AIMS Mathematics, 11(3): 5853–5896.
DOI: 10.3934/math.2026242
Received: 26 December 2025
Revised: 13 February 2026
Accepted: 02 March 2026
Published: 09 March 2026

https://www.aimspress.com/journal/Math

Research article

DeepONet-based surrogate modeling for bond option pricing

Sanghyun Lee[1], **Jeonggyu Huh**[1] **and Seungwon Jeong**[2,*]

[1] Department of Mathematics, Sungkyunkwan University, Suwon 16419, Republic of Korea
[2] Global-Learning & Academic research institution for Master's Ph·D students, and Postdocs, Chonnam National University, Gwangju 61186, Republic of Korea

* **Correspondence:** Email: junior1492@jnu.ac.kr.

Abstract: Deep learning provides surrogates for computationally intensive interest rate option pricing, but practical deployment requires not only accurate prices but also reliable sensitivities and robustness to regime shifts. We presented a unified, task-driven comparison of a deep operator network (DeepONet), physics-informed neural networks (PINNs), and the deep backward stochastic differential equation (DeepBSDE) for European bond call options under the one-factor Hull–White and two-factor G2++ Gaussian affine term structure models. Using historical US Treasury term structures, we constructed smooth yield curve inputs and sampled broad option/model parameter configurations; closed-form formulas supply reference prices and analytical volatility sensitivities (Vegas). DeepONet was trained by supervised operator learning on price labels, whereas the PINN and DeepBSDE relied solely on partial differential equation (PDE) and backward stochastic differential equation (BSDE) constraints without price supervision. Although training is primarily aligned with pricing, we evaluated out-of-sample price accuracy together with automatic differentiation-based Vega accuracy and price robustness under an out-of-distribution volatility stress test. Across both models, supervised operator learning via DeepONet achieved the highest pricing accuracy on the held-out test set and the highest Vega accuracy, and exhibited the smallest degradation in pricing accuracy under the out-of-distribution volatility stress test.

Keywords: DeepONet; PINN; DeepBSDE; operator learning; interest rate model; option pricing
Mathematics Subject Classification: 91G15, 68T07, 65C20

1. Introduction

Deep learning surrogates are increasingly used to approximate option pricing maps in quantitative finance. Their practical value, however, depends on more than pointwise price accuracy. In downstream applications, a surrogate may match prices, while producing noisy automatic-differentiation (autodiff)

Greeks. Moreover, strong in-distribution (ID) performance does not necessarily translate into robust behavior under distribution shifts such as volatility regime changes. These requirements create an objective mismatch between how surrogates are trained and how they are used. This mismatch motivates an evaluation framework that is explicitly task-driven rather than price-driven alone.

This objective mismatch is particularly relevant for interest rate derivatives, where the natural market input is functional (a yield curve) rather than a small finite dimensional vector. Pricing a bond option therefore amounts to learning a mapping from a term structure to prices across heterogeneous contract and model configurations. This mapping is naturally viewed as an operator, and a deep operator network (DeepONet) was designed for such function-to-function (or function-to-value) mappings via a branch–trunk decomposition [4, 27].

The same pricing problem also admits partial differential equation (PDE) and backward stochastic differential equation (BSDE) characterizations via the Feynman–Kac correspondence [32, 37]. These viewpoints have, in turn, inspired theory-informed learning paradigms that constrain neural networks by the governing equation rather than by explicit price labels, including the physics-informed neural network (PINN) [34] and the deep backward stochastic differential equation (DeepBSDE) method [10, 17]. In what follows, we compare these three paradigms—DeepONet, PINN, and DeepBSDE—through the lens of the financial objectives they are intended to support.

Despite the rapid growth of deep learning-based option pricing, two gaps are especially salient in the term structure setting targeted by this paper. First, although DeepONet-style operator learning is designed for functional inputs, explicit studies that examine the yield curve to bond option pricing operators under classical interest rate models remain relatively limited. This contrasts with the extensive literature on equity options, where interest rates are often treated as scalars or fixed constants. Second, validation practice remains largely focused on price fit. For practical deployment, however, a surrogate must also provide reliable volatility sensitivities (Vegas) and behave stably under volatility regime shifts, which can be viewed as a controlled form of out-of-distribution (OOD) stress testing.

In this paper, we developed a task-driven benchmark for deep learning surrogates in interest rate derivative pricing, focusing on European call options on zero coupon bonds under Gaussian affine term structure models. We used the canonical one-factor Hull–White model [19] and the two-factor G2++ model [3] as testbeds, where closed-form pricing and Vega formulas provide clean ground truth. The initial yield curve is treated as a functional input, aligning the pricing problem with operator learning. Within a unified experimental pipeline, we trained and compared three paradigms: supervised DeepONet operator learning, PDE residual learning via the PINN approach, and BSDE-based learning via DeepBSDE. We evaluated out-of-sample pricing accuracy together with autodiff-based Vega accuracy. Separately, we assessed pricing robustness under a controlled volatility-only stress test.

Our contributions are threefold.

1. We introduced a unified, task-driven benchmark for European bond option pricing with functional yield curve inputs, comparing DeepONet, PINN, and DeepBSDE under the Hull–White and G2++ models with closed-form prices and analytical Vegas as ground truth.

2. We evaluated each surrogate along three practical axes: Out-of-sample pricing accuracy, autodiff-based Vega accuracy, and pricing robustness under a controlled volatility-only regime shift.

3. In our experiments, supervised DeepONet achieved the highest test set pricing accuracy and Vega

accuracy among the three methods and exhibited the smallest degradation in pricing accuracy under the volatility-only OOD stress test. These results offer practical guidance on when learned surrogates can be trusted for risk-sensitive fixed-income tasks.

To place these contributions in context, we briefly review three related strands of prior work: Supervised neural surrogates for option pricing, operator learning for parametric families of PDEs, and theory-informed approaches based on PDE or BSDE constraints.

A large body of work has studied supervised neural networks for option pricing and hedging. Early learning network approaches include [22, 29]. Subsequent deep learning implementations have explored architectural choices and extensions to higher dimensional or exotic payoffs [7, 11, 18]. More recently, supervised surrogates have also been developed for interest rate term structure settings. In these works, neural networks are trained on synthetically generated labels with yield and volatility term structure inputs to accelerate pricing under tractable short-rate or term structure models [23]. These studies demonstrate that supervised models can achieve high accuracy on the training distribution. However, they provide limited evidence on the stability of autodiff-based Greeks and performance when key risk factors—especially volatility—move outside the calibration range.

Neural operator methods view derivative pricing as learning an operator that maps functional inputs to prices or solution fields. A representative architecture is DeepONet, which couples a branch network that encodes the input function with a trunk network that encodes the query location, thereby targeting function to value (or function to function) mappings [27]. In quantitative finance, this operator perspective has been used to learn pricing maps over time and state variables, potentially with functional coefficients. For example, martingale-based objectives can avoid explicit PDE residuals while still enforcing a core no-arbitrage implication [6]. Operator learning has also been studied in probabilistic forward–backward stochastic differential equation (FBSDE) formulations, where the goal is to amortize repeated solves over families of contracts (e.g., recovering optimal stopping boundaries for new strikes without retraining) [2].

Related strands combine operator architectures with equation-based supervision. For example, a physics-informed DeepONet can be trained for the Heston model without labeled prices. It enforces the governing equation together with hard constraints at the terminal payoff. This approach yields strong price accuracy across parameter regimes [28]. At the same time, several works have noted that accurate prices do not automatically translate into accurate sensitivities, especially near challenging regions such as at-the-money (ATM). This has motivated derivative-aware objectives and regularization schemes for neural operators [33]. Complementary evidence from differential learning emphasizes that derivative losses can materially improve the quality of learned gradients and Hessians. It also shows that DeepONet-style constructions can be adapted to handle families of PDEs indexed by varying terminal conditions [25]. Despite this progress, operator learning studies explicitly tailored to term structure to interest rate option mappings remain comparatively scarce. Moreover, empirical validation is often less standardized with respect to Greek fidelity and robustness under controlled stress shifts. These gaps motivate a task-driven benchmark and call for a comparison with complementary equation-based approaches.

A complementary strand leverages the PDE formulation of derivative pricing. The PINN approach embeds the pricing PDE residual together with boundary and terminal conditions into the training objective, enabling label-free learning of price surfaces [34]. In quantitative finance, this idea has been pursued both in canonical option pricing PDE settings and in parametric PDE solvers that train a single

network across families of PDEs while targeting the fast evaluation of prices and Greeks [9, 15, 39]. The approach has been extended to more complex environments such as stochastic volatility, regime-switching, and exotic contracts, often requiring architectural or optimization refinements to stabilize residual-based training [5, 16, 31]. Fixed-income applications are beginning to appear, including PINN variants for Hull–White dynamics and related valuation tasks [35]. However, empirical validation remains frequently centered on price fit or residual diagnostics. Evidence on autodiff-based Greek fidelity and robustness under explicit stress shifts is less standardized, which motivates the objective-driven comparisons in this paper.

BSDE-based learning provides a probabilistic alternative grounded in the Feynman–Kac representation. Building on the foundational theory of BSDEs [32], deep BSDE solvers approximate the forward–backward system via Monte Carlo simulation and terminal payoff consistency, thereby avoiding the explicit second-order differentiation required by PDE residual objectives [10, 17]. Subsequent work has developed backward schemes and regression-based variants to improve stability and scalability, analyzed the approximation error, and adapted the framework to path-dependent features, barrier type events, and hedge-relevant sensitivities [1, 12, 13, 21, 30, 38, 40]. Nevertheless, fixed-income applications have focused on swaption-type products [38]. Controlled benchmarks for bond option pricing under classical Gaussian affine short-rate term structure models remain comparatively scarce. More broadly, validation is still heterogeneous and less standardized with respect to the joint assessment of price accuracy, sensitivity fidelity, and robustness under volatility stress tests.

Taken together, these strands suggest that controlled studies in term structure settings remain relatively scarce, and that validation is still less standardized with respect to Greek fidelity and robustness under explicit stress shifts. Against this background, we emphasize that our contribution is a controlled, task-driven benchmark rather than a new pricing architecture. Within a unified pipeline, we compare supervised operator learning (DeepONet) with two label-free, equation-informed paradigms (PINN and DeepBSDE) under identical yield curve inputs and matched parameter sampling. We evaluate out-of-sample pricing accuracy, autodiff-based Vega fidelity, and robustness under a volatility-only OOD stress test, with all metrics validated against closed-form prices and Vegas in the Hull–White and G2++ models.

The remainder of this paper is organized as follows. Section 2 reviews Gaussian affine term structure models and presents the PDE and BSDE representations of bond option pricing. Section 3 introduces the three surrogate paradigms and the evaluation protocols used to assess price accuracy, Vega accuracy, and robustness. Section 4 describes the experimental setup, including synthetic data generation from historical yield curves and architecture/training specifications, and reports numerical results for all three objectives under both Hull–White and G2++. Section 5 concludes with directions for extending this study.

2. Theoretical background

This section outlines the theoretical foundations used in our benchmark for interest rate derivative pricing. We first specify the reference interest rate models used for data generation. We then present the PDE and BSDE representations of the bond option price, which underpin the PINN and DeepBSDE baselines in Section 3.

2.1. Interest rate models

To ensure a rigorous evaluation of the surrogate models, we employ Gaussian affine term structure models. These models are characterized by their analytical tractability, guaranteeing the existence of closed-form solutions for standard instruments [20]. In this study, our target product is a European bond call option on a zero-coupon bond, and we denote its price as C. As in standard arbitrage-free implementations, the time-dependent drift terms are calibrated to match the initial term structure observed at $t = 0$. This calibration links the model's dynamics to the instantaneous forward rate curve $f(0, T)$.

2.1.1. The one-factor Hull–White model

As a baseline, we utilize the one-factor Hull–White model [19]. Under the risk-neutral measure $\mathbb{Q}$, the short-rate r_t is as follows:

$$dr_t = (\theta(t) - a \cdot r_t)dt + \sigma dW_t \tag{2.1}$$

where a denotes the speed of mean reversion, σ is the constant volatility, and W_t is a standard Wiener process under $\mathbb{Q}$. The deterministic function $\theta(t)$ is uniquely determined to match the initial forward rate curve $f(0, t)$.

This model is attractive as a benchmark because it is analytically tractable. However, with only one factor, rate movements across maturities are perfectly correlated, which limits its ability to represent non-parallel yield curve shifts. Since empirical bond returns are driven by multiple factors (e.g., level, slope, and curvature) [26], the one-factor model is structurally restrictive in that regard. The closed-form bond option price C used to generate the labels is reported in Appendix A.

2.1.2. The two-factor G2++ model

To address the aforementioned limitations and evaluate the surrogate models in a higher-dimensional setting, we use the G2++ model [3]. This model characterizes the short-rate r_t as the sum of two correlated stochastic factors, ξ_t and ν_t, and a deterministic calibration function $\phi(t)$

$$r_t = \xi_t + \nu_t + \phi(t).$$

The stochastic factors evolve according to the following Ornstein–Uhlenbeck (OU) processes:

$$\begin{aligned} d\xi_t &= -a\xi_t dt + \sigma dW_t^1, \\ d\nu_t &= -b\nu_t dt + \eta dW_t^2, \end{aligned}$$

with the correlation $dW_t^1 dW_t^2 = \rho dt$. Here a, b are mean-reversion speeds, σ, η are factor volatilities, and $\rho \in [-1, 1]$ controls dependence between the two Brownian drivers. As in the Hull–White model, $\phi(t)$ is calibrated to fit the initial term structure. Closed-form bond and bond-option formulas for label generation are provided in Appendix A.

2.2. Partial differential equation representation

Beyond the analytical formulas, the derivative price C can be characterized as the solution to a PDE. By the Feynman–Kac theorem, the risk-neutral expectation of a discounted payoff equals the solution

of a corresponding parabolic PDE [37]. This formulation forms the theoretical bedrock for the PINN approach used in this paper, where the network is trained by minimizing the PDE residual [34].

For the one-factor Hull–White model, the option price $C(t, r)$ is written as:

$$C(t,r) = \mathbb{E}^{\mathbb{Q}}\left[\exp\left(-\int_t^T r_s ds\right) g(r_T) \,\middle|\, r_t = r\right],$$

where $g(\cdot)$ denotes the terminal payoff at maturity T. Applying the Feynman–Kac theorem yields the following PDE on $t \in [0, T)$:

$$\frac{\partial C}{\partial t} + (\theta(t) - a \cdot r)\frac{\partial C}{\partial r} + \frac{1}{2}\sigma^2 \frac{\partial^2 C}{\partial r^2} - rC = 0, \quad C(T, r) = g(r).$$

Here, the term $\theta(t) - ar$ corresponds to the risk-neutral drift, and $\frac{1}{2}\sigma^2$ represents the diffusion coefficient.

In the G2++ framework, where the short-rate is decomposed as $r_t = \xi_t + \nu_t + \phi(t)$, the option price $C(t, \xi, \nu)$ admits the analogous risk-neutral representation

$$C(t,\xi,\nu) = \mathbb{E}^{\mathbb{Q}}\left[\exp\left(-\int_t^T (\xi_s + \nu_s + \phi(s))ds\right) g(\xi_T, \nu_T) \,\middle|\, \xi_t = \xi, \nu_t = \nu\right].$$

The corresponding two-dimensional PDE is given by

$$\frac{\partial C}{\partial t} - a\xi\frac{\partial C}{\partial \xi} - b\nu\frac{\partial C}{\partial \nu} + \frac{1}{2}\sigma^2\frac{\partial^2 C}{\partial \xi^2} + \frac{1}{2}\eta^2\frac{\partial^2 C}{\partial \nu^2} + \rho\sigma\eta\frac{\partial^2 C}{\partial \xi \partial \nu} - (\xi + \nu + \phi(t))C = 0, \quad C(T, \xi, \nu) = g(\xi, \nu).$$

The cross-derivative term reflects the factor correlation ρ and makes the PDE numerically more challenging than in the one-factor case.

2.3. BSDE formulation

The third perspective uses the stochastic counterpart of the Feynman–Kac theorem, which leads to a BSDE formulation. A BSDE is solved backward from the terminal payoff at $t = T$. This probabilistic perspective underlies the DeepBSDE approach [17] and avoids explicit spatial discretization, which is attractive in higher dimensions.

For the one-factor Hull–White model, let $Y_t = C(t, r_t)$ denote the option price process. Then (Y_t, Z_t) satisfies

$$-dY_t = -r_t Y_t dt - Z_t dW_t, \quad Y_T = g(r_T).$$

The driver term $-r_t Y_t$ represents the discounting effect at the short-rate. Financially, Y_t corresponds to the option value and Z_t is related to the hedging component, in particular, $Z_t = \sigma\frac{\partial C}{\partial r}$. The existence and uniqueness of solutions to such non-linear BSDEs were established in the seminal work [32].

In the G2++ framework, the BSDE becomes a two-dimensional system driven by correlated Brownian motions:

$$-dY_t = -(\xi_t + \nu_t + \phi(t))Y_t dt - Z_t^1 dW_t^1 - Z_t^2 dW_t^2, \quad Y_T = g(\xi_T, \nu_T).$$

Here, $Z_t = (Z_t^1, Z_t^2)$ is vector-valued and represents the sensitivities to the factor shocks. DeepBSDE learns Y_0 and the process Z_t by simulating the factor paths and enforcing the terminal payoff condition, thereby producing both prices and gradients without using a PDE grid.

