

Following Riesz representation theorem [49], $\mathcal{R}_\star$ is an isometric isomorphism. Hence, for each aggregated load functional $\mathfrak{F}^{(n)} \in V_0'$, there exists a unique Riesz representer $q^{(n)} = \mathcal{R}_\star^{-1}\mathfrak{F}^{(n)} \in V_0$ satisfying

$$a(q^{(n)}, v; \mathbf{k}_\star) = \mathfrak{F}^{(n)}[v] \qquad \forall v \in V_0. \tag{3.23}$$

Approximating $\mathfrak{F}^{(n)}$ in the dual norm $\|\cdot\|_{V_0'}$ is therefore equivalent to approximating its representer $q^{(n)}$ in the V–norm:

$$\min_{w \in W_r} \|\mathfrak{F}^{(n)} - \mathcal{R}_\star w\|_{V_0'} \;=\; \min_{w \in W_r} \|q^{(n)} - w\|_V.$$

Thus, we project functionals by first mapping them to q via (3.23) and then working in the primal space V_0. Let $W_r \subset V_0$ be the current source subspace and P_{W_r} the $(\cdot,\cdot)_V$–orthogonal projector, we define the *a posteriori* estimator

$$\eta_f(n) := \left\| q^{(n)} - P_{W_r} q^{(n)} \right\|_V = \min_{w \in W_r} \left\| q^{(n)} - w \right\|_V = \min_{w \in W_r} \left\| \mathfrak{F}^{(n)} - \mathcal{R}_\star w \right\|_{V_0'}. \tag{3.24}$$

Then, $\eta_f(n)$ is precisely the best-approximation error of the functional in $\|\cdot\|_{V_0'}$. The greedy update selects $n^\star = \arg\max_n \eta_f(n)$. The corresponding residual is orthonormalized in V–norm and appended to W_r. The iteration terminates once $\max_n \eta_f(n) \le \epsilon_f$ for a prescribed tolerance ϵ_f. The resulting source modes are $\{W_f^{(m)}\}_{m=1}^{r_f}$, which span W_{r_f}.

For any parameter $\mathbf{k}$, let $w(\mathbf{k}) \in V_0$ solve the homogeneous problem with load $\mathfrak{F}^{(n)}$ and let $w_r(\mathbf{k}) \in V_0$ solve the same problem with projected load $(P_{W_r} q^{(n)}, \cdot)_V$. Then, using the uniform coercivity of $a(\cdot,\cdot;\mathbf{k})$, we obtain the reliability bound

$$\left\| w(\mathbf{k}) - w_r(\mathbf{k}) \right\|_V \;\le\; \alpha_{\mathrm{LB}}^{-1}\, \eta_f(n) \qquad \forall\, \mathbf{k} \in \mathcal{D},\ \forall\, n \in \{1, \cdots, N_f\}. \tag{3.25}$$

Combining (3.22) and (3.25) shows that, for tolerances ϵ_g, ϵ_f, the boundary and source projections contribute at most $O(\epsilon_g)$ and $O(\epsilon_f)$, respectively, to the V–norm error for all $k \in D$ and for all training snapshots $g^{(m)} \in G$, $F^{(n)} \in F$. Consequently, we feed only the projection coordinates into the network, thereby avoiding any manipulation of full-order vectors or matrices during training and inference. On these offline snapshot sets, the induced modeling error is bounded by $O(\epsilon_g + \epsilon_f)$ and becomes negligible once r_g, r_f are chosen sufficiently large.

Remark 3.2 (Projection error for unseen data)**.** *For a new boundary datum $\tilde{g} \in G_D$ and a new aggregated load $\tilde{F} \in V_0'$, the modeling error induced by the boundary and source projections is exactly the distance of the data to the spaces E_{r_g} and W_{r_f}. If the admissible boundary and source data lie in the closures of* span($\mathcal{G}$) *and* span($\mathcal{F}$) *(for instance when they are generated by smooth parametric families and the snapshot sets sample the corresponding parameter domains densely), then increasing r_g and r_f drives these distances, and hence the projection error, to zero. In our numerical experiment (Example 5.2), all evaluation data are drawn from the same data families as the offline snapshots, so the observed projection errors are of the same order as predicted above.*

3.4.2. Modal encoding and RB right-hand-side assembly

By the discussion above, we construct a boundary subspace $E_{r_g} = \mathrm{span}\{\eta_n\}_{n=1}^{r_g} \subset G_D$ and a source subspace $W_{r_f} = \mathrm{span}\{W_f^{(m)}\}_{m=1}^{r_f} \subset V_0$. In the online stage, rather than supplying the branch with full-order data (f, g_D, h_N, r_R), we use their modal coordinates obtained by projecting g_D and the source functional onto E_{r_g} and W_{r_f}, respectively. This encoding yields a strict offline/online split and substantially reduces memory and latency. Although truncation of the data modes introduces an additional error, the *a priori* estimates(3.22), (3.25) show that it is controlled by the chosen dimensions (r_g, r_f) and can be made arbitrarily small.

Define the vector $\mathbf{b} := [b_1, \cdots, b_{r_g}]^\top \in \mathbb{R}^{r_g}$ by

$$b_n \;:=\; \langle\, g_D,\, \eta_n \,\rangle_{D,\star}, \qquad n = 1, \ldots, r_g, \tag{3.26}$$

i.e., the orthogonal projection of g_D onto E_{r_g} in $\langle\cdot,\cdot\rangle_{D,\star}$. Since $\{W_f^{(m)}\}_{m=1}^{r_f}$ is orthonormal in $\langle\cdot,\cdot\rangle_V$, applying (3.23), the coefficients $\mathbf{a} := [a_1, \cdots, a_{r_f}]^\top \in \mathbb{R}^{r_f}$ are obtained by

$$a_m \;:=\; \mathfrak{F}(\mathbf{k})\Big[\, W_f^{(m)} \,\Big], \qquad m = 1, \ldots, r_f. \tag{3.27}$$

Recall that the reduced operator $\mathbf{A}_{\mathrm{rb}}(\mathbf{k})$ is assembled via the affine/EIM expansion introduced earlier in (3.18). For Case II, the reduced right–hand side should make use of the source and boundary modes as well as their coordinate vectors:

$$\mathbf{F}_{\mathrm{rb}}(\mathbf{k}_{\mathrm{aug}}) \;:=\; \underbrace{\sum_{m=1}^{r_f} a_m(\mathbf{k})\, \mathbf{F}_{m,\mathrm{rb}}^{(s)}}_{\text{source contribution}} \;-\; \underbrace{\sum_{n=1}^{r_g} b_n \Big(\sum_{p=1}^{Q_a} \Theta_p^{(a)}(\mathbf{k})\, \mathbf{G}_{n,\mathrm{rb}}^{(p)} \Big)}_{\text{Dirichlet lifting}}, \tag{3.28}$$

where each term is precomputable offline:

$$\mathbf{F}_{m,\mathrm{rb}}^{(s)} \;:=\; \Psi^\top \mathbf{F}(W_f^{(m)}) \in \mathbb{R}^N, \qquad \mathbf{G}_{n,\mathrm{rb}}^{(p)} \;:=\; \Psi^\top \mathbf{A}_p\, \mathbf{E}_n \in \mathbb{R}^N. \tag{3.29}$$

Here, $\mathbf{E}_n \in \mathbb{R}^{N_0}$ are vectors of the lifted boundary modes $\mathcal{E}_n$ on $\mathcal{I}$. The vector $\mathbf{F}(v) \in \mathbb{R}^{N_0}$ associated with any $v \in V_0$ is the discrete Riesz image $[\mathbf{F}(v)]_i := a(v, \phi_i; \mathbf{k}_\star)$, and satisfies $\Psi^\top \mathbf{F}(v) = \big(a(v, \psi_j; \mathbf{k}_\star)\big)_{j=1}^{N_0}$. Then, the RB approximation solves

$$\mathbf{A}_{\mathrm{rb}}(\mathbf{k})\, c_N(\mathbf{k}_{\mathrm{aug}}) \;=\; \mathbf{F}_{\mathrm{rb}}(\mathbf{k}_{\mathrm{aug}}), \tag{3.30}$$

and we target the RB coefficient vector $c_N(\mathbf{k}) \in \mathbb{R}^N$. Under the uniform continuity and coercivity assumptions on $a(\cdot,\cdot;\mathbf{k})$, (3.30) is well posed.

Remark 3.3. *The vector $F_{\mathrm{rb}}(\mathbf{k}_{\mathrm{aug}})$ in* (3.28) *should be interpreted as an approximate RB right-hand side in general. The "exact" RB right-hand side $F_{\mathrm{rb}}^{\mathrm{exact}}(\mathbf{k})$ depends on the complete data (f, g_D, h_N, r_R). $F_{\mathrm{rb}}(\mathbf{k}_{\mathrm{aug}})$ in* (3.28) *approximates the boundary and source contributions by the projections onto E_{r_g} and W_{r_f}, which can be assembled online from the low-dimensional coordinates* **a**, **b**, *and the affine/EIM coefficients $\Theta_p^{(a)}(\mathbf{k})$, without manipulating any full-order vectors or matrices.*

By the reliability bounds for the boundary and source projections, (3.22)–(3.25)*, we have an error bound of the form*

$$\big\|\mathbf{c}_N(\mathbf{k}_{\mathrm{aug}}) - \mathbf{c}_N^{\mathrm{exact}}(\mathbf{k})\big\| \;\le\; C_{\mathrm{rb}}\,(\epsilon_g + \epsilon_f),$$

for some uniform constant $C_{\mathrm{rb}} > 0$. Here, $\mathbf{c}_N^{\mathrm{exact}}(\mathbf{k})$ denotes the RB coefficient vector obtained by solving the RB–Galerkin system with $F_{\mathrm{rb}}^{\mathrm{exact}}(\mathbf{k})$. As r_g, r_f increase and $\epsilon_g, \epsilon_f \to 0$, the reduced right-hand side and the solution of (3.30) *converge to those of the fully constrained RB–Galerkin system, while strict offline–online separation is preserved.*

We feed the branch network with the concatenated feature vector

$$\mathbf{k}_{\mathrm{aug}} = [\, \mathbf{k}^\top,\; \mathbf{a}^\top,\; \mathbf{b}^\top \,]^\top \in \mathbb{R}^{p+r_f+r_g}, \tag{3.31}$$

and obtain the RB coefficients $\mathbf{c}_\theta(\mathbf{k}_{\mathrm{aug}}) \in \mathbb{R}^N$. The residual load $F_{\mathrm{rb}}(\mathbf{k}_{\mathrm{aug}})$ (3.28) is used to form the loss function (3.20). With the fixed trunk $\Psi = \{\psi_j\}_{j=1}^N$ and precomputed liftings $\{\mathcal{E}_n\}_{n=1}^{r_g}$, the RB–DeepONet prediction reads

$$u_\theta(\mathbf{x}; \mathbf{k}_{\mathrm{aug}}) = w_\theta(\mathbf{x}; \mathbf{k}_{\mathrm{aug}}) + \sum_{n=1}^{r_g} b_n\, \mathcal{E}_n(\mathbf{x}).$$

3.5. Summary and comparison

Figure 1 summarizes the RB–DeepONet pipeline, highlighting the fixed RB trunk, the residual-based training objective, and the Case II feature extraction for $\mathbf{a}, \mathbf{b}$. Table 1 positions our method against POD–DeepONet [33] and FEONet [34]. In POD–DeepONet, the trunk is prescribed by a POD basis computed from labeled solution snapshots, while the branch network takes as input a discrete representation of the PDE data (e.g., coefficient fields, forcing terms, or boundary traces), and outputs the coefficient vector $\mathbf{c}_\theta(\mathbf{k}) \in \mathbb{R}^N$. The network is trained in a fully supervised fashion,

$$\mathcal{L}_{\text{sup}} = \frac{1}{N_s} \sum_{i=1}^{N_s} \left\| u_\theta(\cdot; \mathbf{k}_i) - u_h(\cdot; \mathbf{k}_i) \right\|^2_{L^2(\Omega)}. \tag{3.32}$$

Dirichlet data are enforced exactly by a lifting function E_D, i.e., $u_\theta(\cdot; \mathbf{k}) = \Psi(\mathbf{x})\, \mathbf{c}_\theta(\mathbf{k}) + E_D(g_D)$. FEONet keeps the full FE basis as trunk and predicts the FE coefficient vector $\mathbf{w}_\theta(\mathbf{k}) \in \mathbb{R}^{N_0}$ for each input features (e.g., forcing and/or coefficient fields and boundary values), while minimizing the FE variational residual,

$$\mathcal{L}(\theta) = \frac{1}{N_s} \sum_{i=1}^{N_s} \left\| \mathbf{F}_I(\mathbf{k}_i) - \mathbf{A}_{II}(\mathbf{k}_i)\, \mathbf{w}_\theta(\mathbf{k}_i) \right\|^2_2. \tag{3.33}$$

This formulation enables unsupervised training and guarantees exact enforcement of Dirichlet boundary conditions at the nodal degrees of freedom.

As a comparison, we briefly summarize the complexity profiles of RB–DeepONet, POD–DeepONet, and FEONet.

Offline. All three methods start from the same full-order discretization and require assembling the FE stiffness matrices and load vectors (with affine or empirical interpolation decompositions). Beyond this baseline, POD–DeepONet solves the full-order problem for N_k parameter samples to generate snapshots, and then performs a POD and projection to obtain the POD trunk and the associated reduced operators. Its offline cost is therefore dominated by the N_k truth solves. RB–DeepONet instead requires only N truth solves to construct the RB trunk and its reduced operators, so its additional offline cost scales with N. The offline cost of FEONet is only dominated by the full-order assembly.

Training. Let net(M) denote the cost of one forward/backward pass of the branch network with M-dimensional output, for a fixed hidden architecture (same depth and hidden widths across methods). RB–DeepONet evaluates the PDE residual entirely in the reduced space, so the per-sample cost is net(N) + RB, where RB denotes the reduced residual in $\mathbb{R}^N$ (order N^2), with an additional low-rank term $O(Nr_g + Nr_f)$ in Case II. POD–DeepONet uses the same branch architecture but, at each step, reconstructs the full field and compares it to FEM snapshots, leading to net(N) + full(N_0N) per sample, where full(N_0N) summarizes the full-field reconstruction and supervised loss on N_0 degrees of freedom. FEONet directly predicts a full-order solution in $\mathbb{R}^{N_0}$ and minimizes the FE variational residual, so its per-sample cost is net(N_0) + FE, with FE denoting a full-order FE residual evaluation. For fixed depth and hidden widths, net(M) grows approximately linearly with M, so net(N_0) is larger than net(N) by a factor of order N_0/N. Thus, in the typical regime $N \ll N_0$, FEONet is the most expensive to train, POD–DeepONet lies in between, and RB–DeepONet is the cheapest.

Online prediction. Given a trained model, RB–DeepONet and POD–DeepONet evaluate only the branch network $c_\theta(\mathbf{k}) \in \mathbb{R}^N$, so the network evaluation cost scales as $O(N)$ (or $O(N + r_g + r_f)$ in Case II). FEONet, in contrast, outputs a full-order vector $w_\theta(\mathbf{k}) \in \mathbb{R}^{N_0}$, and its online cost scales linearly with N_0. Reconstructing the full field $w_\theta(\mathbf{k}) = \Psi\, c_\theta(\mathbf{k})$ adds a common $O(N_0N)$ matrix–vector multiplication to both RB–DeepONet and POD–DeepONet, and is therefore omitted from the network-cost comparison in Table 1.

Remark 3.4 (Generality of RB–DeepONet)**.** *For clarity, we develop the method in the prototypical second–order elliptic setting* (2.1)*. However, the algorithm only relies on three ingredients: (A1) a variational formulation (linear or semilinear) that is uniformly well-posed ; (A2) an offline reduced trial space* $\text{span}\{\psi_i\}_{i=1}^N \subset V$ *together with the RB residual operator* $\mathbf{A}_{\text{rb}}(\mathbf{k})$ *obtained by testing with a compatible space; and (A3) an affine or EIM/DEIM surrogate enabling offline–online separation. Whenever (A1)–(A3) hold, the trunk fixing, label–free residual training, and the boundary/source mode treatment carry over verbatim.*

4. Convergence analysis of RB–DeepONet

We now analyze the convergence of the RB–DeepONet prediction $w_\theta(\mathbf{k}) = \Psi\, \mathbf{c}_\theta(\mathbf{k})$ to the reduced-basis Galerkin solution $w_{\text{rb}}(\mathbf{k}) = \Psi\, \mathbf{c}_N(\mathbf{k})$, with the finite element mesh $\mathcal{T}_h$ and the RB space $V_{\text{rb}} = \text{span}\{\psi_i\}_{i=1}^N \subset V_h$ fixed. To

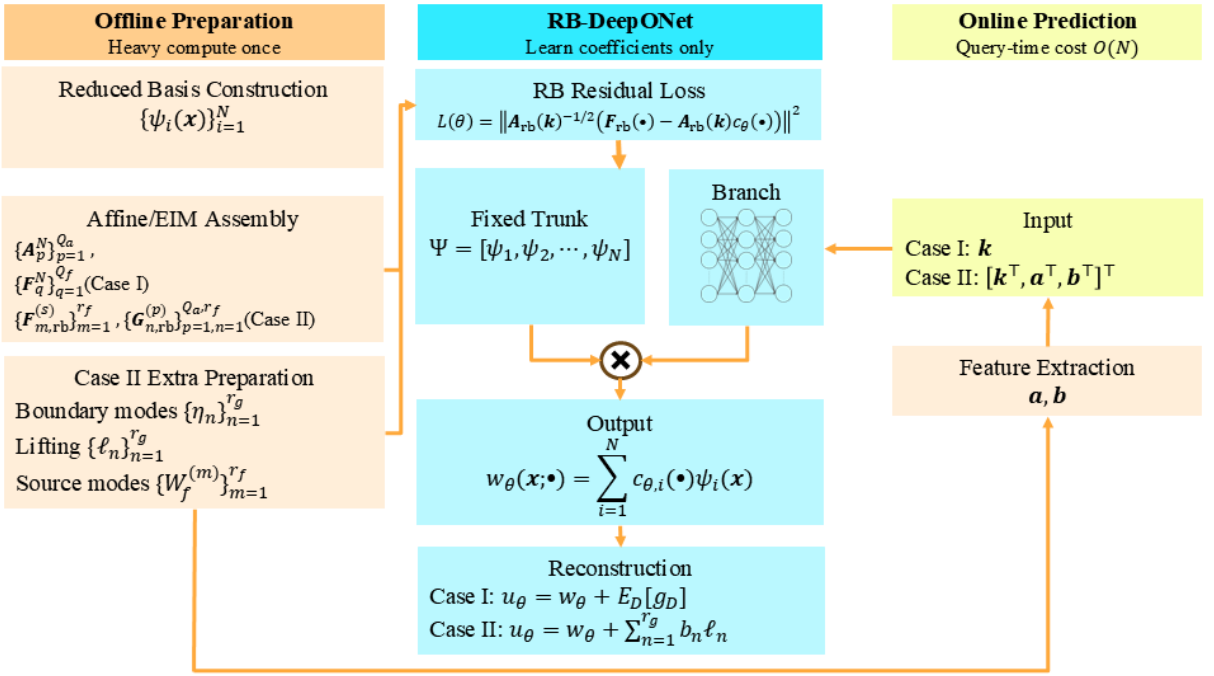


Fig. 1. RB–DeepONet workflow: offline preparation (Greedy RB basis, affine precomputation, and boundary/source modes), residual-based training with a fixed trunk and a branch predicting $\mathbf{c}_\theta$, and online prediction with inputs k (Case I) or [k, a, b] (Case II). In the middle column, $\bullet = k$ (Case I), $\bullet = k_{\mathrm{aug}}$ (Case II).

Table 1. Comparison of RB–DeepONet, POD–DeepONet, and FEONet.

	RB–DeepONet		POD–DeepONet	FEONet
Feature	Case I	Case II	[33]	[34]
Trunk	RB	RB	POD	FE
Input	$\mathbf{k}$	$\mathbf{k}_{\mathrm{aug}}$	task dep.[a]	task dep.[a]
Output	$\mathbf{c}_\theta \in \mathbb{R}^N$	$\mathbf{c}_\theta \in \mathbb{R}^N$	$\mathbf{c}_\theta \in \mathbb{R}^N$	$\mathbf{u}_\theta \in \mathbb{R}^{N_0}$
Training	unsup.	unsup.	sup.	unsup.
Dirichlet BCs	lifting	modes[b]	lifting	nodal
Offline PDE solves	N	N	N_k	0
Training cost (per sample)	net(N)[c]+ RB[d]	net(N) + RB	net(N) + full(N_0N)[e]	net(N_0) + FE[g]
Online prediction cost	$O(N)$	$O(N + r_g + r_f)$	$O(N)$	$O(N_0)$

[a] Task-dependent subset of $\{\mathbf{k}, f, g_D, h_N, r_R\}$.
[b] Approximate via boundary modes (with projection error).
[c] net(M): cost of one forward/backward pass of the branch network with M-dimensional output.
[d] RB: reduced residual evaluation in $\mathbb{R}^N$ (order N^2) with additional $O(Nr_g + Nr_f)$ work in Case II.
[e] full(N_0N): full-field reconstruction and supervised loss on N_0 degrees of freedom.
[f] FE: full-order FE residual evaluation.

simplify the discussion, we focus on Case I, where the PDEs are fully parameterized. For Case II, the process is similar: we only need to replace $\mathbf{k}$ with $\mathbf{k}_{\text{aug}}$ and $\mathbf{F}_{\text{rb}}(\mathbf{k})$ with $\mathbf{F}_{\text{rb}}(\mathbf{k}_{\text{aug}})$, and the only additional modeling error is the boundary and source projections, which is quantified in Section 3.4.1. Note that we regard the reduced basis $\{\psi_i\}_{i=1}^N$ as predetermined by Algorithm 1 on a sufficiently fine finite element mesh $\mathcal{T}_h$ such that the standard reduced basis method already provides an approximation within the desired accuracy. Hence, the error associated with RB space construction itself is not the subject of the following analysis.

Training adopts the RB variational residual (3.19) as the foundation for residual minimization. Accordingly, the loss function (3.20) measures the departure of the residual from zero. Our convergence analysis therefore depends on two key factors:

1. the expressive power of the branch network, which is influenced by its width and depth;
2. the number of parameter samples N_s used to calculate the loss function (3.20).

This separation isolates the network approximation from the RB discretization. We split the total error as

$$w - \widehat{w}_{n,N_s} \;=\; \underbrace{(w - w_{\text{rb}})}_{\text{RB discretization}} \;+\; \underbrace{(w_{\text{rb}} - \widehat{w}_n)}_{\text{approximation}} \;+\; \underbrace{(\widehat{w}_n - \widehat{w}_{n,N_s})}_{\text{generalization}}, \tag{4.1}$$

Here, the first term $(w - w_{\text{rb}})$ is the (fixed) RB discretization error that depends only on the Greedy construction and decays rapidly given the exponential decay of the Kolmogorov width for many elliptic PDE families [11, 9, 10]. The second term $(w_{\text{rb}} - \widehat{w}_n)$ is the network *approximation* error for a class of neural network with parameter size n (e.g., width and depth), and $(\widehat{w}_n - \widehat{w}_{n,N_s})$ is the *generalization* error, i.e., the difference between the idealized neural network solution $\widehat{w}_n$ obtained from exact residual minimization and the practical solution $\widehat{w}_{n,N_s}$ obtained using a finite number N_s of training samples. In the following, we focus on the latter two terms.

Proposition 4.1. *Assume Assumption 2.1 holds. Furthermore, we assume the maps* $\mathbf{k} \mapsto a(\cdot,\cdot;\mathbf{k})$ *and* $\mathbf{k} \mapsto \ell(\cdot;\mathbf{k})$ *are continuous on* $\mathcal{D}$*. Then, for a fixed RB basis* $\Psi = [\psi_1, \dots, \psi_N] \subset V_0$ *as in Section 3.1 and* $\mathbf{A}_{\text{rb}}(\mathbf{k})$, $\mathbf{F}_{\text{rb}}(\mathbf{k})$ *as in* (3.15)*, there exist constants* $0 < \sigma_* \le \sigma^* < \infty$ *and* $C_F > 0$*, independent of* $\mathbf{k}$*, such that*

$$\sigma_* \|\mathbf{z}\|_2^2 \;\le\; \mathbf{z}^\top \mathbf{A}_{\text{rb}}(\mathbf{k})\,\mathbf{z} \;\le\; \sigma^* \|\mathbf{z}\|_2^2, \qquad \|\mathbf{F}_{\text{rb}}(\mathbf{k})\| \;\le\; C_F, \quad \forall\, \mathbf{z} \in \mathbb{R}^N, \quad \forall\, \mathbf{k} \in \mathcal{D}.$$

Consequently, the RB system $\mathbf{A}_{\text{rb}}(\mathbf{k})\,\mathbf{c}_N(\mathbf{k}) = \mathbf{F}_{\text{rb}}(\mathbf{k})$ *is uniformly well posed for all* $\mathbf{k} \in \mathcal{D}$*, and the coefficient map* $\mathbf{k} \mapsto \mathbf{c}_N(\mathbf{k})$ *is continuous on* $\mathcal{D}$*.*

Recall that we work on the compact parameter set $\mathcal{D} \subset \mathbb{R}^p$, endowed with the uniform probability measure ρ, i.e., $d\rho(\mathbf{k}) = |\mathcal{D}|^{-1}\, d\mu(\mathbf{k})$ where μ is the Lebesgue measure and $|\mathcal{D}|$ the volume of $\mathcal{D}$. Before proceeding with the analysis, we introduce the spaces and norms used below. We write $L^2(\Omega)$ for the space of square-integrable functions on Ω with norm

$$\|v\|_{L^2(\Omega)}^2 := \int_\Omega |v(x)|^2\, dx.$$

For a coefficient map $\mathbf{c} : \mathcal{D} \to \mathbb{R}^N$, endowed pointwise with the Euclidean norm $\|\cdot\|_2$ on $\mathbb{R}^N$, we define

$$\|\mathbf{c}\|_{L^2(\mathcal{D};\rho)}^2 := \int_{\mathcal{D}} \|\mathbf{c}(\mathbf{k})\|_2^2\, d\rho(\mathbf{k}).$$

We write $C(\mathcal{D};\mathbb{R}^N)$ for the Banach space of continuous maps $g : \mathcal{D} \to \mathbb{R}^N$ endowed with the sup-norm

$$\|g\|_{C(\mathcal{D})} := \sup_{\mathbf{k}\in\mathcal{D}} \|g(\mathbf{k})\|_2.$$

For a function-valued map $w : \mathcal{D} \to L^2(\Omega)$ that is Bochner measurable, we define

$$\|w\|_{L^2(\mathcal{D};L^2(\Omega))}^2 := \int_{\mathcal{D}} \|w(\mathbf{x};\mathbf{k})\|_{L^2(\Omega)}^2\, d\rho(\mathbf{k}) = \int_{\mathcal{D}} \int_\Omega |w(\mathbf{x};\mathbf{k})|^2\, dx\, d\rho(\mathbf{k}). \tag{4.2}$$

Let $\{\psi_i\}_{i=1}^N \subset L^2(\Omega)$ be the fixed RB basis and define the linear map $\mathcal{T} : \mathbb{R}^N \to L^2(\Omega)$ by $\mathcal{T}(\boldsymbol{\alpha}) := \sum_{i=1}^N \alpha_i \psi_i$. By finite dimensionality,

$$\|\mathcal{T}\| \;\le\; \Big(\sum_{i=1}^N \|\psi_i\|_{L^2(\Omega)}^2\Big)^{1/2} =: C_\psi < \infty.$$

Consequently, for any $\mathbf{c} \in L^2(\mathcal{D};\mathbb{R}^N)$, the map $w(\cdot;\mathbf{k}) := \mathcal{T}(\mathbf{c}(\mathbf{k})) = \sum_{i=1}^N c_i(\mathbf{k})\psi_i$ belongs to $L^2(\mathcal{D};L^2(\Omega))$ and satisfies the lifting estimate

$$\|w\|_{L^2(\mathcal{D};L^2(\Omega))} \;\le\; \|\mathcal{T}\|\,\|\mathbf{c}\|_{L^2(\mathcal{D};\rho)} \;\le\; C_\psi\,\|\mathbf{c}\|_{L^2(\mathcal{D};\rho)}. \tag{4.3}$$

Given a function class $\mathcal{F} \subset \{f : \mathcal{D} \to \mathbb{R}\}$ and a probability measure P on $\mathcal{D}$, we define

$$\|f-g\|_{L_2(P)} \;:=\; \Big(\int_{\mathcal{D}} |f(\mathbf{k})-g(\mathbf{k})|^2\,dP(\mathbf{k})\Big)^{1/2}.$$

For a sample $S = (\mathbf{k}_1,\dots,\mathbf{k}_{N_s})$, the corresponding empirical norm is

$$\|f-g\|_{L_2(P_S)} \;:=\; \Big(\tfrac{1}{N_s}\sum_{i=1}^{N_s} |f(\mathbf{k}_i)-g(\mathbf{k}_i)|^2\Big)^{1/2}.$$

Given any coefficient map $\mathbf{c} : \mathcal{D} \to \mathbb{R}^N$, we now define the population loss

$$\mathcal{L}^{\mathrm{pop}}(c) := \int_{\mathcal{D}} \big\|\mathbf{A}_{\mathrm{rb}}(\mathbf{k})\,\mathbf{c}(\mathbf{k}) - \mathbf{F}_{\mathrm{rb}}(\mathbf{k})\big\|_2^2\,d\rho(\mathbf{k}), \tag{4.4}$$

which represents the expected residual error with respect to the uniform distribution on $\mathcal{D}$. In practice, we approximate the population loss (4.4) by Monte–Carlo integration using i.i.d. samples $\{\mathbf{k}_j\}_{j=1}^{N_s} \overset{\text{i.i.d.}}{\sim} \rho$. This yields the empirical loss function

$$\mathcal{L}^{N_s}(\mathbf{c}) := \frac{1}{N_s}\sum_{j=1}^{N_s} \big\|\mathbf{A}_{\mathrm{rb}}(\mathbf{k}_j)\mathbf{c}(\mathbf{k}_j) - \mathbf{F}_{\mathrm{rb}}(\mathbf{k}_j)\big\|_2^2\,. \tag{4.5}$$

Note that by the uniform spectral bounds on $\mathbf{A}_{\mathrm{rb}}(\mathbf{k})$, the preconditioned loss (3.20) and the unweighted residual loss (4.5) are uniformly equivalent; in particular, they share the same population minimizer.