3. Methodology

This section details the architectures and training methodologies for the three surrogate models evaluated in this study: DeepONet, PINN, and DeepBSDE. We present DeepONet as a data-driven supervised learner, while PINN and DeepBSDE are implemented as unsupervised, theory-informed solvers relying on the PDE and BSDE formulations derived in Section 2. For each learning architecture, we denote the collection of all trainable parameters as θ for simplicity. We then define the evaluation metrics (pricing accuracy, Vega accuracy, and OOD robustness) used in Section 4.

3.1. Data-driven operator learning (DeepONet)

We frame the pricing task not as a simple function approximation but as an operator learning problem. Let U denote a suitable function space of admissible initial yield curves $u : [0, \infty) \to \mathbb{R}$. Let a bounded domain $Y \subset \mathbb{R}^d$ denote the bounded domain of contract and model parameters y. We view the output as a price function on Y and define the output space as $V := L^2(Y)$ (or any comparable function space over Y). We then define the nonlinear pricing operator $G : U \to V$ by

$$G(u)(y) = C(u, y),$$

where $C(u, y)$ denotes the bond option price implied by the initial yield curve u and the configuration y that collects the relevant finite-dimensional variables. Thus, for each fixed curve $u \in U$, the function $y \mapsto G(u)(y)$ represents the entire bond-option pricing map over the parameter domain Y, and a single pricing query corresponds to evaluating $G(u)$ at the desired y.

To approximate this operator G, we use the DeepONet architecture as shown in Figure 1. Unlike standard neural networks that flatten all inputs into a single vector, DeepONet utilizes a dual-network structure to strictly separate the processing of the functional input u and the coordinate input y. This structural separation is crucial for effectively learning the operator's dependency on the yield curve shape while simultaneously handling the heterogeneous option parameters.

The first component, termed the branch net, is responsible for encoding the infinite-dimensional input function u. We implement this sub-network using a one-dimensional convolutional neural network (1D-CNN) to capture the inherent local dependencies and geometric features of the term structure, such as slope and curvature. Mathematically, the branch net maps the discretized yield curve u to a vector of latent basis coefficients, denoted $B_\theta(u) = [b_1(u), \ldots, b_p(u)]^\top$, where p represents the dimension of the latent space.

Simultaneously, the second component, termed the trunk net, processes the finite-dimensional parameter vector y such as maturity and strike. Since the elements of y represent distinct state variables without intrinsic spatial ordering, we use a standard multi layer perceptron (MLP) as a universal approximator. This sub network evaluates the learned basis functions at the coordinate y, outputting a vector of basis values denoted $T_\theta(y) = [t_1(y), \ldots, t_p(y)]^\top$. The final approximation of the operator, denoted G_θ, is obtained by the inner product of these features:

$$G_\theta(u)(y) = \langle B_\theta(u), T_\theta(y) \rangle.$$

The operator learning formulation makes DeepONet a natural modeling choice in the term structure setting. The market input is inherently functional (a yield curve), while practical use requires repeated

valuation across many heterogeneous contracts and model configurations under the same or similar curves. DeepONet explicitly separates these roles: The branch network learns a representation of the yield curve's shape, and the trunk network evaluates prices at a query point y. Once trained, this yields an amortized surrogate $G_\theta(u)(y)$ that supports fast evaluation for large batches of contracts and enables sensitivity computation via automatic differentiation with respect to the coordinates in y.

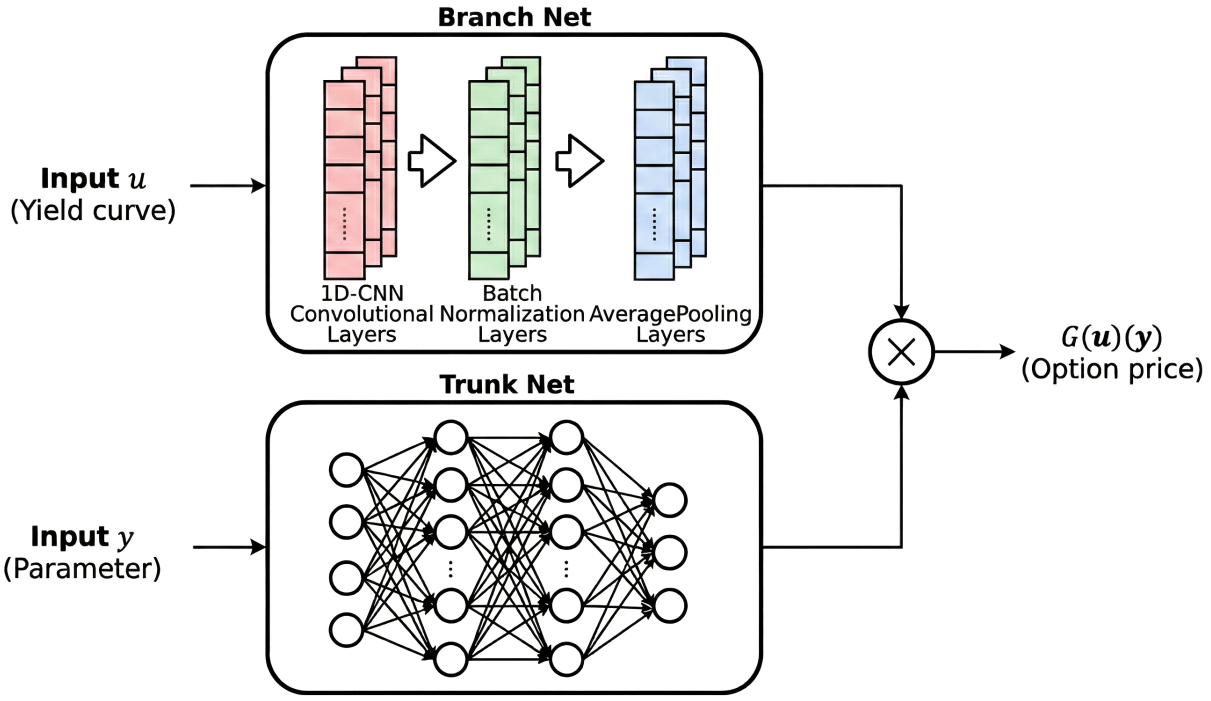


Figure 1. Schematic of the DeepONet architecture used in this study. The branch network encodes the discretized yield curve u into the latent coefficients $B_\theta(u)$, while the trunk network encodes the option/model input vector y into the features $T_\theta(y)$. The predicted option price is obtained via the inner product $G_\theta(u)(y) = \langle B_\theta(u), T_\theta(y)\rangle$.

The DeepONet approach differs fundamentally from the competing approaches considered below. PINNs and DeepBSDE are trained by enforcing equation-based constraints derived from the PDE/BSDE characterizations, without direct supervision from labeled prices. Consequently, their training involves either higher-order automatic differentiation of the PDE residual (PINN) or Monte Carlo path simulation and time discretization of the backward system (DeepBSDE). In contrast, DeepONet directly approximates the pricing operator from supervised price labels, targeting the mapping from functional market inputs to option prices over the parameter domain.

To rigorously justify the learning capability of the proposed DeepONet framework, we now analyze its statistical error decomposition. In this analysis, we view the DeepONet architecture described above as a parametric family of operators $\{G_\theta\}_{\theta\in\Theta}$ approximating the true pricing operator G, where the parameter set Θ consists of all weight configurations of such DeepONet architectures. Let $\mathfrak{D} = \{(u_i, y_i, z_i)\}_{i=1}^n$ denote the training dataset, where u_i is the discretized yield curve, y_i collects the option and model parameters, and $z_i = G(u_i)(y_i)$ is the corresponding analytical price. Using the squared loss,

$l(z, z^*) = (z - z^*)^2$, we define the population and empirical risks as

$$R(\theta) = \mathbb{E}_{(u,y)\sim\mu}\left[(G_\theta(u)(y) - G(u)(y))^2\right], \quad R_n(\theta) = \frac{1}{n}\sum_{i=1}^{n}(G_\theta(u_i)(y_i) - z_i)^2,$$

where μ is the joint distribution of the input pair (u, y). Following the standard learning-theoretic decomposition [36], we separate the total error into three contributions:

$$\varepsilon_{\text{app}} := \inf_\theta R(\theta), \quad \varepsilon_{\text{opt}}(K) := R_n(\theta_K) - \inf_\theta R_n(\theta), \quad \varepsilon_{\text{gen}}(n) := \sup_\theta |R(\theta) - R_n(\theta)|,$$

corresponding to the approximation, optimization, and generalization errors, respectively. Here, θ_K denotes the parameter vector obtained after K steps of stochastic gradient descent (SGD). We state the main decomposition result here and defer the detailed technical assumptions and supporting lemmas to Appendix B.

Theorem 1. *Let $R(\theta), R_n(\theta)$ and the error terms $\varepsilon_{app}, \varepsilon_{opt}(K)$, and $\varepsilon_{gen}(n)$ be defined as described above. Assume that the structural conditions on the pricing operator G, and Assumptions 2, 5, 6, and 8 in Appendix B hold. Then, for any architecture size N, sample size n, and number of iterations K, we have*

$$\mathbb{E}\left[R(\theta_K)\right] \le \varepsilon_{app} + \mathbb{E}\left[\varepsilon_{opt}(K)\right] + 2\varepsilon_{gen}(n),$$

where the expectation is taken with respect to the randomness of the iterates, conditional on the training sample $\mathfrak{D}$. Moreover, if the constants $\alpha > 0$ and $C > 0$ exist, independent of N, n, and K, such that

$$\varepsilon_{app} \le CN^{-\alpha}, \quad \mathbb{E}\left[\varepsilon_{opt}(K)\right] \le CK^{-1/2}, \quad \varepsilon_{gen}(n) \le Cn^{-1/2}.$$

Consequently,

$$\mathbb{E}\left[R(\theta_K)\right] \le C\left(N^{-\alpha} + K^{-1/2} + n^{-1/2}\right),$$

that is,

$$\mathbb{E}_{(u,y)\sim\mu}\left[(G_{\theta_K}(u)(y) - G(u)(y))^2\right] = O(N^{-\alpha}) + O(K^{-1/2}) + O(n^{-1/2}).$$

In particular, as $N \to \infty$, $K \to \infty$, and $n \to \infty$, the DeepONet estimator G_{θ_K} converges in the mean square to the true pricing operator,

$$\mathbb{E}_{(u,y)\sim\mu}\left[(G_{\theta_K}(u)(y) - G(u)(y))^2\right] = \mathbb{E}\left[R(\theta_K)\right] \to 0.$$

Proof. Fix the training sample $\mathfrak{D}$. For any $\theta \in \Theta$, we have

$$R(\theta) = R(\theta) - R_n(\theta) + R_n(\theta) \le |R(\theta) - R_n(\theta)| + R_n(\theta) \le \varepsilon_{\text{gen}} + R_n(\theta).$$

Adding and subtracting $\inf_\theta R_n(\theta)$ yields

$$R(\theta) \le \varepsilon_{\text{gen}} + \left(R_n(\theta) - \inf_\theta R_n(\theta)\right) + \inf_\theta R_n(\theta). \tag{3.1}$$

By the definitions of approximation and generalization errors, we have

$$\inf_\theta R_n(\theta) \le \inf_\theta \left(R(\theta) + |R(\theta) - R_n(\theta)|\right) \le \varepsilon_{\text{app}} + \varepsilon_{\text{gen}}(n). \tag{3.2}$$

Combining Eqs (3.1) and (3.2), we obtain

$$R(\theta) \le \varepsilon_{\text{gen}} + \left(R_n(\theta) - \inf_{\theta} R_n(\theta)\right) + 2\varepsilon_{\text{gen}}(n).$$

We now specialize to $\theta = \theta_K$, the parameter vector produced by K steps of the SGD. Taking the expectation with respect to the SGD's randomness and the definition of the optimization error give the announced decomposition

$$\mathbb{E}\left[R(\theta_K)\right] \le \varepsilon_{\text{app}} + \mathbb{E}\left[\varepsilon_{\text{opt}}(K)\right] + 2\varepsilon_{\text{gen}}(n).$$

The quantitative bounds follow from Appendix B. Under Assumption 2, Lemma 3 shows that the constants $C > 0$ and $\alpha > 0$ exist such that

$$\varepsilon_{\text{app}} = \inf_{\theta} R(\theta) \le CN^{-\alpha}.$$

Under Assumptions 5 and 6, Lemma 7 implies that the expected optimization error satisfies

$$\mathbb{E}\left[\varepsilon_{\text{opt}}(K)\right] = \mathbb{E}\left[R_n(\theta_K)\right] - \inf_{\theta} R_n(\theta) \le CK^{-1/2}.$$

Finally, under Assumption 8, Lemma 9 yields a high-probability bound

$$\varepsilon_{\text{gen}}(n) = \sup_{\theta\in\Theta} |R(\theta) - R_n(\theta)| \le Cn^{-1/2},$$

for each fixed architecture size N. Substituting these three estimates into the decomposition above and absorbing all constants into a generic constant $C > 0$ leads to

$$\mathbb{E}\left[R(\theta_K)\right] \le C\left(N^{-\alpha} + K^{-1/2} + n^{-1/2}\right),$$

and hence $\mathbb{E}\left[R(\theta_K)\right] \to 0$ as $N, K, n \to \infty$. This completes the proof. □

3.2. Physics-informed neural network (PINN)

The second surrogate paradigm directly exploits the PDE representation of the option price derived in Section 2.2. For each fixed choice of the model and option parameters, collected in a finite-dimensional vector y, we use $C(t, x)$ to denote the corresponding European bond option price, viewed as a function of time t and the state vector $x \in \mathcal{X} \subset \mathbb{R}^d$. In the one-factor Hull–White model, we have $d = 1$ and $x = r_t$, whereas in the two-factor G2++ model, we have $d = 2$ and $x = (\xi_t, \nu_t)$.

For each fixed parameter vector y with the option maturity T, the function C is characterized as the unique solution of the backward parabolic PDE

$$(\mathcal{P}_y C)(t, x) = 0, \quad (t, x) \in (0, T) \times \mathcal{X},$$

subject to the terminal condition

$$C(T, x) = g(x; y).$$

Here, $\mathcal{P}_y$ denotes the pricing differential operator associated with the parameter vector y. The dependence on y enters $\mathcal{P}_y$ through the model coefficients and calibration functions determined by the initial yield curve encoded in y.

Our aim is to approximate the family of solutions $\{C\}$ arising as y varies over a high-dimensional parameter space. We introduce a parameterized surrogate

$$G_{\mathrm{PINN}}(t, x, y; \theta)$$

where θ collects all trainable weights. For each fixed parameter vector y, the map $(t, x) \mapsto G_{\mathrm{PINN}}(t, x, y; \theta)$ is intended to approximate the corresponding solution $C(t, x)$ of the PDE with the operator $\mathcal{P}_y$ and terminal condition $g(\cdot; y)$. The vector y aggregates the option characteristics such as strike prices and maturity, the model parameters, and a finite-dimensional encoding of the initial yield curve. In this way, G_{PINN} can be interpreted as an operator-valued surrogate that takes y as the input and returns an approximate pricing surface in (t, x).

The training of G_{PINN} in this work is purely theory-informed: We do not use any observed or analytically generated option prices as direct supervision. Instead, the network is constrained only through the governing PDE and the associated terminal and boundary conditions. For a generic point (t, x, y), we define the PDE residual

$$R_{\mathrm{PDE}}(t, x, y; \theta) := (\mathcal{P}_y G_{\mathrm{PINN}})(t, x, y; \theta).$$

To define the PDE loss, we sample a collection of collocation points $\left\{t_j, x_j, y_j\right\}_{j=1}^{M_{\mathrm{PDE}}}$ in the time and state parameter domain. The parameter vectors y_j are taken from the same sampling ranges as described in Section 4. Given y_j with the option expiry T_j, we sample t_j uniformly from $[0, T_j]$. We then draw the state x_j from Gaussian distributions tailored to each model: In the Hull–White model, r_j is sampled from a Gaussian whose mean follows the term structure's implied level and whose variance is scaled consistently with the OU dynamics, while in G2++, (ξ_j, ν_j) is drawn from a centered Gaussian proposal with a fixed scale. The PDE loss is then defined as the mean squared residual over these collocation points

$$\mathcal{L}_{\mathrm{PDE}}(\theta) = \frac{1}{M_{\mathrm{PDE}}} \sum_{j=1}^{M_{\mathrm{PDE}}} \left|\mathcal{R}_{\mathrm{PDE}}(t_j, x_j, y_j; \theta)\right|^2.$$

In addition to the interior PDE constraint, we explicitly enforce the terminal payoff. For each parameter vector y_i with the maturity T_i, we sample the terminal states $\left\{x_k^{(i)}\right\}_{k=1}^{M_{\mathrm{TC}}^{(i)}}$ on the hypersurface $t = T_i$ and penalize deviations from the payoff

$$\mathcal{L}_{\mathrm{TC}}(\theta) = \frac{1}{\sum_i N_{\mathrm{TC}}^{(i)}} \sum_i \sum_{k=1}^{N_{\mathrm{TC}}^{(i)}} \left|G_{\mathrm{PINN}}\left(T_i, x_k^{(i)}, y_i; \theta\right) - g\left(x_k^{(i)}; y_i\right)\right|^2.$$

The total PINN loss minimized during training is given by

$$\mathcal{L}_{\mathrm{PINN}}(\theta) = \mathcal{L}_{\mathrm{PDE}}(\theta) + \beta \mathcal{L}_{\mathrm{TC}}(\theta),$$

with a positive weight $\beta > 0$ used to balance the enforcement of the interior PDE against the terminal condition. Stochastic gradient-based optimization is then applied to minimize $L_{\mathrm{PINN}}(\theta)$ over θ. Once the training has converged, the learned surrogate

$$(t, x, y) \longmapsto G_{\mathrm{PINN}}(t, x, y; \theta^*)$$

provides a mesh-free approximation of the corresponding pricing surface: For any new parameter vector y, evaluating $G_{\mathrm{PINN}}(t, x, y; \theta^*)$ yields an approximate option price surface consistent with the PDE formulation in Section 2.2.

3.3. Deep backward stochastic differential equation (DeepBSDE)

We next consider a surrogate construction based on the BSDE representation of the option pricing problem. As established in Section 2.3, for each choice of the parameters y and the associated state process $(X_t)_{t\in[0,T]}$, the pair $(Y_t, Z_t)_{t\in[0,T]}$ solves the BSDE

$$-dY_t = f(t, X_t, Y_t, Z_t; y)\,dt - Z_t \cdot dW_t, \quad Y_T = g(X_T; y),$$

where Y_t denotes the option value, Z_t is the corresponding hedging process, W_t is the driving Brownian motion, and the driver f is determined by the short-rate dynamics and the discounting structure. To approximate the family of BSDE solutions over the parameter space, we introduce an operator-valued network

$$G_{\mathrm{BSDE}}(t, x, y; \theta) = \left(G^Y_{\mathrm{BSDE}}(t, x, y; \theta), G^Z_{\mathrm{BSDE}}(t, x, y; \theta)\right),$$

where θ collects the trainable weights. The inputs (t, x, y) are defined consistently with Sections 3.2. The vector x represents the state variables and y encodes the model and option parameters together with a finite-dimensional description of the initial yield curve. For a fixed y, the map $(t, x) \mapsto G_{\mathrm{BSDE}}(t, x, y; \theta)$ is designed to approximate the solution pair (Y_t, Z_t) of the corresponding BSDE.

The DeepBSDE formulation is driven solely by the terminal payoff condition. At the level of the underlying stochastic model, the BSDE implies the identity

$$Y_T = Y_0 - \int_0^T f(s, X_s, Y_s, Z_s; y)ds + \int_0^T Z_s dW_s,$$

so that the terminal value Y_T is a functional of the entire path $(Y_s, Z_s)_{s\in[0,T]}$. This leads to the population loss

$$\mathcal{L}_{\mathrm{BSDE}}(\theta) = \mathbb{E}\left[\left|G^Y_{\mathrm{BSDE}}(T, X_T, y; \theta) - g(X_T; y)\right|^2\right],$$

where the expectation is taken with respect to the joint distribution of (X_T, y) under the risk-neutral dynamics and the sampling measure on the parameter space.

Although this loss is written explicitly only in terms of the Y-component, it depends on both outputs of the network. Indeed, through the BSDE relation above, any candidate terminal value $G^Y_{\mathrm{BSDE}}(T, X_T, y; \theta)$ arises from a pair of processes (Y_s, Z_s) that must jointly satisfy the backward dynamics. As a consequence, the functional $\mathcal{L}_{\mathrm{BSDE}}$ acts on the full pair $\left(G^Y_{\mathrm{BSDE}}, G^Z_{\mathrm{BSDE}}\right)$, and the gradient-based optimization of θ implicitly adjusts the representation of the hedging component G^Z_{BSDE} as well.

3.4. Performance evaluation framework

To provide a comprehensive comparison, we evaluated the surrogate pricing models along several complementary axes that reflect different aspects of practical utility in computational finance. All metrics were computed on held-out test sets generated from the reference pricing models, with training and validation performed on disjoint subsets of the data.

3.4.1. Out-of-sample pricing accuracy

First, we quantify the out-of-sample pricing accuracy of each surrogate, viewing it as an approximation of the reference pricing operator generated by the analytical and numerical solvers.

The most fundamental requirement for a surrogate pricing model is its ability to faithfully reproduce the ground-truth prices $z_i = C(t_i, x_i; y_i)$ on unseen inputs. For each surrogate $\hat{G} \in \{G_\theta, G_{\text{PINN}}, G_{\text{BSDE}}\}$, we regard it as a map $(t, x, y) \mapsto \hat{z} = \hat{G}(t, x, y)$ and evaluate its performance on a disjoint test set $\{(t_i, x_i, y_i)\}_{i=1}^M$.

The primary measure of out-of-sample accuracy is the mean square error (MSE) on the test set, defined as follows:

$$\text{MSE} = \frac{1}{M}\sum_{i=1}^{M}(z_i - \hat{z}_i)^2,$$

which provides an absolute measure of the average squared discrepancy between the surrogate and reference prices over the evaluation domain. Additionally, we report the coefficient of determination:

$$R^2 = 1 - \frac{\sum_{i=1}^M (z_i - \hat{z}_i)^2}{\sum_{i=1}^M (z_i - \bar{z})^2}, \quad \bar{z} = \frac{1}{M}\sum_{i=1}^{M} z_i,$$

which quantifies the fraction of variance in the reference prices explained by the surrogate. Values of R^2_{test} close to 1 indicate that the architecture provides an accurate approximation of the pricing operator. Beyond these aggregate statistics, we also analyze the structure of the pricing errors $\hat{z}_i - z_i$ by inspecting their empirical distribution and scatter plots against the true prices. These diagnostics allow for a visual assessment of potential systematic biases and heteroskedasticity across different regions of the state-parameter space.

3.4.2. Vega accuracy

Second, we assess the accuracy of the inferred risk sensitivities, Vegas, obtained via automatic differentiation of the learned operators. In many financial applications, accurate prices alone are insufficient; reliable sensitivities with respect to model parameters are equally critical for risk management and hedging. We therefore complement the price-level evaluation with an assessment of the Vega accuracy.

For each surrogate, we compute the model-implied Vegas by differentiating the learned mapping $\hat{G}(t, x, y)$ with respect to the volatility parameters. For a model output that depends on σ (and, in the G2++ specification, on both σ and η), we compute

$$V_\sigma = \frac{\partial \hat{z}}{\partial \sigma}, \quad V_\eta = \frac{\partial \hat{z}}{\partial \eta}$$

in the G2++ case, and the analogous single-parameter Vega V_σ for the Hull–White model. These model-derived Vegas are compared against the analytical Vegas obtained from the closed-form formulas of the reference models, using the identical test inputs employed for the price evaluation. The agreement between the analytical and surrogate Vegas is summarized via the R^2 score on the test set, computed between the pairs of Vegas rather than prices. The test set R^2 score computed on the Vega pairs captures how well each surrogate has learned not only the value of the pricing map but also its local gradient with respect to the key risk parameters, a property that is essential for stable and reliable hedging performance. In practical risk management, a Vega directly determines the sensitivity-based hedge ratio against volatility risk. Therefore, biased or noisy Vega estimates can lead to systematic under- or over-hedging and larger hedging profit and loss (P&L) errors, even when the price predictions are accurate.

3.4.3. OOD robustness

Third, we investigate the extrapolation performance under an OOD volatility stress test to probe the robustness of pricing accuracy beyond the training regime. Robustness to volatility shifts is particularly relevant in interest rate markets: Models are typically calibrated and validated under normal market conditions, yet they are often deployed for pricing and risk measurement precisely when volatility spikes and market dynamics become stressed. We therefore design a controlled OOD experiment that can be interpreted as a stress test in the volatility dimensions. Specifically, we ask whether a learned surrogate preserves the pricing accuracy when moving from a low-volatility business as usual regime to a high-volatility regime reminiscent of crisis-like episodes. Concretely, this OOD test is designed as a controlled volatility-only regime shift: We explicitly change only the volatility parameter's range, while keeping the sampling protocol for the yield curves and the remaining contract/model parameters consistent with the training regime.

We construct a restricted training regime in which the volatility parameters are confined to a lower range. For the Hull–White model, the training data are regenerated with σ constrained to a prescribed low-volatility interval, while for the G2++ model, both σ and η are restricted to analogous low ranges. Each candidate architecture is then retrained from scratch on this restricted dataset, adhering to the same training protocol as in the baseline experiments.

The resulting surrogates are evaluated on two disjoint test sets. The ID test set consists of samples satisfying the same volatility constraints imposed during training, thus examining the performance within the training regime. The OOD high-volatility test set is formed by samples drawn exclusively from a higher volatility range not observed during training, representing stressed conditions. For each model, we compute the test set R^2 score separately on the ID and OOD sets, denoted R^2_{ID} and R^2_{OOD}, respectively, and assess robustness according to the change in performance as follows:

$$\Delta R^2 = R^2_{\mathrm{OOD}} - R^2_{\mathrm{ID}}.$$

Values of ΔR^2 close to zero indicate that the surrogate largely preserves its pricing accuracy under volatility stress, whereas large negative values reflect substantial degradation when extrapolating from normal regimes to high-volatility regimes—an outcome that is undesirable in applications where reliable pricing under market stress is essential.

4. Numerical experiments and results

We evaluated the three learning architectures on a common dataset of European bond options generated from historical term structure data under the Hull–White and G2++ models. The numerical results in this section compare their out-of-sample performance—both for the ID scenario and under the controlled OOD volatility stress test—and illustrate the effect of incorporating a model-based structure through PDE and BSDE formulations.

4.1. Experimental setup

The numerical study is built around a common experimental pipeline shared by the three architectures considered in this work. In this subsection, we describe how the supervised dataset of bond-option prices is constructed from historical term structure data, how the neural architectures for

DeepONet, PINN, and DeepBSDE are specified, and which training protocol and data-splitting strategy are adopted for model comparison.

4.1.1. Synthetic data generation and parameter configuration

The numerical experiments are based on daily US Treasury yield curve data over the period 2010–2024, covering 3,752 trading days. For each trading day, the term structure is observed at 13 standard maturities, ranging from short-term horizons starting at 1 month to long-term horizons extending up to 30 years. These discrete observations are lifted to a continuous yield curve $u(t)$ by fitting a cubic smoothing spline to the observed points for that day. Representative examples of the resulting smoothed term structures are shown in Figure 2. We use the resulting smooth term structure in two roles: (i) As the functional input to DeepONet and (ii) as the initial term structure used to calibrate the Hull–White and G2++ models.

For any bond maturity T_B, we denote the associated zero coupon bond price process as $S_t = P(t, T_B)$, which serves as the underlying asset for the corresponding European bond call option. The smoothed yield curve uniquely determines the initial discount curve and instantaneous forward rates, and thereby fully specifies the calibration functions entering the no-arbitrage dynamics of the short-rate and factor processes (see Appendix A). These quantities are used in the analytical pricing formulas, which provide ground-truth option prices for supervised training and for evaluation of all methods.

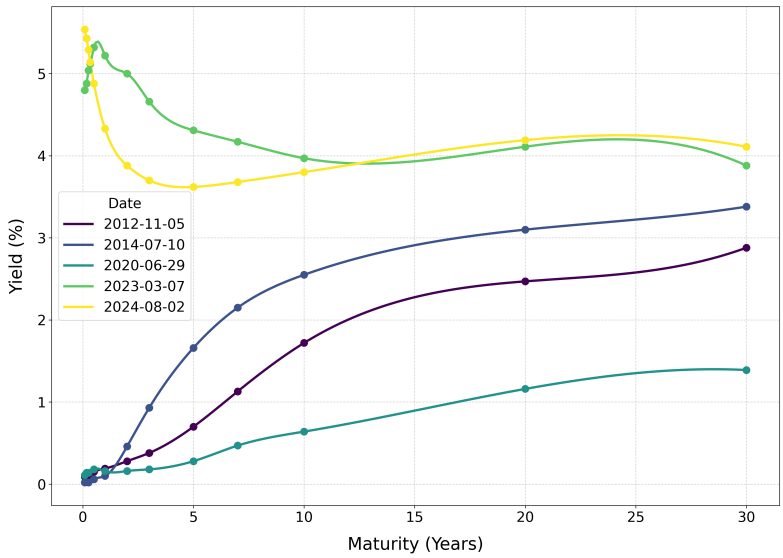


Figure 2. Representative U.S. Treasury yield curves reconstructed by fitting a cubic smoothing spline to the observed maturities (13 standard tenors). The resulting smooth term structure $u(t)$ is used both as the functional input to DeepONet and as the initial term structure for calibrating the Hull–White and G2++ models.

Option configuration. Given a continuous yield curve $u(t)$ for a particular trading day, we define a family of European bond-option contracts under the Hull–White and G2++ models. For each yield curve, we draw a fixed number N_s of option instances. For each instance, we first sample an option maturity T and an underlying bond maturity T_B from the ranges reported in Table 1, subject to the

constraint $0 < T < T_B$, so that the option expires at time T and is written on a zero coupon bond maturing at T_B.

Table 1. Sampling distributions for the option and model parameters. Here, $F(0, T, T_B)$ denotes the forward price defined as $P(0, T_B)/P(0, T)$, where $P(0, T)$ is the price of a zero coupon bond maturing at T.

Model	Parameter	Description	Sampling distribution
Both	T	Option expiry	$\mathcal{U}(0.5, 5.0)$
	T_B	Bond maturity	$T + \mathcal{U}(1.0, 10.0)$
	K	Strike	$F(0, T, T_B) \times \mathcal{U}(0.9, 1.1)$
Hull–White	a	Mean reversion speed	$\mathcal{U}(0.05, 0.5)$
	σ	short-rate volatility	$\mathcal{U}(0.01, 0.15)$
G2++	a	Mean reversion speed of ξ	$\mathcal{U}(0.05, 1.0)$
	b	Mean reversion speed of ν	$\mathcal{U}(0.01, 0.5)$
	σ	Volatility of ξ	$\mathcal{U}(0.01, 0.1)$
	η	Volatility of ν	$\mathcal{U}(0.01, 0.1)$
	ρ	Correlation	$\mathcal{U}(-0.9, 0.9)$

Model parameters are sampled independently from the corresponding distributions in Table 1: (a, σ) for the Hull–White specification and $(a, b, \sigma, \eta, \rho)$ for the G2++ specification. This sampling strategy covers a broad range of interest rate environments around the calibrated term structures while preserving the arbitrage-free structure of each model.

The strike price K is specified relative to the ATM forward price of the underlying bond at time zero

$$F(0, T, T_B) = \frac{P(0, T_B)}{P(0, T)},$$

by sampling a moneyness multiplier κ uniformly from a prescribed interval $[\kappa_{\min}, \kappa_{\max}]$ around one and setting

$$K = \kappa F(0, T, T_B).$$

Values $\kappa > 1$ correspond to out-of-the-money call options, $\kappa < 1$ to in-the-money calls, and $\kappa = 1$ to ATM contracts. In the numerical experiments, we chose $[\kappa_{\min}, \kappa_{\max}] = [0.9, 1.1]$, yielding a representative range of moneyness levels around the forward price as shown in Table 1. We fixed the number of option instances per yield curve to $N_s = 20$, which allows us to explore a diverse range of option configurations for each observed term structure while keeping the sampling scheme uniform across trading days.

Data generation. For each complete input pair (u, y), consisting of the smoothed yield curve u and the option/model parameter vector y, we evaluate the corresponding analytical pricing formula under the Hull–White or G2++ model to obtain the reference option price z. Recall that the parameter set y is T, T_B, K, a, σ for the Hull–White model and $T, T_B, K, a, b, \sigma, \eta, \rho$ for G2++. In the numerical implementation, we further augment y with two deterministic features derived from (T, T_B, K): The

time to maturity $\tau := T_B - T$ and the moneyness $\kappa = K/F(0, T, T_B)$. We denote the resulting augmented parameter vector as $\tilde{y} := (y, \tau, \kappa)$. We then discretize the yield curve $u(t)$ on a fixed grid of standard maturities $\{t_k\}$ and store the resulting vector representation $u_i = (u(t_k))_k$, together with the associated parameter vector y_i and the analytical price z_i.

Repeating this procedure across all trading days yields a dataset of 75,040 option price samples (3,752 trading days × 20 options per day). The resulting dataset is then split into training, validation, and test sets as described in Section 4.1.3. While the resulting dataset provides exact option prices used for supervised training of DeepONet, the same data are used solely for evaluation purposes in the PINN and DeepBSDE frameworks.

4.1.2. Model architectures

DeepONet. We first describe DeepONet with a branch–trunk architecture. The branch network is a 1D-CNN that maps the discretized yield curve (u) to a latent representation in $\mathbb{R}^p$ with $p = 256$. The branch network consists of three convolutional blocks with the channel sizes of 32, 64, and 128, respectively. The first two blocks perform downsampling via one-dimensional average pooling layers with a kernel size of 2 and a stride of 2 (`AvgPool1d(2,2)`), while the final block applies global pooling through an one-dimensional adaptive average pooling layer producing a single output per channel (`AdaptiveAvgPool1d(1)`). The pooled features are flattened and passed through a small and fully connected network consisting of a hidden linear layer with a width of 256 with rectified linear unit (ReLU) activation and dropout with a probability of 0.2, followed by a final linear projection to the latent dimension p.

The trunk network is implemented as a fully connected MLP that maps the finite-dimensional option/model input $\tilde{y}$ to a latent feature vector in $\mathbb{R}^p$ with $p = 256$. Specifically, it consists of two hidden linear layers with widths of 256 and 512, respectively; each hidden layer is followed by batch normalization, a Gaussian error linear unit (GELU) activation, and dropout with a probability of 0.2. A final linear layer then projects the resulting representation to the p-dimensional trunk feature vector.