Then, we have the following proposition.

Proposition 4.2. *Under Proposition 4.1, the unique minimizer of* (4.4) *over* $C(\mathcal{D};\mathbb{R}^N)$ *is the continuous RB coefficient map* $\mathbf{c}_N(\mathbf{k}) = (\mathbf{A}_{\mathrm{rb}}(\mathbf{k}))^{-1}\mathbf{F}_{\mathrm{rb}}(\mathbf{k})$ *for* $\mathbf{k} \in \mathcal{D}$*, i.e.,*

$$\mathbf{c}_N = \arg\min_{\mathbf{c}\in C(\mathcal{D};\mathbb{R}^N)} \mathcal{L}^{\mathrm{pop}}(\mathbf{c}). \tag{4.6}$$

Moreover, for any $\mathbf{c} \in C(\mathcal{D};\mathbb{R}^N)$*,*

$$\mathcal{L}^{\mathrm{pop}}(\mathbf{c}) \;\ge\; \sigma_*^2\,\|\mathbf{c}-\mathbf{c}_N\|^2_{L^2(\mathcal{D};\rho)}. \tag{4.7}$$

Proof. Fix $\mathbf{k} \in \mathcal{D}$ and define

$$f_{\mathbf{k}}(\mathbf{z}) := \|\mathbf{A}_{\mathrm{rb}}(\mathbf{k})\mathbf{z} - \mathbf{F}_{\mathrm{rb}}(\mathbf{k})\|_2^2,$$

where $\mathbf{z} \in \mathbb{R}^N$. Since $\mathbf{A}_{\mathrm{rb}}(\mathbf{k})$ is invertible, $f_{\mathbf{k}}$ is a strictly convex quadratic with unique minimizer $(\mathbf{A}_{\mathrm{rb}}(\mathbf{k}))^{-1}\mathbf{F}_{\mathrm{rb}}(\mathbf{k})$, which proves (4.6). For any $\mathbf{c} \in C(\mathcal{D};\mathbb{R}^N)$, a pointwise identity holds:

$$f_{\mathbf{k}}(\mathbf{c}(\mathbf{k})) - f_{\mathbf{k}}(\mathbf{c}_N(\mathbf{k})) = \big\|\mathbf{A}_{\mathrm{rb}}(\mathbf{k})(\mathbf{c}(\mathbf{k})-\mathbf{c}_N(\mathbf{k}))\big\|_2^2.$$

Integrating over $\mathcal{D}$ yields

$$\begin{aligned}\mathcal{L}^{\mathrm{pop}}(\mathbf{c}) - \mathcal{L}^{\mathrm{pop}}(\mathbf{c}_N) =& \big\|\mathbf{A}_{\mathrm{rb}}(\mathbf{k})(\mathbf{c}-\mathbf{c}_N)\big\|^2_{L^2(\mathcal{D};\rho)}\\ \ge& \sigma_*^2\|\mathbf{c}-\mathbf{c}_N\|^2_{L^2(\mathcal{D};\rho)},\end{aligned} \tag{4.8}$$

which proves (4.7). □

Let $\{\mathcal{N}_n\}_{n\in\mathbb{N}} \subset C(\mathcal{D};\mathbb{R}^N)$ be a nested family of vector-valued neural-network hypothesis classes. For each n, fix a network architecture of size n (e.g., increasing width/depth) and let $\Theta_n \subset \mathbb{R}^{P_n}$ denote its parameter space. Define

$$\mathcal{N}_n \;:=\; \{\,\mathbf{c}_\theta : \mathcal{D} \to \mathbb{R}^N \mid \theta \in \Theta_n\,\}.$$

To quantify the expressive capacity of such network families, we first recall the notion of pseudo-dimension, and then impose the structural assumptions summarized in Assumption 4.1 below.

Definition 4.1 (Pseudo-dimension [50, 51, 52])**.** *Let $\mathcal{F}$ be a class of real-valued functions on a set $\mathcal{X}$. The pseudo-dimension $Pdim(\mathcal{F})$ is the largest integer m for which there exist points $\{x_i\}_{i=1}^m \subset \mathcal{X}$ and thresholds $\{y_i\}_{i=1}^m \subset \mathbb{R}$ such that for every $\mathbf{b} = (b_1, \dots, b_m) \in \{0,1\}^m$ there exists $f \in \mathcal{F}$ with*

$$\mathbf{1}\{f(x_i) \geq y_i\} = b_i, \qquad i = 1, \dots, m.$$

The pseudo-dimension provides a measure of the complexity of neural networks. In particular, it is directly connected to uniform convergence properties and generalization error bounds (see, e.g., [50, 51]). Then, we assume the following standard NN structural conditions.

Assumption 4.1. *The classes $\{\mathcal{N}_n\}_{n\in\mathbb{N}}$ satisfy:*

1. ***Nestedness:*** *$\mathcal{N}_n \subset \mathcal{N}_{n+1}$ for all $n \in \mathbb{N}$.*
2. ***Uniform boundedness:*** *There exists $M_\infty > 0$ such that*

$$sup_{g\in\cup_n \mathcal{N}_n} \|g\|_{C(\mathcal{D})} \leq M_\infty.$$

3. ***Density:*** *The union $\bigcup_n \mathcal{N}_n$ is dense in the closed ball $B_{M_\infty} := \{g \in C(\mathcal{D};\mathbb{R}^N) : \|g\|_{C(\mathcal{D})} \leq M_\infty\}$, i.e., for every $g^* \in C(\mathcal{D};\mathbb{R}^N)$ with $\|g^*\|_{C(\mathcal{D})} \leq M_\infty$,*

$$\lim_{n\to\infty} \inf_{g\in\mathcal{N}_n} \|g - g^*\|_{C(\mathcal{D})} = 0.$$

4. ***Finite pseudo-dimension:*** *For each n, $p_n := Pdim(\mathcal{N}_n) < \infty$.*

The above assumption follows [53, 34], which provides the minimal analytic framework for establishing both approximation and generalization properties of the neural network. In particular, conditions (1)–(3) ensure the representational capability of the networks through nestedness, uniform boundedness, and density, while condition (4) controls their statistical complexity via the pseudo-dimension. Together with Proposition 4.1, these requirements guarantee that the network classes $\mathcal{N}_n$ enjoy the universal approximation property [53, Theorem 2.2]:

Theorem 4.1. *Under Proposition 4.1 and Assumption 4.1, we have*

$$\lim_{n\to\infty} \inf_{\mathbf{c}_n\in\mathcal{N}_n} \|\mathbf{c}_n - \mathbf{c}_N\|_{C(\mathcal{D})} = 0. \tag{4.9}$$

Given a neural-network hypothesis class $\mathcal{N}_n \subset C(\mathcal{D};\mathbb{R}^N)$, we define the population risk minimizer and the associated RB approximation

$$\widehat{\mathbf{c}}_n = \arg\min_{\mathbf{c}\in\mathcal{N}_n} \mathcal{L}^{\text{pop}}(\mathbf{c}), \qquad \widehat{w}_n(\mathbf{x};\mathbf{k}) = \sum_{i=1}^N \widehat{\mathbf{c}}_{n,i}(\mathbf{k})\,\psi_i(\mathbf{x}). \tag{4.10}$$

Here $\widehat{\mathbf{c}}_{n,i}(\mathbf{k})$ is the i-th component of $\widehat{\mathbf{c}}_n(\mathbf{k})$. The discrete residual minimization problem on the network class $\mathcal{N}_n$ and the corresponding RB–DeepONet prediction is

$$\widehat{\mathbf{c}}_n^{N_s} := \arg\min_{\mathbf{c}\in\mathcal{N}_n} \mathcal{L}^{N_s}(\mathbf{c}), \qquad \widehat{w}_{n,N_s}(\mathbf{x};\mathbf{k}) := \sum_{i=1}^N \widehat{\mathbf{c}}_{n,i}^{N_s}(\mathbf{k})\,\psi_i(\mathbf{x}). \tag{4.11}$$

Remark 4.2 (On existence of minimizers)**.** *Since $\mathcal{N}_n$ is not assumed to be compact in $C(D;\mathbb{R}^N)$, the minima in* (4.10)–(4.11) *need not be attained* a priori. *Throughout the analysis, we therefore interpret $\widehat{\mathbf{c}}_n$ and $\widehat{\mathbf{c}}_n^{N_s}$ as (possibly approximate) global minimizers, i.e. elements of $\mathcal{N}_n$ whose risks are arbitrarily close to the infima over $\mathcal{N}_n$. This is standard in learning-theoretic analyses and allows us to focus on approximation and generalization errors rather than optimization errors.*

Our first convergence result is presented below:

Theorem 4.3. *Under Proposition 4.1 and Assumptions 4.1,*

$$\lim_{n\to\infty} \|\widehat{\mathbf{c}}_n - \mathbf{c}_N\|_{L^2(\mathcal{D};\rho)} = 0, \qquad \lim_{n\to\infty} \|\widehat{w}_n - w_{\text{rb}}\|_{L^2(\mathcal{D};L^2(\Omega))} = 0, \tag{4.12}$$

where $w_{\text{rb}}(\cdot;\mathbf{k}) = \sum_{i=1}^N \mathbf{c}_{N,i}(\mathbf{k})\,\psi_i(\cdot)$.

Proof. By Proposition 4.2, for any $\mathbf{c} \in C(\mathcal{D}; \mathbb{R}^N)$,

$$\mathcal{L}^{\text{pop}}(\mathbf{c}) \geq \sigma_*^2 \, \|\mathbf{c} - \mathbf{c}_N\|^2_{L^2(\mathcal{D};\rho)}. \tag{4.13}$$

Evaluating at $\widehat{\mathbf{c}}_n$, then using (4.10) and Proposition 4.1,

$$\begin{aligned} \|\widehat{\mathbf{c}}_n - \mathbf{c}_N\|^2_{L^2(\mathcal{D};\rho)} &\leq \sigma_*^{-2} \, \mathcal{L}^{\text{pop}}(\widehat{\mathbf{c}}_n) \;=\; \sigma_*^{-2} \inf_{\mathbf{c}\in\mathcal{N}_n} \mathcal{L}^{\text{pop}}(\mathbf{c}) \\ &= \sigma_*^{-2} \inf_{\mathbf{c}\in\mathcal{N}_n} \|\mathbf{A}_{\text{rb}}(\mathbf{k})\,(\mathbf{c} - \mathbf{c}_N)\,\|^2_{L^2(\mathcal{D};\rho)} \\ &\leq \Big(\frac{\sigma^*}{\sigma_*}\Big)^2 \inf_{\mathbf{c}\in\mathcal{N}_n} \|\mathbf{c} - \mathbf{c}_N\|^2_{L^2(\mathcal{D};\rho)}. \end{aligned} \tag{4.14}$$

By Theorem 4.1, we obtain $\|\widehat{\mathbf{c}}_n - \mathbf{c}_N\|_{L^2(\mathcal{D};\rho)} \to 0$ as $n \to \infty$. Then the function error is derived by the boundedness of the linear map $\mathcal{T}$, which proves (4.12). □

In order to control the *generalization* error between the empirical and population loss, i.e., (4.4) and (4.5), we first introduce the notion of Rademacher complexity.

Definition 4.2 (Rademacher complexity [54])**.** *Let $\mathcal{D} \subset \mathbb{R}^p$ be compact with probability measure ρ. Given a sample $S = (\mathbf{k}_1, \ldots, \mathbf{k}_{N_s}) \overset{i.i.d.}{\sim} \rho$ and i.i.d. Rademacher variables $\varepsilon_1, \ldots, \varepsilon_{N_s}$ with $\mathbb{P}(\varepsilon_i = \pm 1) = \frac{1}{2}$, consider the real-valued loss class*

$$g_{\mathbf{c}}(\mathbf{k}) := \Big\|\mathbf{A}_{\text{rb}}(\mathbf{k})\,\mathbf{c}(\mathbf{k}) - \mathbf{F}_{\text{rb}}(\mathbf{k})\Big\|_2^2, \qquad \mathcal{G}_n := \{\, g_{\mathbf{c}} : \; \mathbf{c} \in \mathcal{N}_n \,\}.$$

The empirical *Rademacher complexity of $\mathcal{G}_n$ on the fixed sample S is*

$$\widehat{R}_S(\mathcal{G}_n) := \mathbb{E}_\varepsilon\left[\sup_{g\in\mathcal{G}_n} \frac{1}{N_s} \sum_{i=1}^{N_s} \varepsilon_i \, g(\mathbf{k}_i) \right].$$

The expected *Rademacher complexity with sample size N_s is*

$$R_{N_s}(\mathcal{G}_n) := \mathbb{E}_S\big[\widehat{R}_S(\mathcal{G}_n)\big] = \mathbb{E}_{S,\varepsilon}\left[\sup_{g\in\mathcal{G}_n} \frac{1}{N_s} \sum_{i=1}^{N_s} \varepsilon_i \, g(\mathbf{k}_i) \right].$$

In this setting, the empirical and population losses can be written compactly as

$$\mathcal{L}^{N_s}(\mathbf{c}) = \frac{1}{N_s} \sum_{i=1}^{N_s} g_{\mathbf{c}}(\mathbf{k}_i), \qquad \mathcal{L}^{pop}(\mathbf{c}) = \int_{\mathcal{D}} g_{\mathbf{c}}(\mathbf{k})\, d\rho(\mathbf{k}).$$

By Proposition 4.1 and Assumption 4.1, we have

$$0 \;\leq\; g_{\mathbf{c}}(\mathbf{k}) \;\leq\; \big(\sigma^* M_\infty + C_F\big)^2 \;=:\; B, \tag{4.15}$$

for all $g_{\mathbf{c}} \in \mathcal{G}_n$, $\mathbf{k} \in \mathcal{D}$ and $\mathbf{c} \in \mathcal{N}_n$. Hence, $\mathcal{G}_n \subset [0, B]^{\mathcal{D}}$. We have derived the following uniform law:

Lemma 4.1 (Uniform convergence [55, Theorem 4.10], [52, Theorem 26.5])**.** *For any $\delta \in (0,1)$, with probability at least $1 - \delta$ over the draw of $\{\mathbf{k}_i\}_{i=1}^{N_s} \overset{i.i.d.}{\sim} \rho$,*

$$\sup_{\mathbf{c}\in\mathcal{N}_n} \Big|\, \mathcal{L}^{N_s}(\mathbf{c}) - \mathcal{L}^{\text{pop}}(\mathbf{c}) \,\Big| \;\leq\; 2\, R_{N_s}(\mathcal{G}_n) \;+\; B \sqrt{\frac{2\log(1/\delta)}{N_s}}. \tag{4.16}$$

We next control $R_{N_s}(\mathcal{G}_n)$ on the RHS of (4.16) in terms of the pseudo-dimension p_n.