PINN. For the PDE-based surrogate, we represent the pricing function by a fully connected network $G_{\text{PINN}}(t, x, \tilde{y}; \theta)$. For the Hull–White model, G_{PINN} is implemented as an MLP with hidden width of 256 and six hidden layers, using hyperbolic tangent activations; the final output is passed through a Softplus activation to enforce non-negativity of the prices.

For the G2++ model, we use an MLP of the same hidden width and depth, where each hidden layer is followed by batch normalization, a GELU activation, and dropout with a probability of 0.2, and the output is again mapped through Softplus. The PDE residual is constructed via automatic differentiation from PyTorch, requiring first- and second-order derivatives.

DeepBSDE. For the BSDE-based surrogate, we parameterize the solution pair (Y_t, Z_t) using two neural components. An initial-value network N_{Y_0} represents Y_0 as a function of the parameter encoding $\tilde{y}$. In our implementation, N_{Y_0} is an MLP with a hidden width of 256 and six hidden layers, where each hidden layer is followed by batch normalization, a GELU activation, and dropout with a probability of 0.2; the output of N_{Y_0} is passed through Softplus to enforce non-negativity.

A second time-dependent network N_Z outputs Z_t along simulated paths. For the Hull–White model, N_Z outputs a scalar and is implemented as an MLP with a hidden width of 128 and four hidden layers,

again using batch normalization, GELU activations, and dropout with a probability of 0.2. For G2++ model, N_Z outputs a two-dimensional vector corresponding to the two Brownian drivers.

4.1.3. Training protocol and data split

For the supervised DeepONet, we constructed an instance-level random split of the generated option price dataset into training, validation, and test sets with proportions of 70%/15%/15% using a fixed random seed. We did not use time-series cross-validation or date-blocked splitting in this benchmark, as our goal is to learn a pricing operator over a distribution of yield curves rather than to perform time-series forecasting. The training split is used exclusively to fit DeepONet by minimizing the price loss, while the validation split is used for model selection and early stopping based on the validation MSE computed against the same closed-form reference prices. The theory-informed models are trained without direct price supervision—using only their PDE/BSDE-based objectives—yet they are evaluated on the same held-out test set as DeepONet to ensure a consistent out-of-sample comparison across paradigms.

Training objectives. DeepONet is trained by minimizing the empirical risk $R_n(\theta)$ defined in Section 3.1, i.e., MSE regression against the closed-form reference prices. In contrast, PINN minimizes the theory-informed objective

$$\mathcal{L}_{\text{PINN}}(\theta) = \mathcal{L}_{\text{PDE}}(\theta) + \beta \mathcal{L}_{\text{TC}}(\theta)$$

where we fix $\beta = 10$ in our implementation, and DeepBSDE minimizes the BSDE-based terminal condition loss $\mathcal{L}_{\text{BSDE}}(\theta)$ introduced in Section 3.3. Although PINN and DeepBSDE do not use reference prices in their optimization objective, price errors against the same closed-form prices are still computed on the validation and test sets to quantify their performance using a common metric across methods.

Optimization settings and model selection. Unless stated otherwise, all models are trained using the Adam optimizer with the initial learning rate 3×10^{-4} with the weight decay 1×10^{-5}. We use the default values for other Adam hyperparameters. For the weight initialization method, we use PyTorch's default parameter initialization (i.e., no custom weight initialization) for all networks. We further use a cosine-annealing warm-restart learning rate schedule (CosineAnnealingWarmRestarts with $T_0 = 100$, $T_{\text{mult}} = 2$, and $\eta_{\min} = 10^{-7}$). Mini batch training uses batch size 256, while the validation and test evaluations use a batch size of 512. We trained up to 1,000 epochs and applied gradient clipping with a maximum norm of 1.0 to prevent exploding gradients. Early stopping was based on the validation mean-squared error: Training is terminated if the validation score does not improve for 100 consecutive epochs. For reproducibility, we fix a random seed across Python/NumPy/PyTorch and enforce deterministic compute unified device architecture (CUDA) behavior.

Numerical discretization and collocation points sampling. For PINN, the PDE residual is enforced at randomly sampled collocation points. For each option instance in a mini batch, collocation times are drawn uniformly from $[0, T]$, where T denotes the option expiry of that instance, and state proposals are sampled from Gaussian distributions tailored to each model: For the Hull–White model, we draw r from a Gaussian whose mean follows the term structure-implied level and whose variance is scaled

consistently with the OU dynamics, while for the G2++ model, we draw the factor state (ξ, ν) from a zero-mean Gaussian proposal with a fixed standard deviation of 0.05. In our implementation, we sample one interior collocation time per option instance in each mini batch (thus the number of collocation points per training iteration equals the mini batch size during training). We also enforce the terminal condition using one terminal-state sample per option instance in the mini batch.

For DeepBSDE, the forward–backward system is evolved on a uniform time grid with 10 time steps (i.e., $\Delta t = T/10$). Each training iteration simulates a mini batch of Monte Carlo paths, where the number of simulated paths equals the mini batch size. The backward component (Y, Z) is advanced by an Euler–Maruyama-type discretization

$$Y_{n+1} = Y_n + r_n Y_n \Delta t + Z_n \Delta W_n,$$

which introduces a standard discretization bias controlled by Δt. In contrast, the underlying Hull–White and G2++ factor processes are simulated using their exact Gaussian one-step transitions over each time step; in the G2++ case, correlated Brownian increments are constructed to match the prescribed correlation parameter ρ.

4.2. Price prediction accuracy

We began by evaluating how accurately each surrogate reconstructs the option prices from held-out test data. The comparison is summarized in Table 2 using the test set MSE and the coefficient of determination (R^2) as described in Section 3.4.1.

Table 2. Out-of-sample price prediction performance on the held-out test set for DeepONet, PINN, and DeepBSDE under the Hull–White and G2++ models. We report the MSE and the R^2.

Model	Architecture	MSE	R^2
Hull–White	DeepONet	3.944×10^{-5}	0.9916
	PINN	4.339×10^{-4}	0.9021
	DeepBSDE	6.560×10^{-4}	0.8663
G2++	DeepONet	9.361×10^{-5}	0.9750
	PINN	1.399×10^{-4}	0.6267
	DeepBSDE	9.767×10^{-4}	0.7393

Table 2 summarizes the held-out test price level's accuracy and shows a consistent ranking across methods under both the Hull–White and G2++ models. The fully supervised DeepONet achieved the highest agreement with the analytical benchmark, with test set $R^2 = 0.9916$ for the Hull–White model and $R^2 = 0.9750$ for the G2++ model. By contrast, PINN and DeepBSDE achieved lower price-level accuracy than DeepONet on the held-out test set. Both methods are trained without price labels and rely instead on PDE/terminal constraints or BSDE terminal constraints. This gap is modest in the one-dimensional Hull–White setting ($R^2 = 0.9021$ for PINN and $R^2 = 0.8663$ for DeepBSDE). It becomes substantially larger in the two-dimensional G2++ setting ($R^2 = 0.6267$ for PINN and $R^2 = 0.7393$ for DeepBSDE).

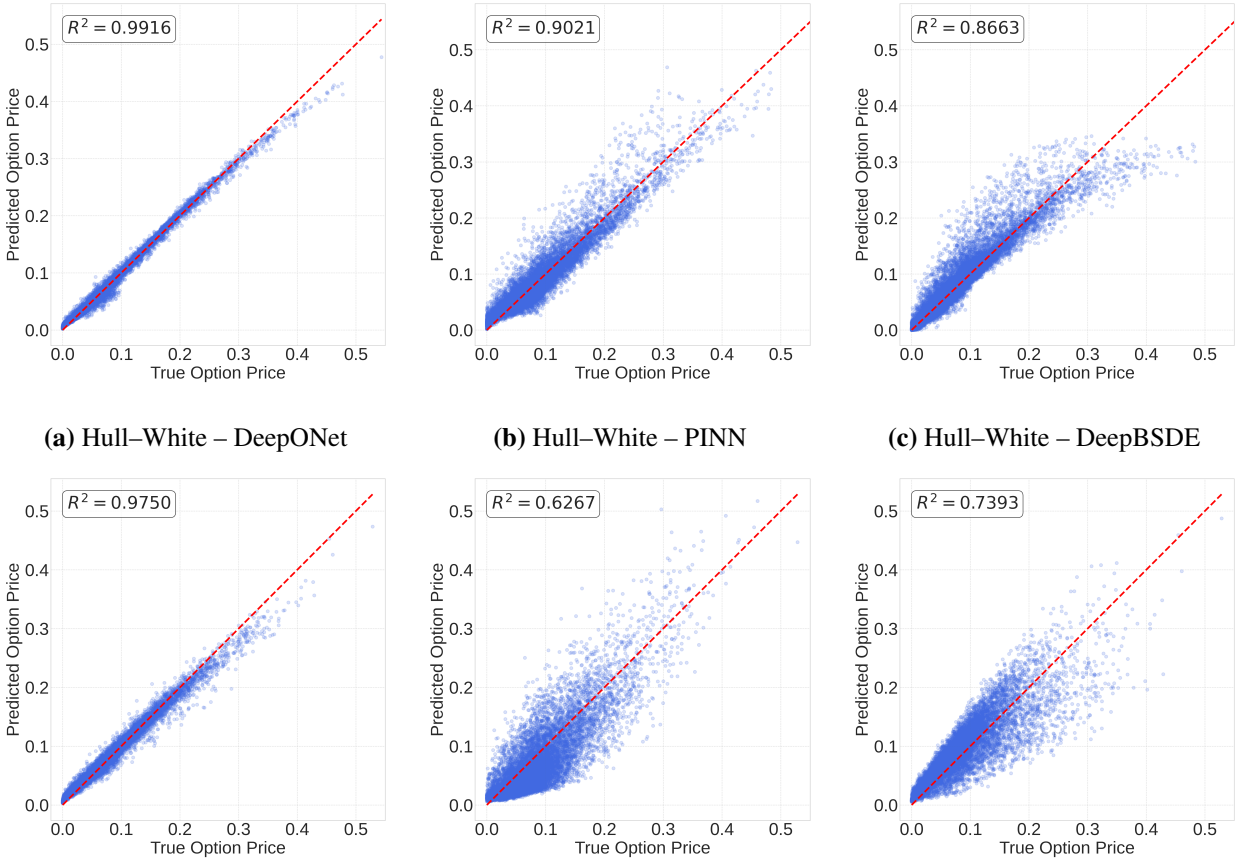
The performance ordering in Table 2 reflects the alignment between the training objective and the evaluation metric. This performance gap also reflects the numerical difficulty of recovering a highly accurate pricing map from indirect constraints over a broad, highly parameterized domain. Since DeepONet is optimized by directly minimizing a price regression loss using analytical labels, the test set MSE/R^2 is naturally most favorable to this supervised baseline. By construction, PINN and DeepBSDE must infer the pricing relationship through theoretical constraints, so their price errors also incorporate the optimization and discretization effects associated with those constraints.

The lower G2++ price accuracy of PINN is consistent with the structure of its learning objective. PINN enforces a local PDE residual loss that repeatedly differentiates the network's output up to the second order, which can amplify the numerical error and induce a stiff optimization landscape. As the underlying model moves from one to multiple factors, these effects tend to compound. The state dimension increases, the parameter interactions become stronger, and the PDE residual involves a richer combination of derivative terms. This makes it harder for collocation-based enforcement to maintain a uniformly accurate curvature across the domain. In such settings, small local irregularities in the learned surface can translate into larger pointwise pricing errors, which is consistent with the sizable drop observed for PINN in Table 2.

DeepBSDE is anchored by a global terminal payoff constraint propagated along the simulated paths. In our implementation, the forward factors are simulated using exact one-step Gaussian transitions. Together, these choices provide a smoother learning signal than pointwise second-derivative residual minimization. Consistent with this, DeepBSDE achieved higher test set price accuracy than PINN in the G2++ setting in Table 2, despite both methods being trained without price labels.

To complement the aggregate statistics, Figure 3 visualizes the predicted versus analytical prices on the test set. DeepONet exhibits tight clustering around the identity line, whereas both theory-informed models display a wider dispersion; the separation is visibly more pronounced in the G2++ case, consistent with the larger performance gaps in Table 2. Beyond these global measures, Appendix D further reports error–price scatter plots and empirical error histograms, providing a more granular view of the systematic bias and heteroskedastic behavior across the spectrum of option values.

Overall, in this benchmark and under the reported training protocols, supervised learning with price labels achieved higher held-out test pricing accuracy than the theory-informed objectives, as shown in Table 2 and Figure 3, with the largest gap observed in the two-factor G2++ model. This comparison is conducted solely in terms of price-level accuracy on a test set. DeepONet is trained by directly minimizing a price-based loss using analytical labels. Therefore, this metric is particularly well aligned with its training objective. At the same time, these findings highlight an important limitation of price-level evaluation alone: High accuracy in the option prices does not necessarily imply accurate learning of the sensitivities. This limitation motivates the next section, where we turn to a detailed analysis of Vega approximation performance.



(a) Hull–White – DeepONet **(b)** Hull–White – PINN **(c)** Hull–White – DeepBSDE

(d) G2++ – DeepONet **(e)** G2++ – PINN **(f)** G2++ – DeepBSDE

Figure 3. Predicted versus analytical option prices on the test set. The first row reports the results for the Hull–White one-factor interest rate model, while the second row corresponds to the G2++ two-factor model. The three columns display results obtained using DeepONet, PINN, and DeepBSDE, respectively. The dashed red line indicates perfect prediction. Each point corresponds to one option sample in the held-out test set.

4.3. Vega accuracy

Vega—the sensitivity of the option price to volatility parameters—plays a central role in hedging and risk monitoring. Following the evaluation protocol in Section 3.4.2, we computed model-implied Vegas via automatic differentiation of the trained surrogate price map and the reported test set accuracy using the R^2 score against the corresponding analytical Vega. For the Hull–White model, we evaluate sensitivity with respect to the short-rate volatility σ, whereas for the G2++ model, we consider both σ and η. Table 3 summarizes the results.

Table 3 reports the held-out test Vega accuracy for each method and model. First, Vega prediction is systematically more challenging than price prediction. The R^2 values are markedly lower than those reported for prices in Table 2. This highlights that matching the function values does not automatically guarantee accurate recovery of the local derivatives. Second, among the three paradigms, DeepONet provides the most consistently accurate Vegas across both models.

Table 3. Out-of-sample Vega prediction accuracy on the held-out test set for DeepONet, PINN, and DeepBSDE under the Hull–White and G2++ models. Accuracy is measured by the R^2 for V_σ (Hull–White) and for V_σ and V_η (G2++).

Model	Architecture	Vega $R^2(\sigma)$	Vega $R^2(\eta)$
Hull–White	DeepONet	0.8442	–
	PINN	0.7048	–
	DeepBSDE	0.6294	–
G2++	DeepONet	0.6153	0.6786
	PINN	0.1528	0.2897
	DeepBSDE	0.6011	0.3340

In the Hull–White case, DeepONet achieves the highest Vega accuracy ($R^2 = 0.8442$), followed by PINN ($R^2 = 0.7048$) and DeepBSDE ($R^2 = 0.6294$). This ranking suggests that, in the present experimental regime, learning the price surface directly over a broad range of volatility inputs leads to the most reliable reconstruction of the volatility gradient. DeepONet is trained with supervised price labels and treats σ as an explicit input coordinate. As a result, it tends to learn a smoother and more globally coherent dependence on σ. This translates into more stable autodiff-based sensitivities. By contrast, the theory-informed approaches recover sensitivities implicitly through their training objectives. PINN's local residual enforcement can introduce local irregularities that are magnified by differentiation. DeepBSDE avoids explicit second-derivative residual minimization, but it remains affected by Monte Carlo and time discretization noise along the simulated paths.

The contrast becomes even sharper in the two-factor G2++ setting. DeepONet continues to perform well, achieving $R^2 = 0.6153$ for V_σ and $R^2 = 0.6786$ for V_η. DeepBSDE achieves a comparable performance for V_σ ($R^2 = 0.6011$) but drops substantially for V_η ($R^2 = 0.3340$). PINN, by comparison, exhibits very limited Vega fidelity in this model ($R^2 = 0.1528$ for V_σ and $R^2 = 0.2897$ for V_η). These results indicate that sensitivity estimation can deteriorate rapidly as the model's complexity increases, and that the relative ranking across paradigms may shift depending on the dimension and interaction structure. One structural explanation is that in G2++, PINN enforces a PDE residual containing not only second-order state derivatives but also mixed cross-derivative terms. This residual must be balanced over an expanded parameter space (σ, η, ρ). The resulting optimization problem can be stiff and error-sensitive. Any residual local irregularities are then amplified when differentiating with respect to volatility. DeepBSDE avoids such explicit high-order residual enforcement, but its gradient fidelity can still degrade in higher dimensions due to pathwise noise, which is consistent with the pronounced drop observed for V_η.

To move beyond aggregate scores and better understand the structure of sensitivity errors, we provide a detailed Vega error analysis in Appendix E. We examine the empirical distributions of the Vega errors and their dependence on strike (moneyness), which helps identify systematic bias and heteroskedastic dispersion across contracts. These diagnostics complement Table 3 by showing where and how each architecture's sensitivity estimates deviate from the analytical benchmark.