Definition 4.3 (Covering number and ϵ-net [56])**.** *Let $(\mathcal{V}, \|\cdot\|)$ be a normed linear space and let $\mathcal{F} \subset \mathcal{V}$. A finite set $E \subset \mathcal{F}$ is called an ϵ-net of $\mathcal{F}$ with respect to $\|\cdot\|$ if for every $f \in \mathcal{F}$ there exists $e \in E$ such that $\|f - e\| \leq \epsilon$. The ϵ-covering number of $\mathcal{F}$ is*

$$\mathcal{N}(\mathcal{F}, \|\cdot\|, \epsilon) \;:=\; \min\{\, |E| : \; E \subset \mathcal{F} \text{ is an } \epsilon\text{-net of } \mathcal{F} \text{ w.r.t. } \|\cdot\| \,\},$$

with the convention $\mathcal{N}(\mathcal{F}, \|\cdot\|, \epsilon) = +\infty$ if no finite ϵ-net exists.

Lemma 4.2. *Fix $n \in \mathbb{N}$. Under Proposition 4.1 and Assumption 4.1, there exists a universal constant $C > 0$, which is independent of n, N_s, such that*

$$R_{N_s}(\mathcal{G}_n) \;\le\; C\,B\,\sqrt{\frac{p_n}{N_s}}. \tag{4.17}$$

In particular, for each fixed n, $R_{N_s}(\mathcal{G}_n) \to 0$ as $N_s \to \infty$.

Proof. For $\mathbf{c}_1, \mathbf{c}_2 \in \mathcal{N}_n$ and any $\mathbf{k} \in \mathcal{D}$, set $h_{\mathbf{c}}(\mathbf{k}) := \mathbf{A}_{\mathrm{rb}}(\mathbf{k})\mathbf{c}(\mathbf{k}) - \mathbf{F}_{\mathrm{rb}}(\mathbf{k})$ and note

$$\left|g_{\mathbf{c}_1}(\mathbf{k}) - g_{\mathbf{c}_2}(\mathbf{k})\right| = \left|\|h_{\mathbf{c}_1}(\mathbf{k})\|_2^2 - \|h_{\mathbf{c}_2}(\mathbf{k})\|_2^2\right| \le \left(\|h_{\mathbf{c}_1}(\mathbf{k})\|_2 + \|h_{\mathbf{c}_2}(\mathbf{k})\|_2\right)\|h_{\mathbf{c}_1}(\mathbf{k}) - h_{\mathbf{c}_2}(\mathbf{k})\|_2.$$

Since we have the uniform range bound (4.15), we can get the Lipschitz transfer

$$\|g_{\mathbf{c}_1} - g_{\mathbf{c}_2}\|_{L_2(P)} \;\le\; L\,\|\mathbf{c}_1 - \mathbf{c}_2\|_{L_2(P)}, \qquad L := 2(\sigma^* M_\infty + C_F)\,\sigma^*.$$

Hence, by the standard covering-number transfer,

$$\mathcal{N}(\mathcal{G}_n, \|\cdot\|_{L_2(P)}, \epsilon) \;\le\; \mathcal{N}(\mathcal{N}_n, \|\cdot\|_{L_2(P)}, \epsilon/L), \qquad \forall\, \epsilon > 0,\ \forall\, P. \tag{4.18}$$

For vector-valued networks with finite pseudo-dimension p_n, standard results in [50, Theorem 18.4] imply that for any probability measure P on $\mathcal{D}$ and any $\varepsilon \in (0, M_\infty]$,

$$\mathcal{N}(\mathcal{N}_n, \|\cdot\|_{L_2(P)}, \varepsilon) \;\le\; \Big(\frac{C_1\,M_\infty}{\varepsilon}\Big)^{C_2\,p_n}, \tag{4.19}$$

for universal constants $C_1, C_2 > 0$. Combining this with (4.18) and absorbing constants yields, for every $\epsilon \in (0, B]$,

$$\mathcal{N}(\mathcal{G}_n, \|\cdot\|_{L_2(P)}, \epsilon) \;\le\; \Big(\frac{C_3\,B}{\epsilon}\Big)^{C_2\,p_n}, \tag{4.20}$$

where $C_3 > 0$ is a universal constant independent of n and N_s.

By Dudley's entropy integral bound for the empirical Rademacher complexity [52, Theorem 27.4], for any $\alpha \in (0, B]$,

$$\widehat{R}_S(\mathcal{G}_n) \;\le\; \alpha \;+\; \frac{6}{\sqrt{N_s}}\int_\alpha^B \sqrt{\log \mathcal{N}(\mathcal{G}_n, \|\cdot\|_{L_2(P_{N_s})}, \epsilon)}\,d\epsilon.$$

Taking expectation in the sample S and using that $\sup_P$ dominates the empirical metric gives

$$R_{N_s}(\mathcal{G}_n) \;\le\; \alpha \;+\; \frac{6}{\sqrt{N_s}}\int_\alpha^B \sqrt{\sup_P \log \mathcal{N}(\mathcal{G}_n, \|\cdot\|_{L_2(P)}, \epsilon)}\;d\epsilon.$$

Inserting (4.20) and choosing $\alpha = B/\sqrt{N_s}$, we obtain

$$\begin{aligned} R_{N_s}(\mathcal{G}_n) \;&\le \alpha \;+\; \frac{6\sqrt{C_2\,p_n}}{\sqrt{N_s}}\int_\alpha^B \sqrt{\log\Big(\frac{C_3\,B}{\epsilon}\Big)}\,d\epsilon \\ &\le C\,B\,\sqrt{\frac{p_n}{N_s}}, \end{aligned} \tag{4.21}$$

for a universal constant $C > 0$. This proves (4.17). □

Combining Lemma 4.1 and Lemma 4.2, we obtain, for any fixed n and any $\delta \in (0, 1)$, with probability at least $1 - \delta$,

$$\sup_{\mathbf{c}\in\mathcal{N}_n}\left|\,\mathcal{L}^{N_s}(\mathbf{c}) - \mathcal{L}^{\mathrm{pop}}(\mathbf{c})\,\right| \;\le\; C\,B\,\sqrt{\frac{p_n}{N_s}} \;+\; B\,\sqrt{\frac{2\log(1/\delta)}{N_s}},$$

which shows uniform convergence of the empirical loss to the population loss at the rate $O\Big(\sqrt{p_n/N_s}\Big)$ for each fixed network size n.

Theorem 4.4. *Suppose Assumption 4.1 holds. Then for any fixed $n \in \mathbb{N}$, we have $\lim_{N_s\to\infty} R_{N_s}(\mathcal{G}_n) = 0$. Moreover, with probability 1 over i.i.d. samples $\{\mathbf{k}_j\}_{j=1}^{N_s}$,*

$$\lim_{N_s\to\infty} \left\|\widehat{\mathbf{c}}_n^{N_s} - \widehat{\mathbf{c}}_n\right\|_{L^2(\mathcal{D};\rho)} = 0, \qquad \lim_{N_s\to\infty} \left\|\widehat{w}_{n,N_s} - \widehat{w}_n\right\|_{L^2(\mathcal{D};L^2(\Omega))} = 0. \tag{4.22}$$

Proof. Since $\widehat{\mathbf{c}}_n^{N_s}$ minimizes $\mathcal{L}^{N_s}$ over $\mathcal{N}_n$,

$$\mathcal{L}^{N_s}(\widehat{\mathbf{c}}_n^{N_s}) \;\le\; \mathcal{L}^{N_s}(\widehat{\mathbf{c}}_n).$$

Hence

$$\mathcal{L}^{\mathrm{pop}}(\widehat{\mathbf{c}}_n^{N_s}) - \mathcal{L}^{\mathrm{pop}}(\widehat{\mathbf{c}}_n) \;\le\; (\mathcal{L}^{\mathrm{pop}} - \mathcal{L}^{N_s})(\widehat{\mathbf{c}}_n^{N_s}) \;-\; (\mathcal{L}^{\mathrm{pop}} - \mathcal{L}^{N_s})(\widehat{\mathbf{c}}_n).$$

Taking the supremum over $\mathcal{N}_n$ on the right-hand side and applying Lemma 4.1 yields, with probability at least $1-\delta$,

$$\mathcal{L}^{\mathrm{pop}}(\widehat{\mathbf{c}}_n^{N_s}) - \mathcal{L}^{\mathrm{pop}}(\widehat{\mathbf{c}}_n) \;\le\; 2\sup_{\mathbf{c}\in\mathcal{N}_n} \left|\mathcal{L}^{N_s}(\mathbf{c}) - \mathcal{L}^{\mathrm{pop}}(\mathbf{c})\right| \;\le\; 4\,R_{N_s}(\mathcal{G}_n) \;+\; 2B\,\sqrt{\tfrac{2\log(1/\delta)}{N_s}}, \tag{4.23}$$

Letting $N_s \to \infty$, Lemma 4.2 shows that the right-hand side of (4.23) converges to 0 almost surely.

Proposition 4.1 implies

$$\sigma_*^2\|\mathbf{c} - \widehat{\mathbf{c}}_n\|_{L^2(\mathcal{D};\rho)}^2 \;\le\; \mathcal{L}^{\mathrm{pop}}(\mathbf{c}) - \mathcal{L}^{\mathrm{pop}}(\widehat{\mathbf{c}}_n) \;\le\; (\sigma^*)^2\|\mathbf{c} - \widehat{\mathbf{c}}_n\|_{L^2(\mathcal{D};\rho)}^2, \qquad \forall\, \mathbf{c} \in \mathcal{N}_n. \tag{4.24}$$

Combining (4.23) and (4.24) with $\mathbf{c} = \widehat{\mathbf{c}}_n^{N_s}$ and $\delta = N_s^{-1/2}$ gives, with probability one,

$$\lim_{N_s\to\infty} \left\|\widehat{\mathbf{c}}_n^{N_s} - \widehat{\mathbf{c}}_n\right\|_{L^2(\mathcal{D};\rho)} = 0. \tag{4.25}$$

Since $\{\psi_i\}_{i=1}^N \subset L^2(\Omega)$ are fixed, the lifting bound (4.3) gives

$$\left\|\widehat{w}_{n,N_s} - \widehat{u}_n\right\|_{L^2(\mathcal{D};L^2(\Omega))} \;\le\; C_\psi \left\|\widehat{\mathbf{c}}_n^{N_s} - \widehat{\mathbf{c}}_n\right\|_{L^2(\mathcal{D};\rho)}, \tag{4.26}$$

which yields the second limit in (4.22). □

Based on Theorems 4.3 and 4.4 and the uniform equivalence of (3.20) and (4.5), now we can address the convergence of the predicted solution by RB–DeepONet to the reduced finite element solution.

Theorem 4.5 (Convergence of RB–DeepONet to the RB solution)**.** *Suppose Assumption 4.1 holds. For the fixed RB basis $\{\psi_i\}_{i=1}^N \subset L^2(\Omega)$, we have with probability 1 over i.i.d. samples $\{\mathbf{k}_j\}_{j=1}^{N_s}$ that*

$$\lim_{n\to\infty} \lim_{N_s\to\infty} \left\|w_\theta - w_{\mathrm{rb}}\right\|_{L^2(\mathcal{D};L^2(\Omega))} \;=\; 0. \tag{4.27}$$

Using (4.27), we are able to show that our algorithm can be sufficiently close to the approximate solution computed by the proposed scheme, as the index n for the neural network architecture and the number of input samples N_k go to infinity.

5. Numerical experiments

This section evaluates RB–DeepONet on representative parametric PDE benchmarks and compares it with POD–DeepONet [33] and FEONet [34] in terms of accuracy, efficiency, and scalability. For a fair comparison, we fix the number of trunk basis functions N in both RB–DeepONet and POD–DeepONet. Specifically, we first construct the POD trunk used in POD–DeepONet with tolerance $\epsilon_{\mathrm{POD}} = 10^{-7}$. This choice sets the reduced dimension N. A Greedy RB trunk of the same dimension is then built from the same snapshot set and used in RB–DeepONet, so that RB–DeepONet and POD–DeepONet are compared at equal trunk size. We follow the supervised training strategy of [33] and train POD–DeepONet by minimizing the empirical discrepancy between the network prediction and the reference FEM solution, as in (3.32). In contrast, RB–DeepONet and FEONet are trained in an unsupervised fashion using the residual-based losses (3.20) and (3.33), respectively.

5.1. Example 1: Fully-parameterized steady heat conduction

First, we consider a fully parameterized steady heat conduction model with piecewise-constant conductivity and a prescribed heat flux at the bottom. The domain is the square $\Omega = (-0.5, 0.5)^2$ with boundary split into three parts as in Fig. 2: the bottom $\Gamma_{\text{base}} = (-0.5, 0.5)\times\{-0.5\}$, the top $\Gamma_{\text{top}} = (-0.5, 0.5)\times\{0.5\}$ and the sides $\Gamma_{\text{side}} = \{\pm 0.5\}\times(-0.5, 0.5)$. An inclusion $\Omega_0 = \{\boldsymbol{x} : \|\boldsymbol{x}\|_2 \le r_0\}$ of radius $r_0 = 0.2$ is embedded and we denote $\Omega_1 := \Omega\backslash\Omega_0$. Consider the thermal conductivity κ to be piecewise constant,

$$\kappa(\mathbf{x}; k_1) = \begin{cases} k_1, & \mathbf{x} \in \Omega_0, \\ 1, & \mathbf{x} \in \Omega_1, \end{cases}$$

so that $k_1 \ge 0$ controls how conductive the inclusion is relative to the background. A constant heat flux k_2 is prescribed on the bottom boundary. We collect the two parameters in

$$\mathbf{k} := (k_1, k_2) \in \mathcal{D} := [0.1, 10] \times [-1, 1].$$

For each $\boldsymbol{k}$, the temperature $u(\cdot; \boldsymbol{k})$ solves

$$\begin{cases} \nabla\cdot(\kappa(\cdot; k_1)\nabla u(\cdot; \boldsymbol{k})) = 0 & \text{in } \Omega, \\ u(\cdot; \boldsymbol{k}) = 0 & \text{on } \Gamma_{\text{top}}, \\ \kappa(\cdot; k_1)\nabla u(\cdot; \boldsymbol{k})\cdot\boldsymbol{n} = 0 & \text{on } \Gamma_{\text{side}}, \\ \kappa(\cdot; k_1)\nabla u(\cdot; \boldsymbol{k})\cdot\boldsymbol{n} = k_2 & \text{on } \Gamma_{\text{base}}. \end{cases} \tag{5.1}$$

Here, $\boldsymbol{n}$ is outward pointing unit normal on $\partial\Omega$. The corresponding weak form reads: for each $\mathbf{k} \in \mathcal{D}$, find $u(\mathbf{k}) := u(\mathbf{x}; \mathbf{k}) \in V := \{v \in H^1(\Omega) : v\big|_{\Gamma_{\text{top}}} = 0\}$ such that

$$a(u(\mathbf{k}), v; \mathbf{k}) = \ell(v; \mathbf{k}), \quad \forall v \in V, \tag{5.2}$$

where

$$a(u, v; \mathbf{k}) := \int_\Omega \kappa(\mathbf{x}; k_1)\nabla u \cdot \nabla v, \qquad \ell(v; \mathbf{k}) := \quad k_2 \int_{\Gamma_{\text{base}}} v. \tag{5.3}$$

Since $\kappa(\mathbf{x}; k_1) \ge 0.1$ for all admissible k_1, the bilinear form is coercive and the Lax-Milgram theorem implies existence and uniqueness of $u(\mathbf{k}) \in V$ for any $\mathbf{k} \in \mathcal{D}$.