4.4. OOD robustness

In interest rate markets, surrogate pricers are most valuable precisely when uncertainty rises, yet models are typically trained and tuned under relatively stable conditions. We therefore examined how each method's pricing accuracy changes when moving from a normal volatility regime to a high-volatility stress regime. Concretely, we retrained the surrogates using only low-volatility samples, while keeping the yield curve sampling and all other parameter ranges unchanged. For the Hull–White model, we trained it on samples with $\sigma \in [0.01, 0.10]$ and evaluated the OOD performance on $\sigma \in (0.10, 0.15]$. For the G2++ model, we trained it on $(\sigma, \eta) \in [0.01, 0.07] \times [0.01, 0.07]$ and evaluated the OOD performance on the higher volatility regime $(\sigma, \eta) \in (0.07, 0.10] \times (0.07, 0.10]$. Robustness is summarized by the change in performance $\Delta R^2 = R^2_{OOD} - R^2_{ID}$. Values close to zero indicate that performance is largely preserved under volatility stress. Large negative values indicate pronounced degradation in crisis-like regimes. The corresponding ID and OOD test set R^2 values, together with the deterioration metric ΔR^2, are summarized in Table 4.

Table 4. OOD volatility stress test results. Each model is trained in a low-volatility regime and evaluated on an ID and an OOD high-volatility test set. We report R^2_{ID}, R^2_{OOD}, and the change in performance $\Delta R^2 = R^2_{OOD} - R^2_{ID}$, where values of ΔR^2 close to zero indicate stronger robustness.

Model	Architecture	R^2_{ID}	R^2_{OOD}	ΔR^2
Hull–White	DeepONet	0.9861	0.9721	-0.0140
	PINN	0.3096	-0.2991	-0.6087
	DeepBSDE	0.8030	0.7577	-0.0454
G2++	DeepONet	0.9715	0.9561	-0.0155
	PINN	0.4299	0.2034	-0.2264
	DeepBSDE	0.8259	0.6874	-0.1385

As shown in Table 4, across both interest rate models, DeepONet showed the smallest deterioration in pricing accuracy under the OOD volatility stress test. This indicates that, even when trained on a restricted volatility range, the supervised operator learner captures a pricing map that extrapolates smoothly in the volatility direction. One structural driver is the training signal available to DeepONet. It is trained directly on supervised price labels, with volatility entering explicitly as an input. This tends to promote a globally coherent parametric dependence that remains stable under extrapolated volatility.

By contrast, the theory-informed models display substantially weaker OOD stability, with the deterioration being most severe for PINN. In the Hull–White stress test, PINN exhibited a substantial loss of pricing accuracy, with $R^2_{OOD} = -0.2991$ and $\Delta R^2 = -0.6087$, indicating performance worse than a constant mean predictor in the stressed regime. This sharp collapse is consistent with the structure of the PDE residual in the low-volatility training regime. Since the diffusion contribution in the Hull–White residual is scaled by σ^2, when training is restricted to $\sigma \in [0.01, 0.10]$, second-derivative information may be comparatively weakly weighted. Under OOD evaluation at a larger σ, the diffusion term becomes more influential and can amplify small curvature inaccuracies into large pricing errors. Moreover, higher volatility broadens the effective state region visited by the short-rate

dynamics. As a result, the OOD test can simultaneously induce a shift in both the volatility and state coverage. This is particularly challenging for collocation based PDE enforcement. More broadly, because PINN relies on local PDE-residual enforcement involving second-order derivatives, regime shifts can exacerbate numerical stiffness and optimization fragility in the learned solution family.

DeepBSDE is noticeably more stable than PINN but still exhibits a non-trivial deterioration under volatility stress, with ΔR^2 remaining meaningfully negative, especially in the two-factor G2++ setting. In G2++, both theory-informed methods degrade as the volatility increases, with DeepBSDE showing a moderate negative change and PINN showing a larger negative change. Overall, DeepBSDE consistently outperforms PINN in OOD robustness, but neither matches the stability of DeepONet in this experiment. This relative stability aligns with DeepBSDE's learning mechanism. DeepBSDE avoids explicit second-derivative residual minimization and is anchored by a global terminal payoff constraint propagated along the simulated paths. However, its pathwise training signal remains subject to Monte Carlo and time discretization noise, and these effects become more consequential in the higher-dimensional two-factor setting, leading to a larger deterioration under stress.

4.5. Computational efficiency and practical considerations

While Sections 4.2–4.4 focused on price accuracy, Vegas, and robustness under volatility stress, practical deployment of surrogate pricers also depends on computational efficiency. We therefore complement the quality metrics with a simple wall-clock timing comparison. Table 5 reports the training time measured until early stopping, together with the corresponding stopping epoch and test time wall-clock evaluation of the entire test set on a single NVIDIA RTX 3090 graphics processing unit (GPU).

Table 5. Wall-clock computational timing results for DeepONet, PINN, and DeepBSDE under the Hull–White and G2++ interest rate models. We report the early stopping epoch, the training time (min) measured until early stopping, and the test time (s) measured as the wall-clock time required to evaluate the entire test set once using the trained surrogate. All timings are measured on a single NVIDIA RTX 3090 GPU.

Model	Architecture	Early stopping epoch	Training time (min)	Test time (s)
Hull–White	DeepONet	185	6.58	0.129
	PINN	272	77.85	0.073
	DeepBSDE	239	34.87	0.110
G2++	DeepONet	330	11.65	0.130
	PINN	121	54.60	0.079
	DeepBSDE	137	82.33	0.140

Two patterns emerge. First, DeepONet is substantially faster to train with both interest rate models. In the Hull–White setting, DeepONet reaches early stopping in 6.58 minutes, compared with 77.85 minutes for PINN and 34.87 minutes for DeepBSDE. In the more complex two-factor G2++ setting, DeepONet still trains in 11.65 minutes, whereas PINN and DeepBSDE require 54.60 minutes and 82.33 minutes, respectively. Notably, this training time advantage is not simply explained by fewer

optimization epochs: for example, DeepONet can require more epochs than PINN in G2++, yet it remains faster in terms of wall-clock time, indicating a markedly lower per-epoch computational cost under our implementation.

This behavior is consistent with the distinct training objectives. DeepONet is trained via a supervised regression loss on option prices, involving standard forward and backward passes with first-order gradients. By contrast, PINN training enforces a PDE residual that contains second-order derivatives of the price with respect to the state variables, which substantially increases the automatic differentiation overhead and can yield a stiffer optimization problem over a broad state–parameter domain. DeepBSDE avoids explicit second-derivative residual minimization, but each training step requires Monte Carlo simulation of the factor paths and time discretization of the backward system, which increases the per-epoch cost and introduces pathwise noise that can slow optimization in practice.

Second, the test time evaluation is fast and comparable across methods: Evaluating the full test set requires less than 0.15 seconds in all cases reported in Table 5. This reflects that test time pricing only requires a forward evaluation of the trained surrogate, whereas the additional computational burdens specific to theory-informed training—PDE residual differentiation for PINN and path simulation/backward recursion for DeepBSDE—are not incurred during forward evaluation. Overall, these timing results complement the accuracy and robustness comparisons in Tables 2–4 by showing that, under our experimental setup, DeepONet combines strong predictive performance with a markedly lower training time, while all three surrogates provide rapid batch evaluation once trained. Peak GPU memory usage during training remained below 1.0 GB for all methods on an RTX 3090. Hence, memory was not a limiting factor during our experiments.

Method-specific limitations and practical considerations. The abovementioned experiments also highlight the limitations of each paradigm. DeepONet relies on supervised labels, so its applicability depends on the availability and cost of generating reliable reference prices, and its generalization is inherently tied to the training distribution over yield curve shapes and parameter ranges. PINN provides a label-free alternative, but PDE residual training requires repeated higher-order differentiation and can be sensitive to collocation design and dimensionality, which may yield a stiff optimization landscape and reduced stability in multi-factor settings. DeepBSDE avoids explicit second-order derivatives and tends to be more stable than PINN in our volatility stress tests, yet its training cost and stability depend on the number of simulated paths and the time discretization, and pathwise variance/discretization effects can become more pronounced as the dimensionality increases. These considerations should be weighed alongside price accuracy, Vega accuracy, robustness, and timing results when selecting a surrogate for a given fixed-income task.

5. Conclusions

Starting from observed U.S. Treasury term structures, we constructed yield curve inputs and paired them with broad contract/model parameter configurations to form a common evaluation dataset for European bond call options under the Hull–White and G2++ Gaussian affine term structure models. Within this unified pipeline, we trained three paradigms—supervised DeepONet operator learning, PDE-residual learning via the PINN approach, and BSDE-based learning via DeepBSDE. We evaluated

them using criteria that reflect practical use: Out-of-sample pricing accuracy, Vega accuracy obtained by automatic differentiation and validated against analytical sensitivities, and pricing robustness under an explicit OOD volatility stress test.

Across this controlled benchmark, DeepONet achieved the strongest overall performance for pricing accuracy, Vega accuracy, and OOD robustness on the held-out evaluations. In particular, the supervised operator learning approach delivered high test set pricing accuracy and comparatively reliable autodiff-based Vegas, exhibiting the smallest decrease in pricing accuracy under the volatility-only OOD stress test. At the same time, our experiments illustrate that a strong price level fit does not automatically translate into stable autodiff-based Greeks or robust extrapolation under stress, and that these gaps become more pronounced for theory-informed surrogates as the model's complexity increases. These findings support an objective driven view of validation, in which surrogate choice should be guided by the intended role—pricing, risk measurement, or robustness under distribution shift—rather than by price accuracy alone.

From an applied perspective, these findings suggest a simple selection rule. When accurate reference prices can be generated at scale, supervised operator learning with DeepONet provides a practical default in our setting. When such supervision is limited or costly, theory-informed methods remain viable label-free alternatives, but they should be validated more carefully for sensitivity-based tasks and stress behavior. In our multi-factor setting, DeepBSDE is generally more stable than PINN, while PINN can deteriorate substantially under volatility shifts.

These conclusions should be interpreted within the scope of a deliberately controlled benchmark. We focus on European bond call options under Gaussian affine term structure models, where reliable reference prices and Greeks are available in closed form and thus provide effective supervision and clean diagnostics. In particular, our robustness study considers a controlled volatility regime shift; extending the same evaluation to other forms of distribution shift—such as changes in the yield curve shape, shifts in contract distributions, or regime changes in mean-reversion/correlation parameters—remains an important direction for future work.

An additional next step is to extend the same task-driven evaluation to products and models without analytical pricing formulas. In such settings, the reference prices and sensitivities must be obtained from numerical procedures—most notably through Monte Carlo simulation and related variance-reduced estimators—so that the central question becomes the trade-off between surrogate quality and label-generation cost. In parallel, while Greek- or hedge-aware training objectives have been explored in parts of the BSDE-based literature, our findings suggest a clear direction specifically for DeepONet in term structure settings: Scalability should be judged not only by the price level's fit but also by the risk sensitivity's fidelity. In particular, augmenting DeepONet training with Greek-aware objectives—such as gradient matching or multi task learning of prices and sensitivities—appears promising when closed-form sensitivities are unavailable or when autodiff-based Greeks are unstable. Together, these extensions would broaden the applicability of the proposed benchmarking lens and clarify when, and to what extent, theory-informed constraints can compensate for limited or costly supervision in realistic fixed-income pricing problems.

Author contributions

Sanghyun Lee: Data curation, formal analysis, software, visualization, writing – original draft; Jeonggyu Huh: Conceptualization, supervision, validation, writing – review and editing, funding acquisition; Seungwon Jeong: Formal analysis, software, validation, writing – review and editing. All authors have read and approved the final version of the manuscript for publication.

Use of Generative-AI tools declaration

The authors declare they have not used Artificial Intelligence (AI) tools in the creation of this article.

Acknowledgments

Jeonggyu Huh received financial support from the National Research Foundation of Korea (No. RS-2025-00562904). Seungwon Jeong was supported by Global-Learning & Academic Research institution for Master's Ph·D students, and Postdocs (LAMP) Program of the National Research Foundation of Korea (NRF) grant funded by the Ministry of Education (No. RS-2024-00442775).

Conflict of interest

All authors declare no conflicts of interest in this paper.

References

1. R. Assabumrungrat, K. Minami, M. Hirano, *Error analysis of option pricing via deep PDE solvers: Empirical study*, In 2024 16th IIAI International Congress on Advanced Applied Informatics (IIAI-AAI), 329–336. IEEE, 2024. http://dx.doi.org/10.1109/IIAI-AAI63651.2024.00068
2. E. Bayraktar, Q. Feng, Z. Zhang, Z. Zhang, Deep neural operator learning for probabilistic models, *arXiv preprint*, 2025. http://dx.doi.org/10.48550/arXiv.2511.07235
3. D. Brigo, F. Mercurio, *Interest rate models—Theory and practice: With smile, inflation and credit*, Springer, 2006. http://dx.doi.org/10.1007/978-3-540-34604-3
4. T. Chen, H. Chen, Universal approximation to nonlinear operators by neural networks with arbitrary activation functions and its application to dynamical systems, *IEEE T. Neural Networ.*, **6** (1995), 911–917. http://dx.doi.org/10.1109/72.392253
5. Y. Chen, X. Lü, H. Tian, R. H. Li, Physics-informed neural network for barrier option pricing in coupled financial quantitative system with varying interest rate and volatility, *Eng. Anal. Bound. Elem.*, **180** (2025), 106457. http://dx.doi.org/10.1016/j.enganabound.2025.106457
6. Y. Cui, L. Li, G. Zhang, W. Zhang, Learning pricing maps for financial derivatives with deep operator networks, Available at SSRN 4670716, 2023. http://dx.doi.org/10.2139/ssrn.4670716
7. R. Culkin, S. R. Das, Machine learning in finance: The case of deep learning for option pricing, *J. Invest. Manag.*, **15** (2017), 92–100.

8. B. Deng, Y. Shin, L. Lu, Z. Zhang, G. E. Karniadakis, Approximation rates of DeepONets for learning operators arising from advection-diffusion equations, *Neural Networks*, **153** (2022), 411–426. http://dx.doi.org/10.1016/j.neunet.2022.06.019
9. A. Dhiman, Y. Hu, Physics informed neural network for option pricing, *arXiv preprint*, 2023. http://dx.doi.org/10.48550/arXiv.2312.06711
10. W. E, J. Han, A. Jentzen, Deep learning-based numerical methods for high-dimensional parabolic partial differential equations and backward stochastic differential equations, *Commun. Math. Stat.*, **5** (2017), 349–380. http://dx.doi.org/10.1007/s40304-017-0117-6
11. R. Ferguson, A. Green, Deeply learning derivatives, *arXiv preprint*, 2018. http://dx.doi.org/10.48550/arXiv.1809.02233
12. N. Ganesan, Y. Yu, B. Hientzsch, Pricing barrier options with deepBSDEs, *arXiv preprint*, 2020. http://dx.doi.org/10.48550/arXiv.2005.10966
13. M. Germain, H. Pham, X. Warin, Approximation error analysis of some deep backward schemes for nonlinear PDEs, *SIAM J. Sci. Comput.*, **44** (2022), A28–A56. http://dx.doi.org/10.1137/20M1355355
14. S. Ghadimi, G. Lan, Stochastic first-and zeroth-order methods for nonconvex stochastic programming, *SIAM J. Optimization*, **23** (2013), 2341–2368. http://dx.doi.org/10.1137/120880811
15. K. Glau, L. Wunderlich, The deep parametric PDE method and applications to option pricing, *Appl. Math. Comput.*, **432** (2022), 127355. http://dx.doi.org/10.1016/j.amc.2022.127355
16. D. Hainaut, A. Casas, Option pricing in the Heston model with physics inspired neural networks, *Ann. Financ.*, **20** (2024), 353–376. http://dx.doi.org/10.1007/s10436-024-00452-7
17. J. Han, A. Jentzen, W. E, Solving high-dimensional partial differential equations using deep learning, *P. Natl. Acad. Sci.*, **115** (2018), 8505–8510. http://dx.doi.org/10.1073/pnas.1718942115
18. A. Hirsa, T. Karatas, A. Oskoui, Supervised deep neural networks (DNNs) for pricing/calibration of vanilla/exotic options under various different processes, *arXiv preprint*, 2019. http://dx.doi.org/10.48550/arXiv.1902.05810
19. J. Hull, A. White, Pricing interest-rate-derivative securities, *Rev. Financ. Stud.*, **3** (1990), 573–592. http://dx.doi.org/10.1093/rfs/3.4.573
20. J. C. Hull, S. Basu, *Options, futures, and other derivatives*, Pearson Education India, 2016.
21. C. Huré, H. Pham, X. Warin, Deep backward schemes for high-dimensional nonlinear PDEs, *Math. Comput.*, **89** (2020), 1547–1579. http://dx.doi.org/10.1090/mcom/3514
22. J. M. Hutchinson, A. W. Lo, T. Poggio, A nonparametric approach to pricing and hedging derivative securities via learning networks, *J. Financ.*, **49** (1994), 851–889. http://dx.doi.org/10.1111/j.1540-6261.1994.tb00081.x
23. J. Kienitz, S. K. Acar, Q. Liang, N. Nowaczyk, Deep option pricing-term structure models, Available at SSRN 3498398, 2019. http://dx.doi.org/10.2139/ssrn.3498398
24. S. Lanthaler, S. Mishra, G. E. Karniadakis, Error estimates for deeponets: A deep learning framework in infinite dimensions, *T. Math. Appl.*, **6** (2022), tnac001. http://dx.doi.org/10.1093/imatrm/tnac001

25. W. Lefebvre, G. Loeper, H. Pham, Differential learning methods for solving fully nonlinear PDEs, *Digit. Financ.*, **5** (2023), 183–229. http://dx.doi.org/10.1007/s42521-023-00077-x