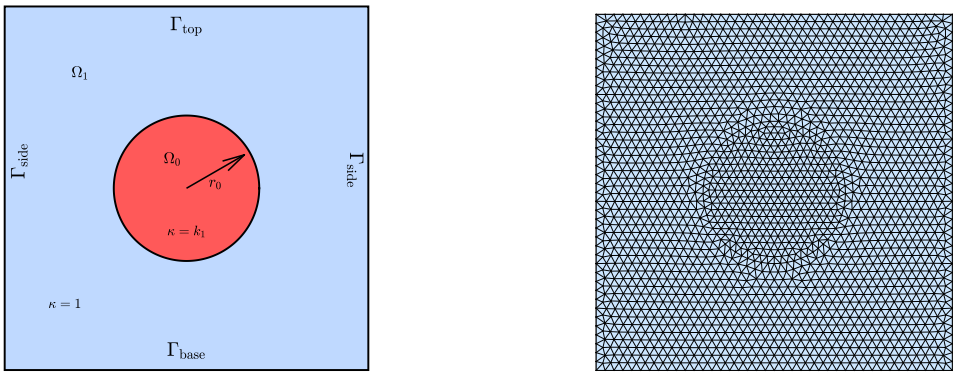


Fig. 2. Geometrical set-up (left) of the heat conductivity problem and mesh on Ω (right).

We discretize (5.2) with continuous, piecewise linear finite elements on the conforming triangular mesh in 2(b), which has 4052 elements and $N_h = 2107$ vertices. Eliminating the homogeneous Dirichlet dofs on Γ_{top} gives

$$\mathbf{A}_{II}(\boldsymbol{k})\,\mathbf{u}_I(\boldsymbol{k}) = \mathbf{F}_I(\boldsymbol{k}), \qquad \mathbf{A}_{II}(\boldsymbol{k}) = \mathbf{A}_1 + k_1\mathbf{A}_0, \quad \mathbf{F}_I(\boldsymbol{k}) = k_2\mathbf{F}.$$

Here $\mathbf{A}_0, \mathbf{A}_1$ are the stiffness contributions from Ω_0, Ω_1 and $\mathbf{F}$ the Neumann load on Γ_{base}. They are defined on the free nodes with $N_0 = 2066$ dofs, i.e., $\mathbf{A}_0, \mathbf{A}_1 \in \mathbb{R}^{N_0 \times N_0}$, $\mathbf{F} \in \mathbb{R}^{N_0}$. The FE solution $\mathbf{u}_I(\boldsymbol{k}) \in \mathbb{R}^{N_0}$ serves only as a reference for RB basis construction and for evaluation. We equip $\mathbb{R}^{N_0}$ with the energy inner product at $\boldsymbol{k}_\star = (1, 1)$, i.e., $(\boldsymbol{u}_I, \boldsymbol{v}_I)_V = \boldsymbol{u}_I^\top \mathbf{A}_{II}(\boldsymbol{k}_\star)\boldsymbol{v}_I$.

From $N_k = 2100$ i.i.d. samples in $\mathcal{D}$, we first construct a POD basis with tolerance $\epsilon_{\text{POD}} = 10^{-7}$, which yields a reduced dimension $N = 3$ and a trunk $\Psi^{\text{POD}} = [\psi_1^{\text{POD}}, \psi_2^{\text{POD}}, \psi_3^{\text{POD}}] \in \mathbb{R}^{N_0 \times N}$ for POD–DeepONet. Using the same parameter samples, we then run Greedy algorithm to generate an RB basis of the same size, $\Psi^{\text{G}} = [\psi_1^{\text{G}}, \psi_2^{\text{G}}, \psi_3^{\text{G}}] \in \mathbb{R}^{N_0 \times N}$, which is fixed as the trunk in RB–DeepONet. Thus, POD– and RB–DeepONet are compared at equal reduced dimension N.

Given the Greedy trunk Ψ^{G}, the intrusive RB–Galerkin system (cf. (3.14)) reads: for each $\mathbf{k} \in \mathcal{D}$, find $\mathbf{c}_{\text{rb}}(\mathbf{k}) \in \mathbb{R}^N$ such that

$$\mathbf{A}_{\text{rb}}(\mathbf{k})\, \mathbf{c}_{\text{rb}}(\mathbf{k}) = \mathbf{F}_{\text{rb}}(\mathbf{k}),$$

with $\mathbf{A}_{\text{rb}}(\mathbf{k})$ and $\mathbf{F}_{\text{rb}}(\mathbf{k})$ assembled from Ψ^{G} as in (3.15). RB–DeepONet learns the coefficient map $\mathbf{k} \mapsto \mathbf{c}_{\text{rb}}(\mathbf{k})$ and predicts

$$u_\theta(\mathbf{k}) := \Psi^{\text{G}}\, \mathbf{c}_\theta(\mathbf{k}).$$

POD–DeepONet uses the same branch architecture but replaces Ψ^{G} by the POD trunk Ψ^{POD}. FEONet, by contrast, keeps the full FE basis $\{\phi_j\}_{j=1}^{N_0}$ (here $N_0 = 2066$ piecewise linear nodal basis functions) as trunk, and the network directly outputs $\mathbf{u}_\theta(\mathbf{k}) \in \mathbb{R}^{N_0}$, the nodal vector of the predicted FE solution.

Here are the details of our architecture and training setup: In all three methods, we employ a fully connected multilayer perceptron (MLP) as the branch network, consisting of four hidden layers, each with a width of 256 and GELU activation functions. The input vector $\mathbf{k} \in \mathbb{R}^2$ is standardized using the mean and variance computed from the training parameter samples, and the same transformation is applied during inference. Network weights are initialized using the Xavier scheme to ensure stable gradients. Training is performed for up to 2000 epochs with mini-batches of size 64 using the AdamW optimizer (initial learning rate 5×10^{-4}, weight decay 10^{-6}). A ReduceLROnPlateau scheduler monitors the validation loss and halves the learning rate after 20 consecutive plateaus. The training set comprises 2000 randomly sampled parameters from $\mathcal{D}$, while validation uses 200 held-out parameters. Early stopping with a patience of 200 epochs is applied to prevent overfitting.

Figure 4 shows the training/validation losses. All three networks reach a small residual level. Figure 3 displays a representative test parameter $\mathbf{k} = (10.0, 0.5)$. The FEM reference (top left) shows a smooth vertical gradient with a localized "thermal bubble" inside the high-conductivity inclusion. FEONet, POD–DeepONet, and RB–DeepONet all reproduce this structure and are visually indistinguishable at the plotting scale.

For a test parameter $\mathbf{k}$, let $\mathbf{u}_I(\mathbf{k})$ be the FE solution vector restricted to the free nodes, and $\mathbf{e}_{\text{nn}}(\mathbf{k}) := \mathbf{u}_\theta(\mathbf{k}) - \mathbf{u}_I(\mathbf{k})$. We report three relative error measures on the free block:

$$\begin{aligned}
\text{rel-}L^2(\mathbf{k}) &:= \frac{\left(\mathbf{e}_{\text{nn}}(\mathbf{k})^\top \mathbf{M}_{II}(\mathbf{k}_\star)\mathbf{e}_{\text{nn}}(\mathbf{k})\right)^{1/2}}{\left(\mathbf{u}_I(\mathbf{k})^\top \mathbf{M}_{II}(\mathbf{k}_\star)\mathbf{u}_I(\mathbf{k})\right)^{1/2}}, \\
\text{rel-energy}(\mathbf{k}) &:= \frac{\left(\mathbf{e}_{\text{nn}}(\mathbf{k})^\top \mathbf{A}_{II}(\mathbf{k}_\star)\mathbf{e}_{\text{nn}}(\mathbf{k})\right)^{1/2}}{\left(\mathbf{u}_I(\mathbf{k})^\top \mathbf{A}_{II}(\mathbf{k}_\star)\mathbf{u}_I(\mathbf{k})\right)^{1/2}}, \\
\text{rel-residual}(\mathbf{k}) &:= \frac{\|\mathbf{A}_{\text{rb}}(\mathbf{k}_\star)^{-1/2}\mathbf{r}(\mathbf{k})\|_2}{\|\mathbf{A}_{\text{rb}}(\mathbf{k}_\star)^{-1/2}\mathbf{F}_{\text{rb}}(\mathbf{k})\|_2}.
\end{aligned}$$

Here, $\mathbf{M}_{II}$ is the mass matrix on V_0. Table 2 reports the mean and 95th percentile of these errors over 1000 i.i.d. test parameters. With only $N = 3$ modes, the RB–Galerkin reference already achieves mean rel-L^2 of order 10^{-7} and mean rel-energy of order 10^{-6}, which confirms that the Greedy basis Ψ^{G} is globally accurate. Among the learned surrogates, POD–DeepONet and RB–DeepONet deliver very similar accuracy. Both attain mean rel-L^2 below 5×10^{-3} and 95th percentiles below 6.1×10^{-3}, so the reconstructed fields differ from the FE solutions by well under 1% on average. POD–DeepONet has a slightly smaller mean errors than RB–DeepONet, whereas RB–DeepONet exhibits smaller 95th-percentile errors, indicating a more uniform control of the RB residual across the parameter space. FEONet attains the smallest rel-L^2 among the three networks, but its rel-energy and rel-residual are an order of magnitude larger, with 95th-percentile values around 3×10^{-2}. This reflects the fact that FEONet operates in the full FE space

and optimizes an unpreconditioned variational residual, which is sensitive to the conditioning of the stiffness matrix. Moreover, FEONet uses an output dimension equal to the number of free FE degrees of freedom ($N_0 \approx 2000$), whereas POD–DeepONet and RB–DeepONet rely on only $N = 3$ trunk modes. Thus, RB–DeepONet and POD–DeepONet achieve FEONet-level accuracy in rel-L^2 while using a trunk that is almost three orders of magnitude smaller and, in the case of RB–DeepONet, preserving RB-type control in energy and residual norms.

Overall, this example shows that RB–DeepONet and POD–DeepONet can match the accuracy of FEONet up to a small factor while using an extremely low-dimensional trunk, and RB–DeepONet in particular inherits the RB–Galerkin error level and lightweight inference of a reduced model. This highlights the advantage of operating entirely in a certified RB space instead of predicting a full-order FE vector.

Trunk	**rel-L^2** (mean / $p95$)	**rel-energy** (mean / $p95$)	**rel-residual** (mean / $p95$)
RB–DeepONet	$\mathbf{4.99\times10^{-3}}$ / $\mathbf{4.66\times10^{-3}}$	$\mathbf{4.99\times10^{-3}}$ / $\mathbf{4.56\times10^{-3}}$	$\mathbf{5.14\times10^{-3}}$ / $\mathbf{4.90\times10^{-3}}$
POD–DeepONet	3.98×10^{-3} / 6.05×10^{-3}	4.08×10^{-3} / 6.30×10^{-3}	4.97×10^{-3} / 7.50×10^{-3}
FEONet	2.67×10^{-3} / 5.17×10^{-3}	1.26×10^{-2} / 2.89×10^{-2}	1.44×10^{-2} / 3.12×10^{-2}
RB–Galerkin	2.11×10^{-7} / 4.23×10^{-7}	2.77×10^{-6} / 5.70×10^{-6}	7.57×10^{-16} / 1.54×10^{-15}

Table 2. Example 5.1: Test errors over 1000 random parameters for RB–DeepONet, POD–DeepONet, FEONet, and RB–Galerkin.

5.2. Example 2: Parametric diffusion–reaction with independently varying data.

Second, we consider the family of second–order elliptic problems. The strong form is

$$\begin{cases} \mathcal{L}(\mathbf{k})\, u(\mathbf{x};\mathbf{k}) = f(\mathbf{x}), & \mathbf{x} \in \Omega, \\ u(\mathbf{x};\mathbf{k}) = g_D(\mathbf{x}), & \mathbf{x} \in \Gamma_D, \\ \kappa_0\, \nabla u(\mathbf{x};\mathbf{k}) \cdot \mathbf{n} = h_N(\mathbf{x}), & \mathbf{x} \in \Gamma_N, \\ \kappa_0\, \nabla u(\mathbf{x};\mathbf{k}) \cdot \mathbf{n} + \beta_0\, u(\mathbf{x};\mathbf{k}) = r_R(\mathbf{x}), & \mathbf{x} \in \Gamma_R, \end{cases} \tag{5.4}$$

Here, $\Omega := (0,1)^2$ with the boundary partition

$$\Gamma_D := \{x = 0\} \cup \{y = 1\}, \qquad \Gamma_N := \{y = 0\}, \qquad \Gamma_R := \{x = 1\},$$

and the operator is defined as

$$\mathcal{L}(\mathbf{k}) := \kappa_0(-\Delta) + \alpha_0 I, \qquad \kappa_0 > 0,\ \alpha_0 \geq 0. \tag{5.5}$$

Here, $\mathbf{k} := [\kappa_0, \alpha_0, \beta_0]^\top$ is the parameter.

To obtain consistent boundary and source pairs, we prescribe a smooth solution family

$$\begin{aligned} u_\star(x, y; \boldsymbol{\xi}) = {} & a_1 \sin(\pi x)\sin(\pi y) + a_2 \sin(2\pi x)\sin(\pi y) + a_3 \exp\Big(-\tfrac{(x-x_c)^2+(y-y_c)^2}{2\sigma^2}\Big) \\ & + a_4 \cos(\pi x)\sinh(y - \tfrac{1}{2}), \end{aligned} \tag{5.6}$$

and introduce an auxiliary vector $\boldsymbol{\xi} := [a_1, a_2, a_3, a_4, x_c, y_c, \sigma]^\top$ to modulate the boundary traces and the distributed source independently of $\mathbf{k}$. With $\mathbf{k}$ governing the operator coefficients, this choice isolates exogenous data variability and directly tests the boundary- and source-mode mechanisms of RB–DeepONet. We sample the parameters independently as

$$a_1, a_2, a_4 \sim \mathcal{U}[-1, 1], \qquad a_3 \sim \mathcal{U}[0, 1], \qquad (x_c, y_c) \sim \mathcal{U}([0.2, 0.8]^2), \qquad \sigma \sim \mathcal{U}[0.05, 0.2],$$

$$\kappa_0 \sim \mathcal{U}[0.5, 2], \qquad \alpha_0 \sim \mathcal{U}[0, 2], \qquad \beta_0 \sim \mathcal{U}[0, 10].$$

The data (f, g_D, h_N, r_R) are then defined by

$$\begin{aligned} f(\cdot) :=& \mathcal{L}(\mathbf{k})\, u_\star(\cdot;\boldsymbol{\xi}), \quad g_D(\cdot) := u_\star(\cdot;\boldsymbol{\xi})|_{\Gamma_D}, \\ h_N(\cdot) :=& \kappa_0\, \nabla u_\star(\cdot;\boldsymbol{\xi}) \cdot \mathbf{n}\big|_{\Gamma_N}, \quad r_R(\cdot) := \kappa_0\, \nabla u_\star(\cdot;\boldsymbol{\xi}) \cdot \mathbf{n} + \beta_0\, u_\star(\cdot;\boldsymbol{\xi})\big|_{\Gamma_R}. \end{aligned} \tag{5.7}$$

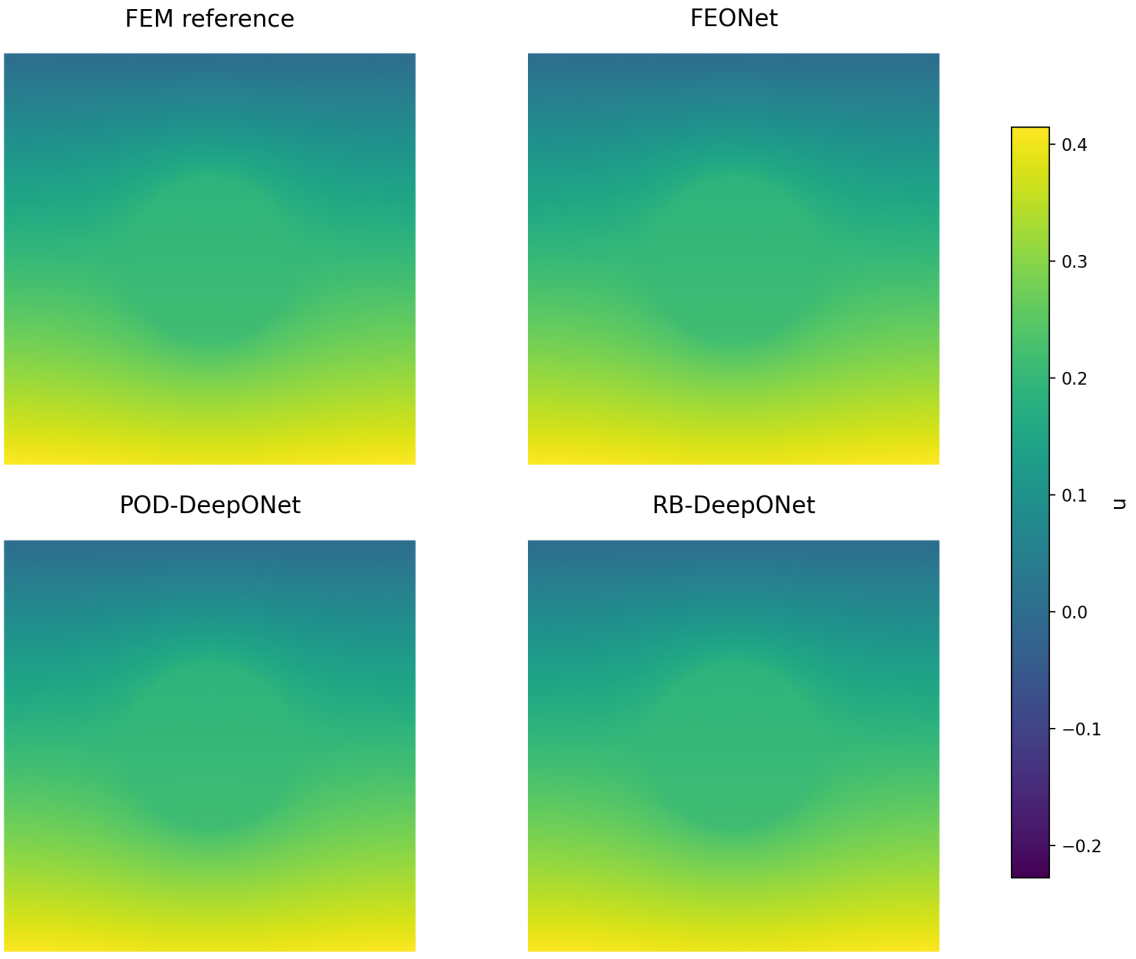
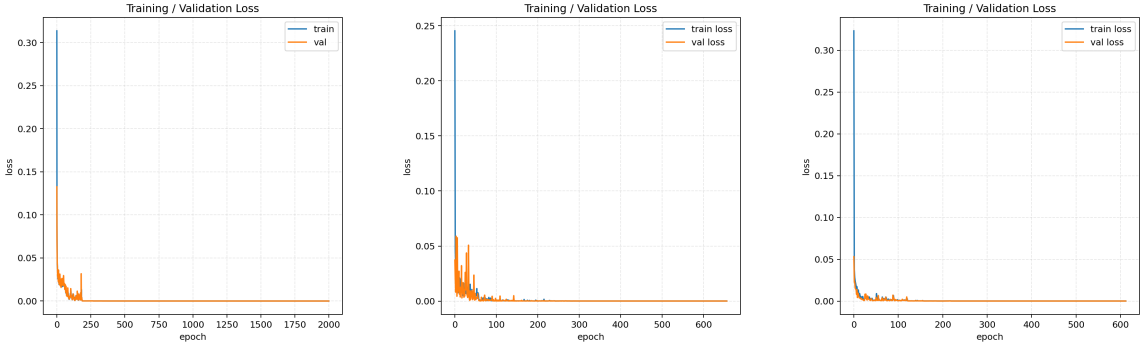


Fig. 3. Example 5.1: Representative-parameter comparison at $\mathbf{k} = (k_1, k_2) = (10, 0.5)$ of (top–left) FEM reference, (top–right) FEONet, (bottom–left) POD–DeepONet, and (bottom–right) RB–DeepONet.



(a) FEONet (b) POD–DeepONet (c) RB–DeepONet

Fig. 4. Example 5.1: Training/validation loss.

We follow Section 3.4 to construct boundary and source modes, and precompute the RB trunk and reduced operators. The domain $\Omega = (0,1)^2$ is discretized by a 64×64 structured triangulation, giving $N_h = 64^2 = 4096$ nodal unknowns. The Dirichlet boundary is $\Gamma_D = \{x = 0\} \cup \{y = 1\}$, containing $|\mathcal{B}_D| = 127$ nodes, so the number of free degrees of freedom is $N_0 = 4096 - 127 = 3969$.

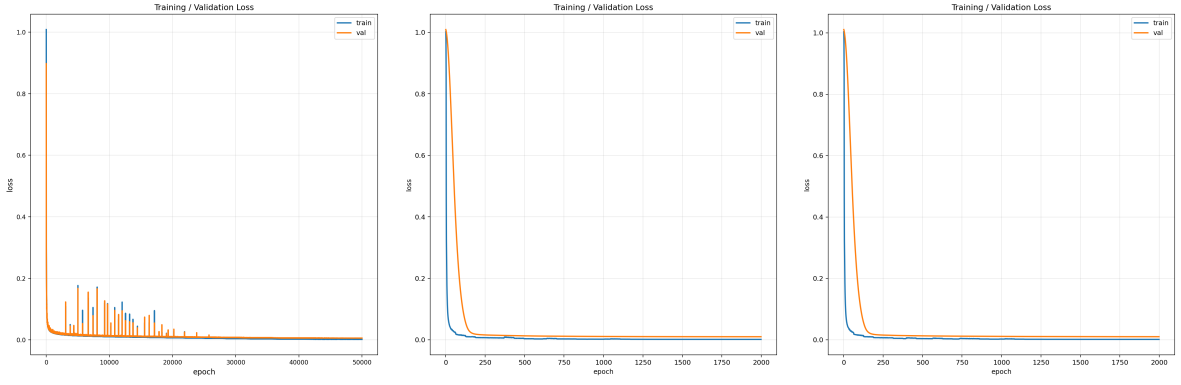
For the offline stage, we draw $N_k = 10{,}000$ parameter–data samples and split them 80%/20% into training and validation sets for learning the branch networks. We fix the reference parameter for the energy norm and lifting as $\mathbf{k}_\star = [1.2, 0.6, 1.0]^\top$. A uniform tolerance 10^{-7} is used for the boundary, source, and trunk constructions, which yields $(N, r_f, r_g) = (209, 128, 16)$. For RB–DeepONet and POD–DeepONet, we target these same ranks to ensure a fair comparison.

Strictly speaking, both POD–DeepONet and FEONet are designed to take as input the full-order discretizations of the data (f, g_D, h_N, r_R) together with the physical parameters, which in this example would lead to prohibitively high-dimensional inputs. Therefore, we instead work with the low-dimensional augmented feature vector $\mathbf{k}_{\text{aug}}$ and the approximate right-hand side obtained by projecting the data onto the boundary and source modes, so that all three networks use $\mathbf{k}_{\text{aug}}$ as input while the variational residuals are evaluated with $\mathbf{F}_{\text{rb}}(\mathbf{k}_{\text{aug}})$ in (3.28) and the corresponding full-order matrices. Thus, the branch input is the feature vector in (3.31) of dimension $3 + r_f + r_g = 147$, and the branch output has dimension $N = 209, 3969$ for RB– and POD–DeepONets and FEONet, respectively.

We train FEONet for 50,000 epochs, whereas RB–DeepONet and POD–DeepONet are trained for 2000 epochs. Notably, RB– and POD–DeepONet output only $N = 209$ coefficients, versus $N_0 = 3969$ for FEONet, about 19 times smaller, while also requiring 25 times fewer training epochs, highlighting their lightweight design and faster, more stable training. All remaining settings (branch architecture, optimizer, learning-rate schedule, residual loss, and batch protocol) follow Example 5.1.

Figure 5 shows the training/validation losses. All three networks eventually converge, with POD–DeepONet and RB–DeepONet reaching a low residual level after roughly 10^3 epochs, whereas FEONet requires substantially more epochs and exhibits larger fluctuations. Figure 6 displays a representative parameter: the ground truth solution, FEONet, POD–DeepONet, and RB–DeepONet are visually indistinguishable, indicating that all surrogates reproduce the mixed boundary conditions and spatially varying load at the plotting scale. Table 3 reports mean and 95th-percentile errors over 1000 test parameters. The RB–Galerkin reference again attains small errors (mean rel-L^2 $\approx 1.4 \times 10^{-3}$), confirming that the RB trunk and the boundary/source modes are accurate for this Case II setting. Among the learned models, POD–DeepONet and RB–DeepONet both achieve mean rel-L^2 of order 10^{-2} with 95th percentiles below $6{\times}10^{-2}$. POD–DeepONet is slightly more accurate in rel-L^2 and rel-energy, whereas RB–DeepONet remains within a modest factor while relying only on residual information and operating entirely in the reduced space. FEONet attains the smallest rel-L^2, but at the cost of predicting $N_0 = 3969$ coefficients instead of $N = 209$ reduced coordinates.

Overall, this example demonstrates that RB–DeepONet can handle exogenous boundary and source data that vary independently of the physical parameters and, with a compact RB trunk, delivers relative L^2 errors of about 1% while avoiding online PDE solves and high-dimensional FE outputs.



(a) FEONet (b) POD–DeepONet (c) RB–DeepONet

Fig. 5. Example 5.2: Training/validation loss for FEONet and POD–, RB–DeepONet with $(N, r_f, r_g) = (209, 128, 16)$.

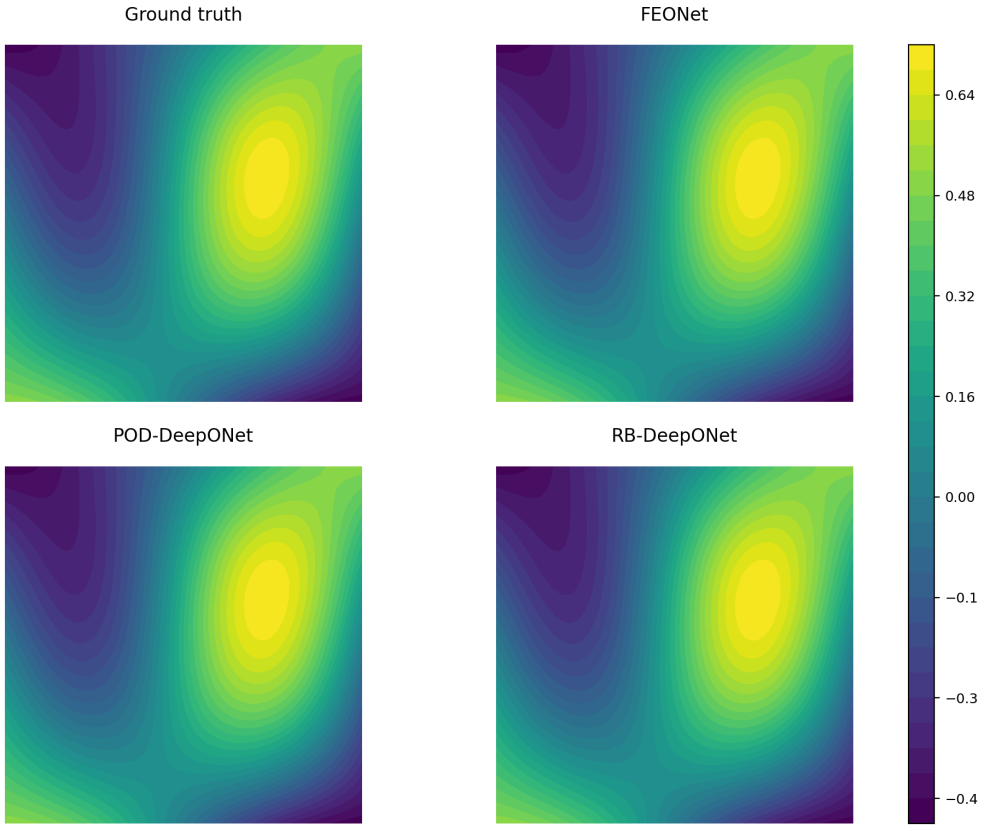


Fig. 6. Example 5.2: Representative-parameter comparison of (top–left) ground truth, (top–right) FEONet, (bottom–left) POD–DeepONet, and (bottom–right) RB–DeepONet.