26. R. B. Litterman, J. Scheinkman, Common factors affecting bond returns, *J. Fix. Income Summer*, **1** (1991), 54–61. http://dx.doi.org/10.3905/jfi.1991.692347

27. L. Lu, P. Jin, G. Pang, Z. Zhang, G. E. Karniadakis, Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators, *Nat. Mach. Intell.*, **3** (2021), 218–229. http://dx.doi.org/10.1038/s42256-021-00302-5

28. K. A. Malandain, S. Kalici, H. Chakhoyan, DeepSVM: Learning stochastic volatility models with physics-informed deep operator networks, *arXiv preprint*, 2025. http://dx.doi.org/10.48550/arXiv.2512.07162

29. M. Malliaris, L. Salchenberger, A neural network model for estimating option prices, *Appl. Intell.*, **3** (1993), 193–206. http://dx.doi.org/10.1007/BF00871937

30. B. Negyesi, C. W. Oosterlee, A deep BSDE approach for the simultaneous pricing and delta-gamma hedging of large portfolios consisting of high-dimensional multi-asset Bermudan options, *arXiv preprint*, 2025. http://dx.doi.org/10.48550/arXiv.2502.11706

31. N. K. Pande, P. Pasricha, A. Kumar, A. K. Gupta, European option pricing in regime switching framework via physics-informed residual learning, *Expert Syst. Appl.*, 2025. http://dx.doi.org/10.1016/j.eswa.2025.128226

32. E. Pardoux, S. Peng, Adapted solution of a backward stochastic differential equation, *Syst. Control Lett.*, **14** (1990), 55–61. http://dx.doi.org/10.1016/0167-6911(90)90082-6

33. Y. Qiu, N. Bridges, P. Chen, Derivative-enhanced deep operator network, *Adv. Neur. Inform. Process. Syst.*, **37** (2024), 20945–20981. https://doi.org/10.52202/079017-0660

34. 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, *J. Comput. Phys.*, **378** (2019), 686–707. http://dx.doi.org/10.1016/j.jcp.2018.10.045

35. I. Rani, C. K. Verma, G-pinns: A Bayesian-optimized gru-enhanced physics-informed neural network for advancing short-rate model predictions, *Eng. Anal. Bound. Elem.*, **179** (2025), 106396. http://dx.doi.org/10.1016/j.enganabound.2025.106396

36. S. Shalev-Shwartz, S. Ben-David, *Understanding machine learning: From theory to algorithms*, Cambridge University Press, 2014. http://dx.doi.org/10.1017/CBO9781107298019

37. S. E. Shreve, *Stochastic calculus for finance II: Continuous-time models*, Springer, **11** (2004). http://dx.doi.org/10.1007/978-1-4757-4296-1

38. H. Wang, H. Chen, A. Sudjianto, R. Liu, Q. Shen, Deep learning-based BSDE solver for LIBOR market model with application to Bermudan swaption pricing and hedging, *arXiv preprint*, 2018. http://dx.doi.org/10.48550/arXiv.1807.06622

39. X. Wang, J. Li, J. Li, A deep learning based numerical PDE method for option pricing, *Comput. Econ.*, **62** (2023), 149–164. http://dx.doi.org/10.1007/s10614-022-10279-x

40. B. Yu, X. Xing, A. Sudjianto, Deep-learning based numerical BSDE method for barrier options, *arXiv preprint*, 2019. http://dx.doi.org/10.48550/arXiv.1904.05921

Appendix

A. Detailed analytical formulas

This appendix provides the explicit closed-form solutions for zero coupon bond prices and European call option prices under the Hull–White and G2++ models.

A.1. Hull–White one-factor model

Recall that, under the risk-neutral measure $\mathbb{Q}$, the one-factor Hull–White short-rate follows the OU dynamics

$$dr_t = (\theta(t) - ar(t))dt + \sigma dW_t,$$

where $\theta(t)$ is chosen to match the initial term structure via the initial instantaneous forward rate curve. Throughout this paper, we adopt the convention

$$f(0,t) := -\frac{\partial}{\partial t}\log P(0,t).$$

Under this notation, the drift calibration function $\theta(t)$ admits the standard expression

$$\theta(t) = \frac{\partial f(0,t)}{\partial t} + af(0,t) + \frac{\sigma^2}{2a}\left(1 - e^{-2at}\right).$$

A key consequence of the Gaussian affine structure is that the time-t price of a zero coupon bond maturing at T_B can be written in exponential–affine form as follows:

$$P(t,T_B) = A(t,T_B)\exp\left(-B(t,T_B)r_t\right),$$

where the loading $B(t,T_B)$ is given by

$$B(t,T_B) = \frac{1 - e^{-a(T_B-t)}}{a}.$$

The prefactor $A(t,T_B)$ is determined by the initial discount curve and the model parameters. In logarithmic form, one convenient representation is

$$\log A(t,T_B) = \log\frac{P(0,T_B)}{P(0,t)} + B(t,T_B)f(0,t) - \frac{\sigma^2}{4a}\left(1 - e^{-2at}\right)B(t,T_B)^2.$$

We next record the closed-form price of a European call option on a zero coupon bond. Let $0 \le t \le T < T_B$, and consider an option with the expiry T, written on the T_B-maturity bond, with a strike $K > 0$. Its time-t price in the Hull–White model is given by the Black–Scholes-type formula

$$C(t;T,T_B,K) = P(t,T_B)\mathcal{N}(h) - KP(t,T)\mathcal{N}(h - \sigma_P),$$

where $\mathcal{N}(\cdot)$ denotes the standard normal cumulative distribution function and

$$h = \frac{1}{\sigma_P}\log\frac{P(t,T_B)}{KP(t,T)} + \frac{\sigma_P}{2}.$$

The effective bond volatility σ_P is

$$\sigma_P^2 = \frac{\sigma^2}{2a}\left(1 - e^{-2a(T-t)}\right)B(T,T_B)^2.$$

A.2. G2++ model formulas

We first recall the analytical ingredients of the G2++ model used in our experiments. The short-rate is represented as

$$r_t = \xi_t + \nu_t + \phi(t),$$

where (ξ_t, ν_t) are two correlated OU factors satisfying

$$d\xi_t = -a\xi_t dt + \sigma dW_t^1, \quad d\nu_t = -b\nu_t dt + \eta dW_t^2,$$

subject to $dW_t^1 dW_t^2 = \rho dt$. The deterministic shift $\phi(t)$ is chosen so that the model reproduces the initial term structure. A closed-form calibration $\phi(t)$ is given by

$$\phi(t) = f(0,t) + \frac{\sigma^2}{2a^2}\left(1 - e^{-at}\right)^2 + \frac{\eta^2}{2b^2}\left(1 - e^{-bt}\right)^2 + \frac{\rho\sigma\eta}{ab}\left(1 - e^{-at}\right)\left(1 - e^{-bt}\right).$$

The corresponding zero coupon bond price admits an exponential–affine representation in the state variables:

$$P(t, T_B) = \exp\left(A(t, T_B) - B_a(t, T_B)\xi_t - B_b(t, T_B)\nu_t\right)$$

where the factor loadings are given by

$$B_a(t, T_B) = \frac{1 - e^{-a(T_B - t)}}{a}, \quad B_b(t, T_B) = \frac{1 - e^{-b(T_B - t)}}{b}.$$

Here, $A(t, T_B)$ is a deterministic function ensuring consistency with the initial discount curve; an explicit expression is standard and can be found in the G2++ reference formulas [3].

We finally record the closed-form price of a European call option with the expiry T written on the zero coupon bond maturing at $T < T_B$ with a strike K:

$$C(t; T, T_B, K) = P(t, T_B)\mathcal{N}(h_1) - KP(t, T)\mathcal{N}(h_2),$$

where

$$h_1 = \frac{1}{\Sigma}\log\frac{P(t, T_B)}{KP(t, T)} + \frac{\Sigma}{2}, \quad h_2 = h_1 - \Sigma$$

and the effective bond volatility Σ is

$$\Sigma^2 = \frac{\sigma^2}{2a}\left(1 - e^{-2a(T-t)}\right)B_a(T, T_B)^2 + \frac{\eta^2}{2b}\left(1 - e^{-2b(T-t)}\right)B_b(T, T_B)^2 + \frac{2\rho\sigma\eta}{a+b}\left(1 - e^{-(a+b)(T-t)}\right)B_a(T, T_B)B_b(T, T_B).$$

B. Convergence analysis of the DeepONet

In this appendix, we first quantify the approximation error of the DeepONet estimator. Throughout, we work on the bounded domain $U_0 \times Y$ and keep the notation of Section 3.1: G denotes the true pricing operator, G_θ is a DeepONet with the parameters θ, and $R(\theta)$ is the corresponding population risk.

B.1. Approximation error

The DeepONet used in this work factors the operator approximation into three distinct stages. We introduce an encoder

$$\mathcal{E} : U_0 \to \mathbb{R}^m, \quad \mathcal{E} = (u(x_1), \cdots, u(x_m)),$$

which maps a yield curve u to its values at m fixed maturity sensors $\{x_j\}_{j=1}^m$. The functional information lost by this discretization is compensated by a deterministic decoder $\mathfrak{D} : \mathbb{R}^m \to U_0$, for instance, given by piecewise–polynomial interpolation on the maturity grid.

To control the approximation error of DeepONet, we impose the following mild regularity and expressivity assumptions on the pricing operator and on the DeepONet hypothesis class.

Assumption 2. *(a) The term structure space contains a bounded subset U_0 and the pricing configurations lie in a bounded set $Y \subset \mathbb{R}^d$ such that the distribution μ is supported on $U_0 \times Y$. The map*

$$G : \mathcal{U}_0 \to L^2(Y), \quad u \mapsto (y \mapsto G(u)(y))$$

is Lipschitz continuous,

$$\|G(u_1) - G(u_2)\|_{L^2(Y)} \le L_G \|u_1 - u_2\|_{U_0},$$

and for each $u \in U_0$, the function $y \mapsto G(u)(y)$ is Hölder-continuous of order $s > 0$ on Y,

$$\|G(u)(y_1) - G(u)(y_2)\| \le L_y \|y_1 - y_2\|^s.$$

(b) The DeepONet G_θ uses m sensors on the maturity interval, a branch network of width N_b, and a trunk network of width N_t. We write N for a generic measure of network size, e.g., $N = m+N_b+N_t$. The parameter set Θ consists of all weight configurations of such DeepONet architectures, and the associated hypothesis class

$$\mathcal{H}_N \equiv \{G_\theta : \theta \in \Theta,\ size(G_\theta) \le N\}$$

is rich enough to approximate Lipschitz–Hölder operators on $U_0 \times Y$ in the sense of the operator–approximation results [4, 8, 24, 27].

Under Assumption 2, we may regard G and the DeepONet family as Lipschitz–Hölder operators on the bounded domain $U_0 \times Y$. We next specify the representation of the output space implemented by the trunk network. To describe the reconstruction step on the output side, we fix a finite family of basis functions $\{\varphi_k\}_{k=1}^p \subset L^2(Y)$ and write

$$P : L^2(Y) \to \mathbb{R}^p, \quad P(v) = \Big(\langle v, \varphi_1\rangle, \cdots, \langle v, \varphi_p\rangle\Big),$$

for the coefficient projection, and

$$\mathcal{R}_\theta : \mathbb{R}^p \to L^2(Y), \quad \mathcal{R}_\theta(b)(y) = \sum_{k=1}^p b_k \varphi_k(y),$$

for the corresponding reconstruction operator implemented by the trunk network.

The learnable part of DeepONet is the finite-dimensional approximator

$$\mathcal{A}_\theta : \mathbb{R}^m \to \mathbb{R}^p,$$

realized by the branch network. With this notation, the DeepONet can be written as the composition

$$G_\theta = \mathcal{R}_\theta \circ \mathcal{A}_\theta \circ \mathcal{E}.$$

We then define the encoding, finite-dimensional approximation, and reconstruction errors as

$$\begin{aligned}
\mathcal{E}_{\text{enc}} &:= \sup_{u \in U_0} \|\mathcal{D} \circ \mathcal{E}(u) - u\|_{U_0}, \\
\mathcal{E}_{\text{fd}} &:= \sup_{u \in U_0} \|\mathcal{A}_\theta(\mathcal{E}(u)) - P(G(u))\|_{\mathbb{R}^p}, \\
\mathcal{E}_{\text{rec}} &:= \sup_{u \in G(U_0)} \|\mathcal{R}_\theta(P(v)) - v\|_{L^2(Y)}.
\end{aligned}$$

The next lemma shows that these three quantities control the approximation error of DeepONet.

Lemma 3. *Assume that the pricing operator $G : U_0 \to L^2(Y)$ is Lipschitz-continuous on U_0 as described in Assumption 2, and that the reconstruction map $\mathcal{R}_\theta$ is Lipschitz on $\mathbb{R}^p$*

$$\|\mathcal{R}_\theta(a) - \mathcal{R}_\theta(b)\|_{L^2(Y)} \le L_R \|a - b\|_{\mathbb{R}^p}.$$

Then, a constant $C > 0$ exists, independent of θ, such that

$$\sup_{u \in U_0} \|G_\theta(u) - G(u)\|_{L^2(Y)} \le C\left(\mathcal{E}_{enc} + \mathcal{E}_{fd}(\theta) + \mathcal{E}_{rec}\right).$$

Proof. For any $u \in U_0$, we insert the intermediate quantities

$$G(u), \quad G(\mathcal{D} \circ \mathcal{E}(u)), \quad \mathcal{R}_\theta(P(G(\mathcal{D} \circ \mathcal{E}(u)))).$$

For simplicity, let us denote $\tilde{u}$ as $\mathcal{D}(\mathcal{E}(u)) \in U_0$ and apply the triangle inequality:

$$\begin{aligned}
\|G_\theta(u) - G(u)\|_{L^2(Y)} \le{}& \|G_\theta(u) - \mathcal{R}_\theta(P(G(\tilde{u})))\|_{L^2(Y)} \\
&+ \|\mathcal{R}_\theta(P(G(\tilde{u}))) - G(\tilde{u})\|_{L^2(Y)} + \|G(\tilde{u}) - G(u)\|_{L^2(Y)}.
\end{aligned}$$

On the right-hand side of the inequality above, the first and third terms are bounded by the Lipschitz assumptions, and the second term is equal to the definition of the reconstruction error $\mathcal{E}_{\text{rec}}$. We therefore get

$$\|G_\theta(u) - G(u)\|_{L^2(Y)} \le L_R \mathcal{E}_{\text{fd}}(\theta) + \mathcal{E}_{\text{rec}} + L_G \mathcal{E}_{\text{enc}},$$

and take a supremum for $u \in U_0$

$$\sup_{u \in U_0} \|G_\theta(u) - G(u)\|_{L^2(Y)} \le C\left(\mathcal{E}_{\text{enc}} + \mathcal{E}_{\text{fd}}(\theta) + \mathcal{E}_{\text{rec}}\right),$$

for constant $C > 0$. □

We now combine the decomposition of Lemma 3 with the existing approximation results for the encoder, branch, and trunk networks to obtain a quantitative bound on the DeepONet approximation error.

Proposition 4. *Suppose that Assumption 2 on the regularity of the pricing operator $G : U_0 \to L^2(Y)$ holds. Then, the exponents $r, \rho, \gamma > 0$ and a constant $C > 0$ exist such that, for each integer $N \geq 1$, one can choose the number of sensors $m = m(N)$, the number of output modes $p = p(N)$, and a DeepONet parameter vector θ_N with an effective size of at most N for which*

$$\sup_{u \in U_0} \|G_\theta(u) - G(u)\|_{L^2(Y)} \leq C \left(m(N)^{-r} + p(N)^{-\rho} + N^{-\gamma} \right).$$

In particular, $\alpha > 0$ and $C > 0$ exist such that

$$\sup_{u \in U_0} \|G_\theta(u) - G(u)\|_{L^2(Y)} \leq CN^{-\alpha}, \quad \varepsilon_{app} = \inf_\theta R(\theta) \leq CN^{-\alpha}.$$

Proof. Lemma 3 bounds the operator approximation error in terms of the three structural error terms $\mathcal{E}_{\text{enc}}, \mathcal{E}_{\text{fd}}$, and $\mathcal{E}_{\text{rec}}$. Under Assumption 2, the assumptions of previous work [24] are satisfied, and their results yield the following bounds.

(i) Encoding error: The encoder–decoder pair can be chosen so that

$$\mathcal{E}_{\text{enc}} \leq C_{\text{enc}} m^{-r}$$

for some $r > 0$, by Proposition 3.15 and Theorem 3.16 in [24].

(ii) Reconstruction error: Using a spectral basis and the corresponding trunk network, the reconstruction error satisfies

$$\mathcal{E}_{\text{rec}} \leq C_{\text{rec}} p^{-\rho}$$

for some $\rho > 0$, by Theorem 3.5 in [24].

(iii) Finite-dimensional approximation error: For a finite-dimensional map $z \mapsto P(G(D(z)))$, Theorem 3.21 in [24] provides a DeepONet branch network with an effective size of at most N with

$$\mathcal{E}_{\text{fd}} \leq C_{\text{fd}} N^{-\gamma}$$

for some $\gamma > 0$.