Trunk	**rel-L^2 (mean / $p95$)**	**rel-energy (mean / $p95$)**	**rel-residual (mean / $p95$)**
RB–DeepONet	$\mathbf{1.17 \times 10^{-2}}$ / $\mathbf{4.33 \times 10^{-2}}$	$\mathbf{1.53 \times 10^{-2}}$ / $\mathbf{5.88 \times 10^{-2}}$	$\mathbf{1.38 \times 10^{-2}}$ / $\mathbf{5.08 \times 10^{-2}}$
POD–DeepONet	9.47×10^{-3} / 2.37×10^{-2}	1.06×10^{-2} / 4.18×10^{-2}	9.68×10^{-3} / 4.33×10^{-2}
FEONet	2.18×10^{-3} / 4.68×10^{-3}	4.88×10^{-3} / 1.39×10^{-2}	7.15×10^{-16} / 7.94×10^{-16}
RB–Galerkin	1.41×10^{-3} / 4.50×10^{-3}	5.99×10^{-3} / 2.33×10^{-2}	4.59×10^{-16} / 8.15×10^{-16}

Table 3. Example 5.2: Test errors (mean / $p95$) over 1000 random parameters.

5.3. Example 3: Non-affine heat conduction

In this example, we demonstrate that our framework can extend seamlessly to non-affine operators via empirical interpolation (EIM). Once the RB trunk is sufficiently expressive, the RB–DeepONet achieves accuracy comparable to the RB–Galerkin solution and the full FEM reference. We report results with the same training budget and evaluation protocol as in Example 5.1.

The geometry and boundary conditions follow Example 5.1. In addition to the parameters k_1, k_2, we parameterize the inclusion radius by k_3 and denote the disk by $\Omega_0(k_3)$. The thermal conductivity is

$$\kappa(\mathbf{x}; k_1, k_3) = \begin{cases} k_1, & \mathbf{x} \in \Omega_0(k_3), \\ 1, & \mathbf{x} \in \Omega_1 := \Omega \setminus \Omega_0(k_3). \end{cases}$$

We write $\mathbf{k} = [k_1, k_2, k_3]^\top$ and aim to learn the mapping $\mathbf{k} \mapsto u(\mathbf{x}; \mathbf{k})$, where $u(\cdot; \mathbf{k})$ solves (5.1) with $\kappa(\cdot; k_1)$ replaced by $\kappa(\cdot; k_1, k_3)$. The bilinear form reads

$$a(u, v; \mathbf{k}) = \int_\Omega \kappa(\mathbf{x}; k_1, k_3)\, \nabla u \cdot \nabla v \, \mathrm{d}\mathbf{x},$$

so the coefficient depends non-affinely on $\mathbf{k}$. We recover an approximate affine separation by EIM as in [2].

Let $\widehat{\Omega} := \widehat{\Omega}_1 \cup \widehat{\Omega}_0$ be a fixed reference domain with reference inclusion radius $r_0 = 0.2$ (as in Example 5.1). We introduce the continuous radial map

$$\mathbf{T}_r(\widehat{\mathbf{x}}) := \varphi_r(|\widehat{\mathbf{x}}|)\widehat{\mathbf{x}}, \qquad \widehat{\mathbf{x}} \in \widehat{\Omega},$$

with

$$\varphi_r(\rho) = \begin{cases} 1, & 0 \le \rho < r_-, \\ \dfrac{r_0(r_0 - \rho) + r(\rho - r_-)}{r_0(r_0 - r_-)}, & r_- \le \rho < r_0, \\ \dfrac{r_0(r_0 - \rho) + r(\rho - r_+)}{r_0(r_0 - r_+)}, & r_0 \le \rho < r_+, \\ 1, & r_+ \le \rho, \end{cases}$$

for some $0 < r_- \le r_{\min} < r_{\max} < r_+ < 1$ and $r \in [r_{\min}, r_{\max}]$. Here, $\rho := |\widehat{\mathbf{x}}|$. Pushing the PDE to $\widehat{\Omega}$ yields: find $\widehat{u}(\mathbf{k}) \in V$ such that

$$\widehat{a}(\widehat{u}, \widehat{v}; \mathbf{k}) = \widehat{\ell}(\widehat{v}; \mathbf{k}) \quad \forall \widehat{v} \in V,$$

where

$$\widehat{a}(\widehat{u}, \widehat{v}; \mathbf{k}) := \int_{\widehat{\Omega}_1} \nabla\widehat{u} \cdot (\mathbf{G}(\widehat{\mathbf{x}}; k_3)\nabla\widehat{v})\, \mathrm{d}\widehat{\mathbf{x}} \; + \; k_1 \int_{\widehat{\Omega}_0} \nabla\widehat{u} \cdot (\mathbf{G}(\widehat{\mathbf{x}}; k_3)\nabla\widehat{v})\, \mathrm{d}\widehat{\mathbf{x}},$$
$$\widehat{\ell}(\widehat{v}; \mathbf{k}) := k_2 \int_{\widehat{\Gamma}_{\text{base}}} \widehat{v}\, \mathrm{d}\widehat{s},$$

and

$$\mathbf{G}(\widehat{\mathbf{x}}; k_3) := \left|\det \widehat{\mathbf{J}}(\widehat{\mathbf{x}}; k_3)\right| \widehat{\mathbf{J}}(\widehat{\mathbf{x}}; k_3)^{-\top} \widehat{\mathbf{J}}(\widehat{\mathbf{x}}; k_3)^{-1}, \qquad (\widehat{\mathbf{J}}(\widehat{\mathbf{x}}; k_3))_{ij} := \frac{\partial(\mathbf{T}_{k_3}(\widehat{\mathbf{x}}))_j}{\partial \widehat{\mathbf{x}}_i}, \quad i, j = 1, 2.$$

We apply the empirical interpolation algorithm for vector fields [2, Chap. 5] to $\mathbf{G}(\cdot; k_3)$ and obtain

$$\mathbf{G}(\widehat{\mathbf{x}}; k_3) \approx \sum_{q=1}^{Q} \alpha_q(k_3)\, \mathbf{H}_q(\widehat{\mathbf{x}}), \tag{5.8}$$

with parameter-independent tensors $\{\mathbf{H}_q\}_{q=1}^{Q}$ and online coefficients $\alpha_q(k_3)$ determined from EIM pivot values. The approximation exhibits an exponential decay as Q increases (see [2]).

Let $\Psi = [\psi_1, \ldots, \psi_N]$ be the RB space obtained by Greedy selection, represented in the FE basis. Using (5.8), the reduced operators take the form

$$\mathbf{A}_{\mathrm{rb}}(\mathbf{k}) = \sum_{q=1}^{Q} \alpha_q(k_3)\, \mathbf{H}_{\mathrm{rb},q}, \qquad \mathbf{F}_{\mathrm{rb}}(\mathbf{k}) = \Psi^\top \mathbf{F}, \qquad \mathbf{H}_{\mathrm{rb},q} := \Psi^\top \mathbf{H}_q\, \Psi. \tag{5.9}$$

In the offline stage, we assemble the Q reduced blocks $\{\mathbf{H}_{\mathrm{rb},q}\}_{q=1}^{Q}$. Then, in the online stage, we no longer touch any full-order matrices. RB–DeepONet fixes Ψ as the trunk and trains only the branch $\mathbf{c}_\theta(\mathbf{k}) \in \mathbb{R}^N$ with the residual loss (3.20) that vanishes exactly at the RB–Galerkin coefficients. The prediction is $u_\theta(\mathbf{k}) = \Psi\, \mathbf{c}_\theta(\mathbf{k})$.

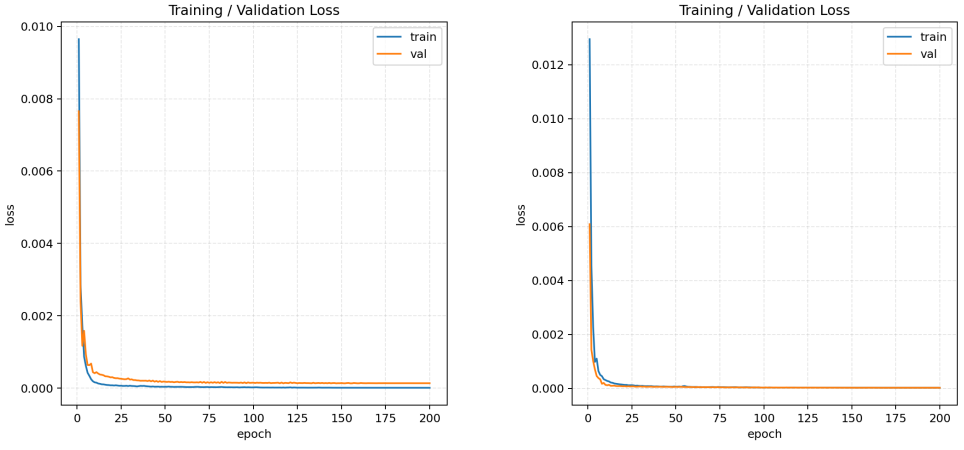
We consider

$$k_1 \in [0.1, 10], \qquad k_2 \in [-1, 1], \qquad k_3 \in [0.05, 0.45].$$

From 4000 random samples we build POD trunks with tolerance $\epsilon_{\mathrm{POD}} = 10^{-7}$, resulting in $N = 5$. For comparison, we also construct the Greedy RB trunk with $N = 5$. For the geometry surrogate, we take $Q = 15$, which yields an average relative L^2 error of about 10^{-4} against direct FEM assembled without the affine approximation as shown in Table 4. All the other network settings are the same as Example 5.1.

Figure 7 shows the training and validation losses. Both POD–DeepONet and RB–DeepONet converge rapidly to a small residual level with $N = 5$ trunk modes. Figure 8 fixes $\mathbf{k} = (6.68, 0.94, 0.2)$ and compares the FEM solution, the RB–Galerkin reference, POD–DeepONet, and RB–DeepONet. All RB-based surrogates are visually indistinguishable from the FEM field. Table 4 reports the mean and 95th-percentile errors over 1000 test parameters. The RB–Galerkin solution is essentially exact (mean rel-$L^2 \approx 1.6\times10^{-4}$), confirming that the RB space obtained from the EIM-assembled operators (5.9) is highly accurate even for this non-affine problem. Among the learned models, POD–DeepONet attains mean rel-L^2 around 9×10^{-3} and RB–DeepONet around 2.3×10^{-2}, with 95th-percentile errors below 3×10^{-2} and 3.3×10^{-2}, respectively. Thus, RB–DeepONet remains within a few percent of the FEM solution while relying only on residual information and never using solution labels, whereas POD–DeepONet requires full-order snapshots for supervised training. Both networks share the same low online complexity, but the Greedy RB trunk can be built from far fewer truth solves than the POD trunk.

Overall, this example shows that the EIM-based reduced operators yield a separable assembly with negligible online cost, and that RB–DeepONet can learn the RB–Galerkin map robustly in this non-affine setting, achieving POD-level accuracy with a fully reduced, label-free training pipeline.



(a) POD–DeepONet (N=5) (b) RB–DeepONet (N=5)

Fig. 7. Example 5.3: Training/validation loss.

Note that unlike Example 5.1, here we do not include FEONet as a baseline in this non–affine setting. The reason is due to the prohibitive training cost arising from per–sample full–order operator application: the original

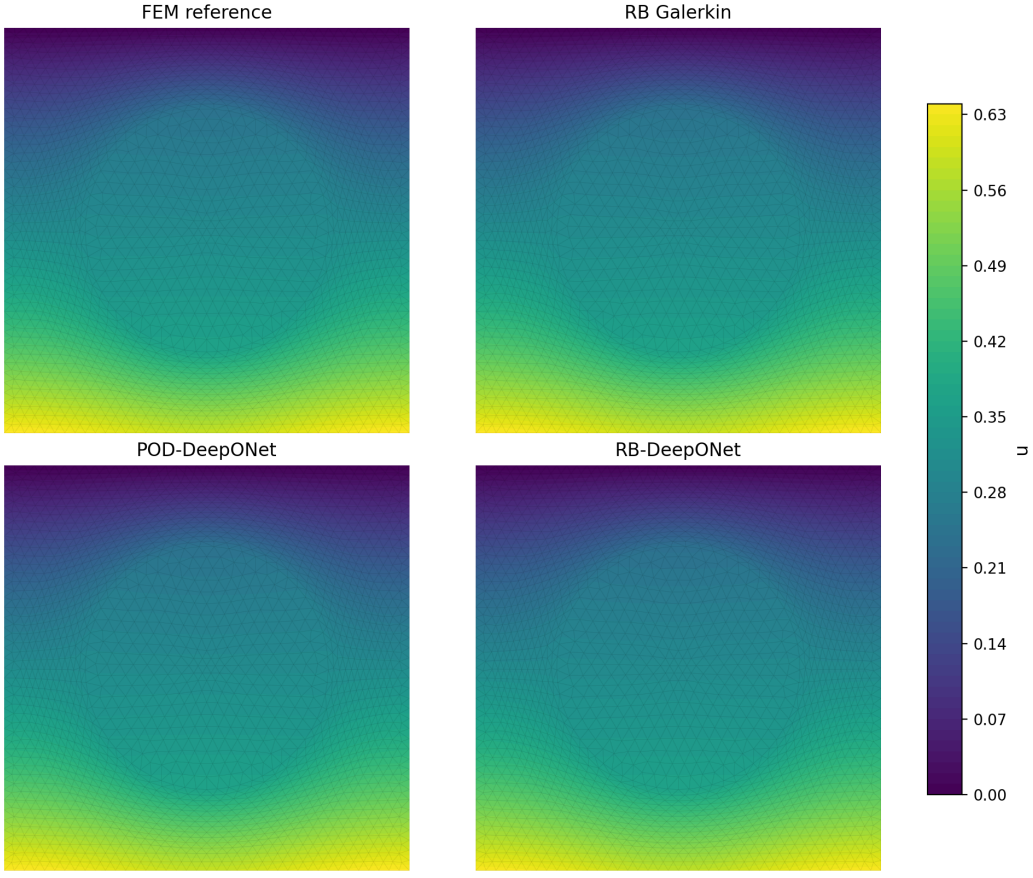


Fig. 8. Example 5.3: Representative-parameter comparison of (top–left) FEM reference, (top–right) RB–Galerkin, (bottom–left) POD–DeepONet, and (bottom–right) RB–DeepONet.

Trunk	**rel-L^2 (mean / $p95$)**	**rel-energy (mean / $p95$)**	**rel-residual (mean / $p95$)**
RB–DeepONet	$\mathbf{2.28 \times 10^{-2}}$ / $\mathbf{3.27 \times 10^{-2}}$	$\mathbf{3.74 \times 10^{-2}}$ / $\mathbf{4.79 \times 10^{-2}}$	$\mathbf{5.48 \times 10^{-2}}$ / $\mathbf{6.84 \times 10^{-2}}$
POD–DeepONet	9.14×10^{-3} / 2.08×10^{-2}	1.82×10^{-2} / 3.81×10^{-2}	1.71×10^{-2} / 4.71×10^{-2}
RB–Galerkin	1.57×10^{-4} / 4.83×10^{-4}	6.14×10^{-4} / 8.73×10^{-4}	1.05×10^{-15} / 3.23×10^{-15}

Table 4. Example 5.3: Test errors (mean / $p95$) over 1000 random parameters.

FEONet formulation [34] treats variable coefficients and boundary data as network inputs and applies the full operator during training on FE basis, without an offline affine surrogate. Consequently, when the operator changes with $\mathbf{k} = [k_1, k_2, k_3]^\top$, each training step requires assembling the full FEM operator $\mathbf{A}(\mathbf{k})$ at dimension N_h, which makes the per–epoch cost scale with N_h and the batch size. In contrast, our EIM–augmented RB formulation yields an online assembly whose evaluation cost scales only with the reduced dimension N and the EIM rank Q. To isolate the effect of the trunk space and to enable a fair and affordable comparison, we therefore benchmark only against the full FEM reference and the RB–Galerkin solution on the same reduced space.

Table 5 summarizes the network sizes and offline costs of the three surrogates. Across all examples, RB–DeepONet uses roughly 2×10^5 trainable parameters, comparable to POD–DeepONet but more than four times smaller than FEONet in Example 5.2. At the same time, the number of offline truth solves required by RB–DeepONet is reduced by two to three orders of magnitude compared with POD–DeepONet. FEONet avoids offline solves but pays for this with a much larger network and the need to evaluate full-order residuals during training. Therefore, RB–DeepONet achieves a favorable balance between sample efficiency, model size, and training cost.

Table 5. Trainable parameters and offline truth solves for three examples. Here d_{in} and d_{out} denote the input and output dimensions of the branch network, respectively.

Example	Method	d_{in}	# trainable params	Offline truth solves
5.1	RB–DeepONet	2	198,915	3
	POD–DeepONet	2	198,915	2,100
	FEONet	2	729,106	0
5.2	RB–DeepONet	147	288,977	337[a]
	POD–DeepONet	147	288,977	10,000
	FEONet	147	1,255,297	0
5.3	RB–DeepONet	3	199,685	5
	POD–DeepONet	3	199,685	4,000

[a] The construction of the source modes requires additionally solving the Riesz–map problems for each of the $r_f = 128$ source modes, so the total number of full-order solves used by RB–DeepONet in this example is $N + r_f = 337$.

6. Conclusions and future work

In this work, we introduced RB–DeepONet, a residual–driven operator–learning framework. The DeepONet trunk is fixed to a rigorously constructed reduced–basis (RB) space, and the branch network predicts the RB coefficients. This gives a principled interface between certified model reduction and operator learning: RB–DeepONet preserves the structure and stability of RB–Galerkin formulations while enabling fast, label–free coefficient prediction through a residual loss. We developed two specializations. In Case I, with fully parameterized coefficients and data, the RB right–hand side is assembled via a standard affine decomposition. In Case II, with operator parametrization and independently varying data, boundary and source functions are compressed into boundary and source modes and incorporated through lifting, without changing the training objective. On the analytical side, we established well–posedness of the RB system and generalization bounds for the residual loss. We also derived error estimates that quantify the impact of modal truncation. On the computational side, three benchmark problems showed that RB–DeepONet attains accuracy comparable to intrusive RB–Galerkin and to POD–DeepONet and FEONet surrogates. At the same time, it uses a compact, interpretable trunk and operates entirely in the reduced space. In particular, Case II shows that exogenous data (boundary conditions and loads) can be handled efficiently through a small number of offline modes, with online complexity proportional to the reduced dimension. The present work has several limitations. In Case II, Dirichlet data are enforced only through a finite boundary mode space, so the boundary conditions are satisfied up to a controlled but nonzero projection error that depends on the mode construction. In addition, the analysis is restricted to linear, coercive elliptic problems with an exact or EIM–based affine operator decomposition and to RB spaces fixed offline. Noncoercive, nonlinear, or strongly nonaffine problems fall outside the current theory. Future work will address these issues. We plan to develop more precise treatments of boundary constraints and adaptive trunk enrichment driven by *a posteriori* indicators. We also aim to extend the framework to time–dependent, nonaffine, and inverse problems, while retaining quantitative error control and reduced online cost.

Acknowledgment

This work was supported by the National Science Foundation (NSF) under grants DMS-2533878, DMS-2053746, DMS-2134209, ECCS-2328241, CBET-2347401 and OAC-2311848, and by the U.S. Department of Energy (DOE) Office of Science Advanced Scientific Computing Research program under award number DE-SC0023161, and the DOE–Fusion Energy Science program, under grant number: DE-SC0024583.

References

[1] P. Benner, S. Gugercin, K. Willcox, A survey of projection-based model reduction methods for parametric dynamical systems, SIAM review 57 (2015) 483–531.
[2] J. S. Hesthaven, G. Rozza, B. Stamm, et al., Certified reduced basis methods for parametrized partial differential equations, volume 590, Springer, 2016.
[3] M. G. Kapteyn, D. J. Knezevic, D. Huynh, M. Tran, K. E. Willcox, Data-driven physics-based digital twins via a library of component-based reduced-order models, International Journal for Numerical Methods in Engineering 123 (2022) 2986–3003.
[4] J. C. Strikwerda, Finite difference schemes and partial differential equations, SIAM, 2004.
[5] R. Eymard, T. Gallouët, R. Herbin, Finite volume methods, Handbook of numerical analysis 7 (2000) 713–1018.
[6] P. G. Ciarlet, The finite element method for elliptic problems, SIAM, 2002.
[7] M. Hinze, R. Pinnau, M. Ulbrich, S. Ulbrich, Optimization with PDE constraints, volume 23, Springer Science & Business Media, 2008.
[8] T. J. Sullivan, Introduction to uncertainty quantification, volume 63, Springer, 2015.
[9] D. J. Lucia, P. S. Beran, W. A. Silva, Reduced-order modeling: new approaches for computational physics, Progress in aerospace sciences 40 (2004) 51–117.
[10] Y. Maday, Reduced basis method for the rapid and reliable solution of partial differential equations, in: International Congress of Mathematicians, volume 3, European Mathematical Society Zürich, 2006, pp. 1255–1270.
[11] A. Quarteroni, A. Manzoni, F. Negri, Reduced basis methods for partial differential equations: an introduction, volume 92, Springer, 2015.
[12] Y. Liang, H. Lee, S. Lim, W. Lin, K. Lee, C. Wu, Proper orthogonal decomposition and its applications—part i: Theory, Journal of Sound and vibration 252 (2002) 527–544.
[13] M. Billaud-Friess, A. Nouy, Dynamical model reduction method for solving parameter-dependent dynamical systems, SIAM Journal on Scientific Computing 39 (2017) A1766–A1792.
[14] E. Lappano, F. Naets, W. Desmet, D. Mundo, E. Nijman, et al., A greedy sampling approach for the projection basis construction in parametric model order reduction for structural dynamics models, in: Proceedings of ISMA, 2016, pp. 19–21.
[15] J. S. Hesthaven, B. Stamm, S. Zhang, Efficient greedy algorithms for high-dimensional parameter spaces with applications to empirical interpolation and reduced basis methods, ESAIM: Mathematical Modelling and Numerical Analysis 48 (2014) 259–283.
[16] C. W. Rowley, T. Colonius, R. M. Murray, Model reduction for compressible flows using pod and galerkin projection, Physica D: Nonlinear Phenomena 189 (2004) 115–129.
[17] Q. Wang, N. Ripamonti, J. S. Hesthaven, Recurrent neural network closure of parametric pod-galerkin reduced-order models based on the mori-zwanzig formalism, Journal of Computational Physics 410 (2020) 109402.
[18] G. Rozza, D. B. P. Huynh, A. T. Patera, Reduced basis approximation and a posteriori error estimation for affinely parametrized elliptic coercive partial differential equations: application to transport and continuum mechanics, Archives of Computational Methods in Engineering 15 (2008) 229–275.
[19] S. A. McQuarrie, P. Khodabakhshi, K. E. Willcox, Nonintrusive reduced-order models for parametric partial differential equations via data-driven operator inference, SIAM Journal on Scientific Computing 45 (2023) A1917–A1946.
[20] M. D. Gunzburger, J. S. Peterson, J. N. Shadid, Reduced-order modeling of time-dependent pdes with multiple parameters in the boundary data, Computer methods in applied mechanics and engineering 196 (2007) 1030–1047.
[21] A. Cosimo, A. Cardona, S. Idelsohn, General treatment of essential boundary conditions in reduced order models for non-linear problems, Advanced Modeling and Simulation in Engineering Sciences 3 (2016) 7.
[22] M. Raissi, P. Perdikaris, G. E. Karniadakis, Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, Journal of Computational physics 378 (2019) 686–707.
[23] G. E. Karniadakis, I. G. Kevrekidis, L. Lu, P. Perdikaris, S. Wang, L. Yang, Physics-informed machine learning, Nature Reviews Physics 3 (2021) 422–440.
[24] L. Lu, X. Meng, Z. Mao, G. E. Karniadakis, Deepxde: A deep learning library for solving differential equations, SIAM review 63 (2021) 208–228.
[25] X. Meng, Z. Li, D. Zhang, G. E. Karniadakis, Ppinn: Parareal physics-informed neural network for time-dependent pdes, Computer Methods in Applied Mechanics and Engineering 370 (2020) 113250.
[26] S. Goswami, M. Yin, Y. Yu, G. E. Karniadakis, A physics-informed variational deeponet for predicting crack path in quasi-brittle materials, Computer Methods in Applied Mechanics and Engineering 391 (2022) 114587.
[27] R. G. Patel, N. A. Trask, M. A. Wood, E. C. Cyr, A physics-informed operator regression framework for extracting data-driven continuum models, Computer Methods in Applied Mechanics and Engineering 373 (2021) 113500.
[28] Z. Li, N. Kovachki, K. Azizzadenesheli, B. Liu, K. Bhattacharya, A. Stuart, A. Anandkumar, Fourier neural operator for parametric partial differential equations, arXiv preprint arXiv:2010.08895 (2020).
[29] L. Lu, P. Jin, G. E. Karniadakis, Deeponet: Learning nonlinear operators for identifying differential equations based on the universal approximation theorem of operators, arXiv preprint arXiv:1910.03193 (2019).
[30] T. Chen, H. Chen, Universal approximation to nonlinear operators by neural networks with arbitrary activation functions and its application to dynamical systems, IEEE transactions on neural networks 6 (1995) 911–917.

[31] Z. Li, H. Zheng, N. Kovachki, D. Jin, H. Chen, B. Liu, K. Azizzadenesheli, A. Anandkumar, Physics-informed neural operator for learning partial differential equations, ACM/IMS Journal of Data Science 1 (2024) 1–27.
[32] W. Hao, J. Wang, Laplacian eigenfunction-based neural operator for learning nonlinear partial differential equations, arXiv preprint arXiv:2502.05571 (2025).
[33] L. Lu, X. Meng, S. Cai, Z. Mao, S. Goswami, Z. Zhang, G. E. Karniadakis, A comprehensive and fair comparison of two neural operators (with practical extensions) based on fair data, Computer Methods in Applied Mechanics and Engineering 393 (2022) 114778.
[34] J. Y. Lee, S. Ko, Y. Hong, Finite element operator network for solving elliptic-type parametric pdes, SIAM Journal on Scientific Computing 47 (2025) C501–C528.
[35] J. S. Hesthaven, S. Ubbiali, Non-intrusive reduced order modeling of nonlinear problems using neural networks, Journal of Computational Physics 363 (2018) 55–78.
[36] N. Dal Santo, S. Deparis, L. Pegolotti, Data driven approximation of parametrized pdes by reduced basis and neural networks, Journal of Computational Physics 416 (2020) 109550.
[37] W. Chen, Q. Wang, J. S. Hesthaven, C. Zhang, Physics-informed machine learning for reduced-order modeling of nonlinear problems, Journal of computational physics 446 (2021) 110666.
[38] P. Sentz, K. Beckwith, E. C. Cyr, L. N. Olson, R. Patel, Reduced basis approximations of parameterized dynamical partial differential equations via neural networks, arXiv preprint arXiv:2110.10775 (2021).
[39] T. O'Leary-Roseberry, X. Du, A. Chaudhuri, J. R. Martins, K. Willcox, O. Ghattas, Learning high-dimensional parametric maps via reduced basis adaptive residual networks, Computer Methods in Applied Mechanics and Engineering 402 (2022) 115730.
[40] Q. Wang, J. S. Hesthaven, D. Ray, Non-intrusive reduced order modeling of unsteady flows using artificial neural networks with application to a combustion problem, Journal of computational physics 384 (2019) 289–307.
[41] K. Lee, K. T. Carlberg, Model reduction of dynamical systems on nonlinear manifolds using deep convolutional autoencoders, Journal of Computational Physics 404 (2020) 108973.
[42] S. A. McQuarrie, C. Huang, K. E. Willcox, Data-driven reduced-order models via regularised operator inference for a single-injector combustion process, Journal of the Royal Society of New Zealand 51 (2021) 194–211.
[43] B. Kramer, B. Peherstorfer, K. E. Willcox, Learning nonlinear reduced models from data with operator inference, Annual Review of Fluid Mechanics 56 (2024) 521–548.
[44] Q. Meng, Y. Li, Z. Deng, X. Liu, G. Chen, Q. Wu, C. Liu, X. Hao, A general reduced-order neural operator for spatio-temporal predictive learning on complex spatial domains, arXiv preprint arXiv:2409.05508 (2024).
[45] H. Zheng, Y. Chen, J. Han, Y. Yu, Rebano: Reduced basis neural operator mitigating generalization gaps and achieving discretization invariance, arXiv preprint arXiv:2509.09611 (2025).
[46] L. C. Evans, Partial differential equations, volume 19, American mathematical society, 2022.
[47] S. Chaturantabut, D. C. Sorensen, Nonlinear model reduction via discrete empirical interpolation, SIAM Journal on Scientific Computing 32 (2010) 2737–2764.
[48] S. Chaturantabut, D. C. Sorensen, Discrete empirical interpolation for nonlinear model reduction, in: Proceedings of the 48h IEEE Conference on Decision and Control (CDC) held jointly with 2009 28th Chinese Control Conference, IEEE, 2009, pp. 4316–4321.
[49] H. Brezis, H. Brézis, Functional analysis, Sobolev spaces and partial differential equations, volume 2, Springer, 2011.
[50] M. Anthony, P. L. Bartlett, Neural network learning: Theoretical foundations, cambridge university press, 2009.
[51] P. L. Bartlett, N. Harvey, C. Liaw, A. Mehrabian, Nearly-tight vc-dimension and pseudodimension bounds for piecewise linear neural networks, Journal of Machine Learning Research 20 (2019) 1–17.
[52] S. Shalev-Shwartz, S. Ben-David, Understanding machine learning: From theory to algorithms, Cambridge university press, 2014.
[53] S. Ko, S.-B. Yun, Y. Hong, Convergence analysis of unsupervised legendre-galerkin neural networks for linear second-order elliptic pdes, arXiv preprint arXiv:2211.08900 (2022).
[54] P. L. Bartlett, S. Mendelson, Rademacher and gaussian complexities: Risk bounds and structural results, Journal of machine learning research 3 (2002) 463–482.
[55] M. J. Wainwright, High-dimensional statistics: A non-asymptotic viewpoint, volume 48, Cambridge university press, 2019.
[56] A. W. Van Der Vaart, J. A. Wellner, Weak convergence, in: Weak convergence and empirical processes: with applications to statistics, Springer, 1996, pp. 16–28.