Substituting these bounds into the inequality of Lemma 3 gives

$$\sup_{u \in U_0} \|G_\theta(u) - G(u)\|_{L^2(Y)} \leq C \left(m(N)^{-r} + p(N)^{-\rho} + N^{-\gamma} \right).$$

If $m(N)$ and $p(N)$ to grow at least linearly in N and setting $\alpha := \min\{r, \rho, \gamma\}$ yields

$$\sup_{u \in U_0} \|G_\theta(u) - G(u)\|_{L^2(Y)} \leq CN^{-\frac{\alpha}{2}}.$$

Since the population risk satisfies

$$R(\theta_N) = \mathbb{E}_{(u,y)\sim\mu} \left[\left(G_{\theta_N}(u)(y) - G(u)(y) \right)^2 \right] \leq \left(\sup_{u \in U_0} \|G_\theta(u) - G(u)\|_{L^2(Y)} \right)^2,$$

we obtain $\varepsilon_{\text{app}} := \inf_\theta R(\theta) \leq R(\theta_N) \leq CN^{-\alpha}$, as claimed. □

B.2. Optimization error for SGD

We next quantify the optimization error incurred when the empirical risk R_n is minimized by SGD. Recall that $R_n(\theta)$ denotes the empirical counterpart of the population risk $R(\theta)$ and $\varepsilon_{\text{opt}}(K) := R_n(\theta_K) - \inf_\theta R_n(\theta)$, as defined in Section 3.1, where θ_K denotes the parameter vector produced after K steps of SGD applied to the empirical risk. Throughout this subsection, we write $\nabla R_n(\theta)$ for the gradient of R_n with respect to the parameter vector θ.

Assumption 5. *(a) The empirical risk $R_n : \mathbb{R}^{d_\theta} \to \mathbb{R}$ is differentiable with L-Lipschitz continuous gradient on Θ as follows:*

$$\|\nabla R_n(\theta_1) - \nabla R_n(\theta_2)\| \le L\,\|\theta_1 - \theta_2\| \quad \text{for all } \theta_1, \theta_2 \in \Theta.$$

At iteration k, SGD generates an update

$$\theta_{k+1} = \theta_k - \eta_k g_k,$$

where g_k is a stochastic gradient computed from a mini batch and the stepsize $\eta_k > 0$.

(b) The stochastic gradients are unbiased with bounded second moments

$$\mathbb{E}\left[g_k|\theta_k\right] = \nabla_\theta R_n(\theta_k), \quad \mathbb{E}\left[\|g_k\|^2\,|\theta_k\right] \le G^2_{grad}$$

for all $k \ge 0$.

(c) The stepsizes $\{\eta_k\}_{k\ge 0}$ are positive and satisfy the classical Robbins–Monro conditions as follows:

$$\sum_{k=0}^{\infty} \eta_k = \infty, \quad \sum_{k=0}^{\infty} \eta_k^2 < \infty.$$

(d) The empirical risk is bounded from below on Θ, $R_n^* := \inf_\theta R_n(\theta) > -\infty$, and the level set $\{\theta : R_n(\theta) \le R_n(\theta_0)\}$ is bounded for the chosen initialization θ_0.

In addition, we impose a mild curvature condition which allows us to convert control of the gradient norm into control of the function values.

Assumption 6. *We assume that the empirical risk R_n satisfies a Polyak–Łojasiewicz type inequality. A constant $c_{PL} > 0$ exists such that*

$$R_n(\theta) - R_n^* \le \frac{\|\nabla R_n(\theta)\|^2}{2c_{PL}} \quad \text{for all } \theta \in \Theta.$$

Under these assumptions, we obtain the following rate for the optimization error.

Lemma 7. *Suppose that Assumptions 5 and 6 hold. Then a constant $C > 0$ exists, independent of K, such that the SGD iterate θ_K satisfies*

$$\mathbb{E}\left[R_n(\theta_K)\right] \le R_n^* + CK^{-1/2}.$$

In particular, the optimization error

$$\varepsilon_{opt}(K) := \mathbb{E}\left[R_n(\theta_K)\right] - R_n^*$$

decays at a rate $\varepsilon_{opt}(K) = O(K^{-1/2})$ as $K \to \infty$.

Proof. The result is a direct consequence of the convergence theory for SGD on smooth objectives. Under Assumption 5, the empirical risk is L-smooth and the stochastic gradients have bounded variance. Following Theorem 2.1 in the seminal work [14], for such problems, SGD with a suitable diminishing stepsize schedule satisfies

$$\frac{1}{K}\sum_{k=0}^{K-1}\mathbb{E}\left[\|\nabla R_n(\theta_k)\|^2\right] \le \frac{2L}{K}\left(R_n(\theta_0^*) - R_n^*\right) + C_1 K^{-1/2},$$

for constant $C_1 > 0$ depending only on L and the variance bound G_{grad}^2. Defining θ_K for the analysis as a randomly selected iterate among $\{\theta_0, \cdots, \theta_{K-1}\}$ with equal probability, the left-hand side is exactly $\|\nabla R_n(\theta_K)\|^2$. Hence, we have

$$\mathbb{E}\left[\|\nabla R_n(\theta_K)\|^2\right] \le C_2 K^{-1/2}$$

for some constant $C_2 > 0$. The curvature condition in Assumption 6 now yields

$$\mathbb{E}\left[R_n(\theta_K) - R_n^*\right] \le \frac{1}{2c_{PL}}\mathbb{E}\left[\|\nabla R_n(\theta_K)\|^2\right] \le CK^{-1/2},$$

for a constant $C > 0$ depending only on $L, G_{\text{grad}}, c_{PL}$, and the initial risk level $R_n(\theta_0)$. This gives the stated bound. □

B.3. Generalization error

We finally address the statistical error arising from replacing the population risk $R(\theta)$ with its empirical counterpart $R_n(\theta)$. Throughout this subsection, we fix the network architecture N and assume that the weights and biases are uniformly bounded

$$\Theta_N := \{\theta \in \mathbb{R}^{d_\theta} : \|\theta\|_\infty \le B\},$$

where d_θ denotes the number of trainable parameters and $B > 0$ is a fixed weight bound. Recall that we define the generalization error as

$$\varepsilon_{\text{gen}} := \sup_{\theta\in\Theta_N} |R(\theta) - R_n(\theta)|.$$

To obtain a quantitative bound, we adopt the following assumptions.

Assumption 8. *(a) The training sample $\{u_i, y_i, z_i\}_{i=1}^n$ is drawn independently and identically distributed (i.i.d.) from the data-generating distribution μ on $U_0 \times Y \times \mathbb{R}$, and*

$$z_i = G(u_i)(y_i), \quad i = 1, \cdots, n.$$

(b) A function $\Psi : U_0 \to [0, \infty)$ and the constants $C > 0$, $\kappa > 0$ exist such that, for all $u \in U_0, y \in Y, \theta \in \Theta_N$, we have

$$|G(u)(y)| \le \Psi(u), \quad |G_\theta(u)(y)| \le \Psi(u), \quad \Psi(u) \le C(1 + \|u\|_{U_0}^\kappa).$$

For the squared loss

$$l_\theta(u, y) := \left(G_\theta(u)(y) - G(u)(y)\right)^2,$$

this implies the uniform bound

$$0 \le l_\theta(u, y) \le 4\Psi(u)^2 \le C(1 + \|u\|_{U_0}^{2\kappa}),$$

and

$$|l_\theta(u, y) - l_{\theta'}(u, y)| \le 4\Psi(u)|G_\theta(u)(y) - G_{\theta'}(u)(y)|.$$

Hence, the squared loss is Lipschitz in the network output with the Lipschitz constant growing at most polynomially in $\|u\|_{U_0}$.

(c) A function $\Phi : U_0 \to [0, \infty)$ *and the constants* $C > 0$, $\kappa > 0$ *exist such that, for all* $u \in U_0, y \in Y$, *and* $\theta, \theta' \in \Theta_N$, *we have*

$$|G_\theta(u)(y) - G_{\theta'}(u)(y)| \le \Phi(u)\|\theta - \theta'\|_\infty, \quad \Phi(u) \le C(1 + \|u\|_{U_0}^{\kappa}).$$

Consequently

$$|l_\theta(u, y) - l_{\theta'}(u, y)| \le 4\Psi(u)\Phi(u)\|\theta - \theta'\|_\infty \le C(1 + \|u\|_{U_0}^{2\kappa})\|\theta - \theta'\|_\infty.$$

The dependence of C on the parameter dimension d_θ *and on a uniform weight bound B is at most polynomial and is absorbed into the constant.*

Under Assumption 8, the loss class

$$\mathcal{F}_N := \{l_\theta : \theta \in \Theta_N\}$$

is uniformly bounded and has finite capacity. In particular, its empirical Rademacher complexity is of order $n^{-1/2}$ for a fixed architecture N.

Lemma 9. *Suppose that Assumption 8 holds and fix the architecture size N. Then a constant* $C > 0$ *exists, independent of the sample size n, such that for all* $n \ge 1$, *we have*

$$\varepsilon_{gen}(n) = \sup_{\theta \in \Theta_N} |R(\theta) - R_n(\theta)| \le Cn^{-1/2}$$

with high probability. In particular, $\varepsilon_{gen}(n) = O\left(n^{-1/2}\right)$ *as* $n \to \infty$ *for any fixed architecture N.*

Proof. By Assumption 8, the loss class $\mathcal{F}_N$ is uniformly bounded and Lipschitz in both the network output and the parameters. Standard symmetrization and contraction arguments for Rademacher complexity yield

$$\sup_{\theta \in \Theta_N} |R(\theta) - R_n(\theta)| \le 2\mathfrak{R}_n(\mathcal{F}_N) + C_1 \sqrt{\frac{\log(1/\delta)}{n}}$$

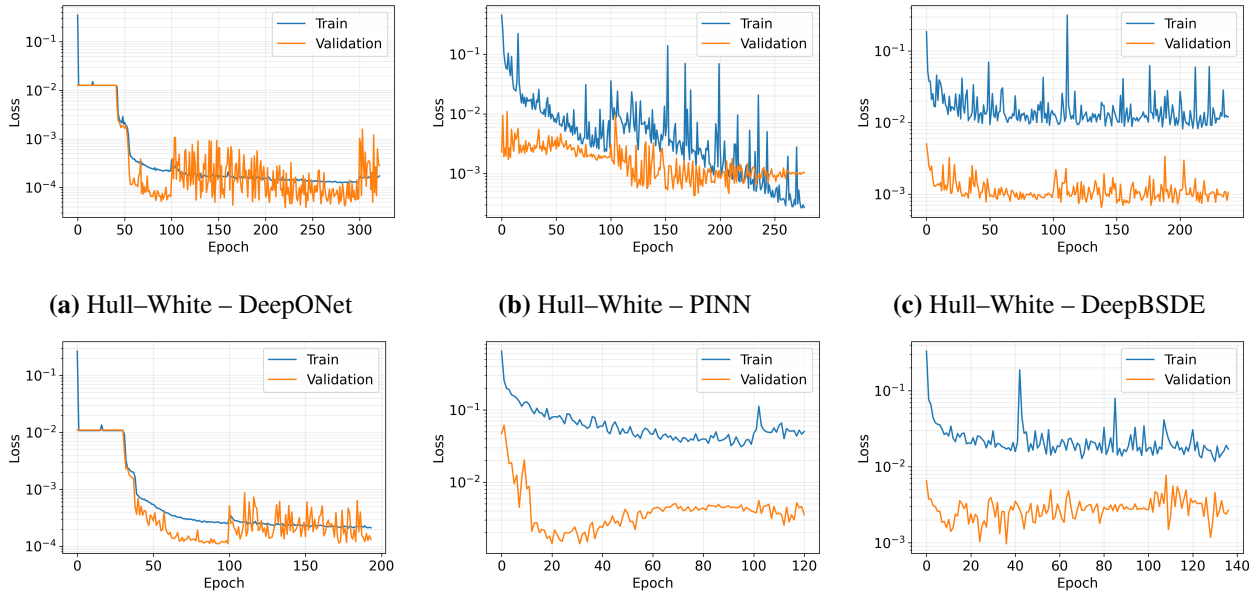
with a probability of at least $1 - \delta$, where $\mathfrak{R}_n(\mathcal{F}_N)$ denotes the empirical Rademacher complexity of $\mathcal{F}_N$ and $C_1 > 0$ depends only on the uniform bound in Assumption 8. For DeepONet architectures satisfying Assumption 8, $\mathfrak{R}_n(\mathcal{F}_N)$ has an upper bound,

$$\mathfrak{R}_n(\mathcal{F}_N) \le C_2 \sqrt{\frac{d_\theta}{n}},$$

for a constant $C_2 > 0$ depending on d_θ, B, and the polynomial degree κ but not on n by the result of [24]. Collecting the terms and absorbing the dependence on $d_\theta, B, \kappa, \delta$ into a single constant C yields the claimed bound $\varepsilon_{\text{gen}} \le Cn^{-1/2}$. □

C. Learning curves

To illustrate the convergence behavior and to complement the early stopping epochs and wall-clock training times reported in Table 5, Figure C.1 shows representative learning curves (log scale) for all method–model combinations. The training loss corresponds to each paradigm's training objective as defined in Section 3, whereas the validation loss is the prices' MSE against analytical reference prices and is used for model selection and early stopping as described in Section 4.1.3.



(a) Hull–White – DeepONet **(b)** Hull–White – PINN **(c)** Hull–White – DeepBSDE

(d) G2++ – DeepONet **(e)** G2++ – PINN **(f)** G2++ – DeepBSDE

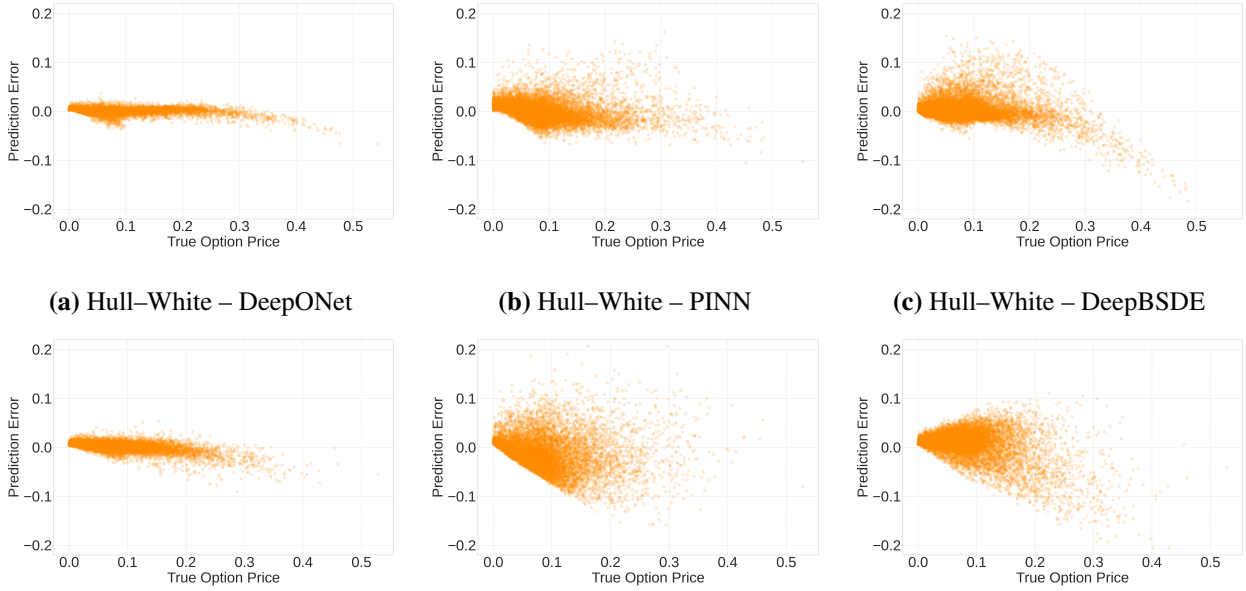
Figure C.1. Training and validation learning curves (log scale) for DeepONet, PINN, and DeepBSDE under the Hull–White and G2++ models. The training curve shows each method's training objective, and the validation curve shows the prices' MSE against the analytical reference prices used for early stopping.

As shown in Figure C.1, DeepONet exhibits a rapid decrease in validation error and reaches a stable low error regime relatively early, consistent with efficient supervised optimization. In contrast, PINN and DeepBSDE show noisier trajectories due to stochastic collocation/path sampling and the higher computational complexity of PDE/BSDE training signals. Occasional spikes in the curves are expected under stochastic training and cosine warm-restart learning rate scheduling.

D. Prediction error analysis

To complement the aggregate pricing metrics reported in Section 4.2, this appendix examines the structure of prediction errors, defined as $\hat{z}_i - z_i$, where z denotes the analytical benchmark price and $\hat{z}$ is the surrogate prediction. Rather than focusing solely on global scores such as MSE and R^2, we study how the errors vary across the range of option values and how they are distributed overall. This provides additional diagnostics for potential systematic bias, heteroskedastic behavior, and tail risk in the pricing errors.

Figure D.2 plots the prediction errors against the corresponding true option prices for all three architectures under the Hull–White and G2++ models. Across both models, the supervised DeepONet exhibits the tightest concentration of errors around the zero line, with dispersion remaining relatively small over most of the price range. At the same time, in the upper tail of the option values, the DeepONet errors show a mild tendency to drift downward, indicating a slight underpricing effect in the relatively sparse high-price region.



(a) Hull–White – DeepONet **(b)** Hull–White – PINN **(c)** Hull–White – DeepBSDE

(d) G2++ – DeepONet **(e)** G2++ – PINN **(f)** G2++ – DeepBSDE

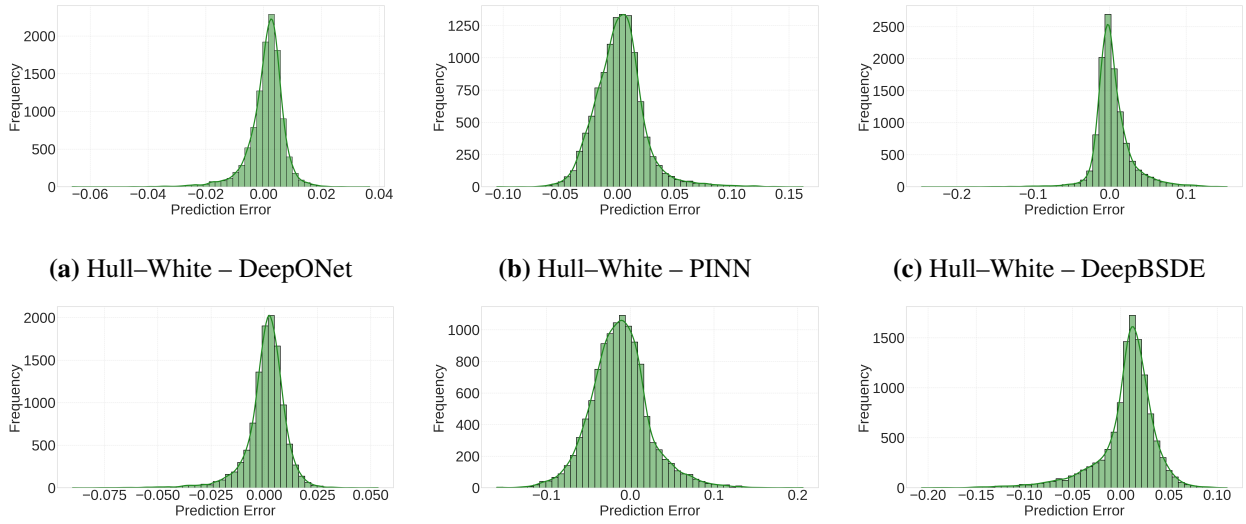
Figure D.2. Prediction errors on the held-out test set plotted against the corresponding analytical option prices. The prediction error is defined as $\hat{z} - z$, where z is the analytical benchmark price and $\hat{z}$ is the surrogate prediction. The first row shows the Hull–White model and the second row shows the G2++ model; columns correspond to DeepONet, PINN, and DeepBSDE.

The theory-informed models display distinctly different error structures. In the Hull–White setting, both PINN and DeepBSDE remain broadly centered around zero, but with visibly wider dispersion than DeepONet. The difference becomes substantially more pronounced in the G2++ case. In particular, the PINN error plot exhibits a clear wedge-shaped pattern, where the spread of the errors increases markedly with the true price and the lower envelope extends to larger negative values. The wedge-shaped error pattern indicates strong heteroskedasticity and suggests that the local PDE residual training signal, which relies on repeated evaluation of higher-order derivatives, can produce a stiffer and less stable optimization landscape as the dimensionality and parameter interactions increase. DeepBSDE, by contrast, shows a more moderate dispersion pattern than PINN, though its errors still widen in higher-price regions, consistent with the accumulation of pathwise variance and discretization effects in the backward recursion.

Figure D.3 reports the histograms of prediction errors for the same test sets. DeepONet yields sharply peaked distributions concentrated near zero in both models, reflecting low variance and

negligible bias for the dominant mass of test samples. The theory-informed models produce broader distributions with heavier tails, indicating a higher frequency of moderate to large deviations. In the G2++ model, this distinction is particularly evident: The PINN's error distribution is the widest, consistent with the pronounced dispersion and heteroskedasticity observed in Figure D.2, while DeepBSDE occupies an intermediate regime that is less dispersed than PINN but clearly broader than DeepONet, reflecting the trade-off between globally constrained pathwise learning and residual-driven local constraints.

In summary, Figures D.2 and D.3 show that price-level errors are generally centered near zero across models, but the error geometry differs substantially across learning paradigms and becomes more discriminative as the model's complexity increases. The supervised DeepONet produces the most concentrated error profiles, whereas the theory-informed surrogates display broader and more structured error patterns, most notably for PINN in the multi factor G2++ setting. These diagnostics support the interpretation in Section 4.2 that when evaluation is restricted to price-level accuracy, objective metric alignment favors supervised learning, while the indirect PDE/BSDE constraints introduce additional numerical and optimization challenges that manifest in the dispersion and tails of the error distributions.



(a) Hull–White – DeepONet **(b)** Hull–White – PINN **(c)** Hull–White – DeepBSDE

(d) G2++ – DeepONet **(e)** G2++ – PINN **(f)** G2++ – DeepBSDE

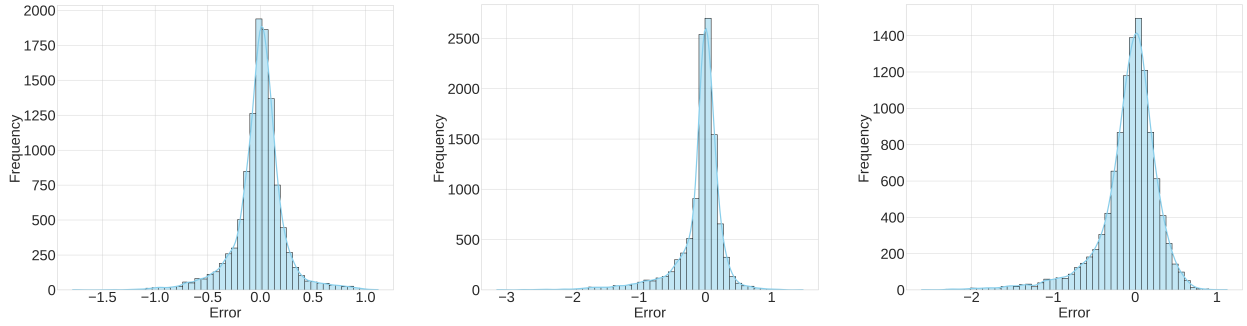
Figure D.3. Histograms of prediction errors $(\hat{z} - z)$ on the held-out test set, where z is the analytical benchmark price and $\hat{z}$ is the surrogate prediction. The first row shows the Hull–White model and the second row shows the G2++ model; the columns correspond to DeepONet, PINN, and DeepBSDE.

E. Vega error analysis

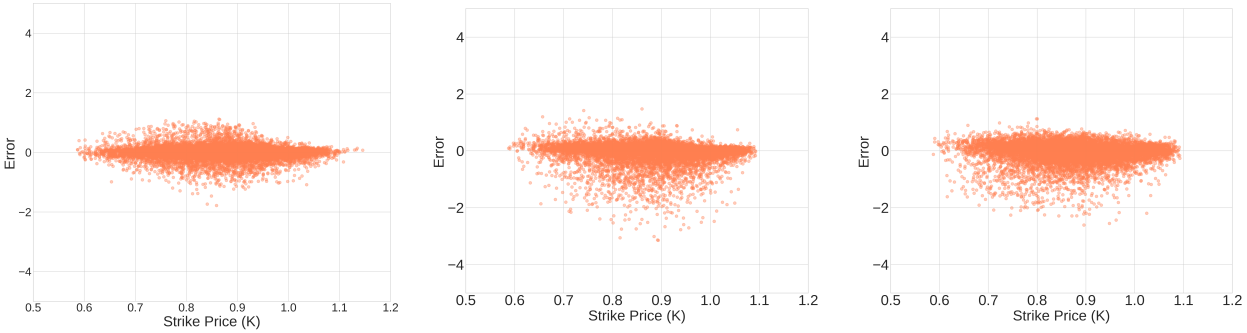
To complement the aggregate Vega R^2 scores reported in Table 3, this appendix examines the structure of Vega errors, defined as the difference between the model-implied Vegas and the corresponding analytical Vegas from the closed-form formulas. We summarize these errors in two complementary ways: (i) Histograms that characterize the overall bias and dispersion, and (ii) scatter

plots of errors against the strike K that reveal whether the inaccuracies concentrate in specific regions of the option domain.

Hull–White model. Figure E.4 reports the Hull–White Vega error distributions and strike-wise error patterns. Across all three architectures, the error histograms are centered close to zero, indicating that none of the methods exhibits a strong systematic bias in V_σ over the test set. The key differences are instead reflected in the dispersion and tail behavior. In particular, DeepONet yields the most concentrated error distribution around zero, consistent with its highest Vega R^2 in Table 3. PINN displays a broader distribution with heavier tails than DeepONet, while DeepBSDE exhibits the widest spread among the three, indicating a higher frequency of moderate to large deviations.



(a) DeepONet: Histogram of V_σ errors **(b)** PINN: Histogram of V_σ errors **(c)** DeepBSDE: Histogram of V_σ errors



(d) DeepONet: V_σ errors across strikes **(e)** PINN: V_σ errors across strikes **(f)** DeepBSDE: V_σ errors across strikes

Figure E.4. Vega error analysis for the Hull–White model on the held-out test set. The top row shows histograms of the V_σ errors, defined as the model-implied (autodiff) V_σ minus the analytical V_σ. The bottom row plots the same V_σ errors versus the strikes. The columns correspond to DeepONet, PINN, and DeepBSDE.

The strike-wise scatter plots further clarify where these deviations occur. For all methods, errors remain relatively tight far from the ATM region, while the dispersion is visibly larger around the ATM region (i.e., strikes close to the corresponding forward). This pattern is consistent with the well-known sensitivity profile of call options: V_σ is typically most pronounced around the ATM, so small imperfections in the learned pricing surface are amplified into larger absolute Vega discrepancies in that region. Conversely, for strikes further away from ATM, Vega is typically smaller, which naturally leads to more muted error magnitudes in those regions. Importantly, the clouds remain broadly symmetric

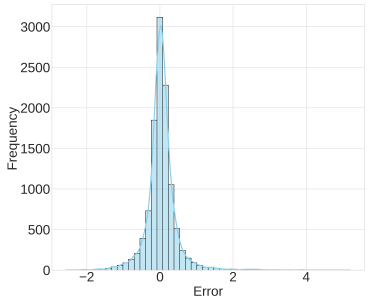
around zero across strikes, suggesting that the dominant effect is increased variance around ATM rather than a systematic strike-dependent directional bias.

G2++ model. Figure E.5 reports the error distributions and strike-wise error patterns for both V_σ and V_η under the two-factor G2++ model. The multi factor setting produces a noticeably more challenging sensitivity landscape, and the error geometry becomes more discriminative across architectures.

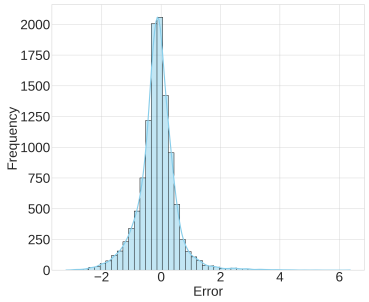
For V_σ, DeepONet and DeepBSDE produce comparatively concentrated histograms around zero, whereas PINN exhibits substantially heavier tails and a wider dispersion. The corresponding scatter plots show that PINN's V_σ errors are more diffuse across strikes, with a larger fraction of outliers and a visibly thicker error band. DeepONet and DeepBSDE, in contrast, maintain a tighter cloud around zero, with dispersion again increasing near the ATM region. As in the Hull–White case, the error band is widest around ATM strikes, where the Vega is largest and differentiation with respect to volatility amplifies small local imperfections in the learned price surface.

For V_η, the separation across methods is even more pronounced. DeepONet shows the narrowest and most symmetric error distribution, and its strike-wise scatter remains relatively compact around zero throughout the strike range. DeepBSDE displays a broader distribution and increased strike-wise dispersion, indicating reduced stability for the second volatility sensitivity. PINN exhibits the least favorable structure: Its histogram is the widest and is visibly shifted away from zero, and the scatter plot shows a consistently larger spread with a noticeable tendency toward negative errors over a broad range of strikes. This qualitative behavior aligns with the ranking implied by Table 3, where DeepONet remains the most reliable architecture for both V_σ and V_η, while the theory-informed approaches—especially PINN for V_η—incur larger dispersion and more pronounced tail risk in the multi-factor regime.

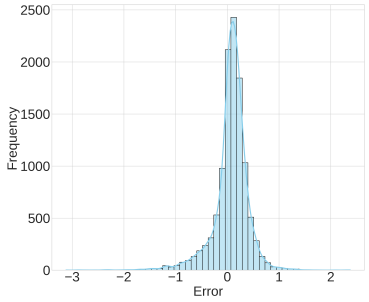
In summary, Figures E.4 and E.5 provide a more granular explanation for the aggregate Vega scores. In the one-factor Hull–White setting, the Vega error distributions are centered close to zero across architectures, suggesting no pronounced systematic bias on average; performance is primarily distinguished by variance and tail thickness, with the largest dispersion concentrated around the ATM region. In the more complex G2++ setting, both dispersion and asymmetry become more prominent, and architecture-dependent differences are amplified—particularly for V_η, where the strike-wise plots reveal broader error clouds and heavier tails and, for some methods, visible shifts away from zero. These diagnostics complement Table 3 by showing not only how much the sensitivities differ from the analytical benchmark on average, but also where the deviations are most likely to occur.



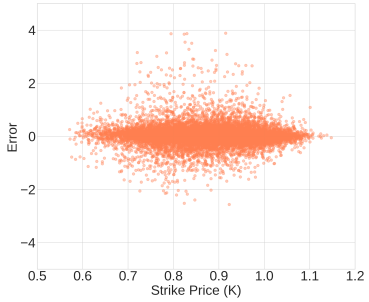
(a) DeepONet: Histogram of V_σ errors



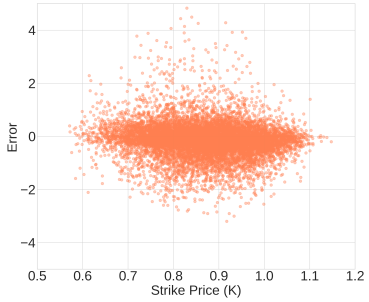
(b) PINN: Histogram of V_σ errors



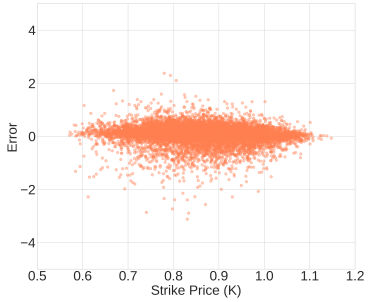
(c) DeepBSDE: Histogram of V_σ errors



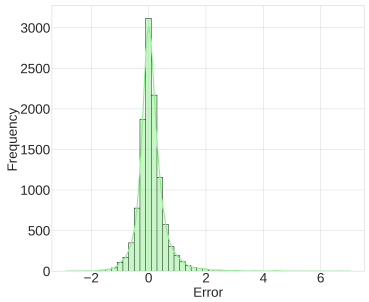
(d) DeepONet: V_σ errors across strikes



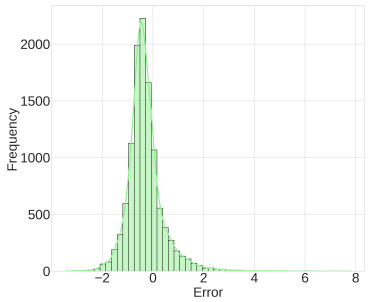
(e) PINN: V_σ errors across strikes



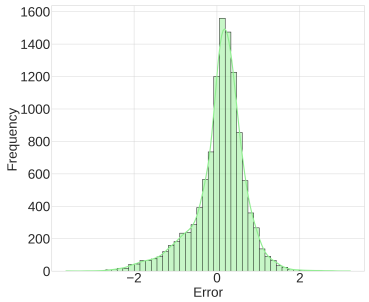
(f) DeepBSDE: V_σ errors across strikes



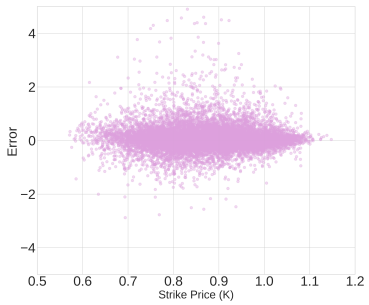
(g) DeepONet: Histogram of V_η errors



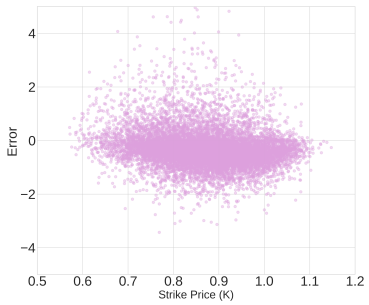
(h) PINN: Histogram of V_η errors



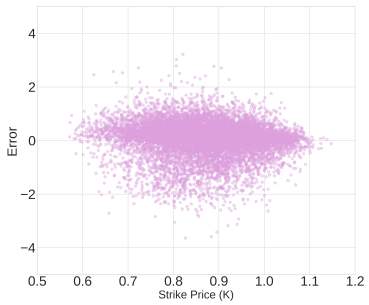
(i) DeepBSDE: Histogram of V_η errors



(j) DeepONet: V_η errors across strikes



(k) PINN: V_η errors across strikes



(l) DeepBSDE: V_η errors across strikes

Figure E.5. Vega error analysis for the G2++ model on the held-out test set. The top half reports V_σ errors (histograms and errors versus strikes), and the bottom half reports V_η errors in the same format. Errors are defined as the model-implied (autodiff) Vega minus the corresponding analytical Vega. Columns correspond to DeepONet, PINN, and DeepBSDE.

AIMS Press