Physics-Informed Neural Networks for High-Dimensional Black–Scholes Option Pricing: Accuracy, Scalability, and Limitations

André Paz Gomes[1(✉)], Isaac P. Santos[1,2], and Lucia Catabriga[1]

[1] PPGI, Universidade Federal do Espírito Santo, Vitória, ES, Brazil
andre.p.gomes@edu.ufes.br, isaac.santos@ufes.br, luciac@inf.ufes.br
[2] DMA, Universidade Federal do Espírito Santo, São Mateus, ES, Brazil

Abstract. This work investigates the use of Physics-Informed Neural Networks (PINNs) for pricing European options under the Black-Scholes equation, including arithmetic basket options, and compares their performance with stochastic and classical numerical methods, namely Monte Carlo, finite differences, and binomial trees. The analysis spans multiple levels of dimensionality, enabling direct comparisons in one and two dimensions and assessing scalability in three and five dimensions. In low-dimensional cases, classical methods remain the most competitive in terms of accuracy and computational efficiency. Finite differences and binomial trees achieve the lowest errors, while Monte Carlo provides robust performance at a higher per-query cost. In these regimes, PINNs yield consistent approximations but do not outperform traditional methods under the adopted computational budget. In higher dimensions, PINNs demonstrate their main advantage by providing mesh-free approximations without the combinatorial growth associated with grid-based methods. Although they exhibit higher errors than Monte Carlo in the tested configurations, they remain applicable and scalable in regimes where deterministic methods become impractical. An additional advantage of PINNs lies in their offline-online structure, enabling fast and reusable inference, as well as efficient computation of sensitivities (Greeks) via automatic differentiation. However, their accuracy remains strongly dependent on training-related factors, with noticeable approximation bias in higher dimensions and near payoff nonsmoothness. Overall, PINNs should not be regarded as replacements for classical methods in low-dimensional problems, but rather as a promising alternative for high-dimensional option pricing, where scalability and flexibility are key requirements.

Keywords: Option pricing · Black–Scholes equation · Physics-informed neural networks · Monte Carlo methods · Finite difference methods · Binomial tree methods

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2027
O. Gervasi et al. (Eds.): ICCSA 2026 Workshops, LNCS 16763, pp. 313–329, 2027.
https://doi.org/10.1007/978-3-032-30530-5_20

1 Introduction

The pricing of financial derivatives relies on accurate and efficient numerical methods, particularly when no closed-form formula is available for the payoff of interest [6]. In the one-dimensional case of the Black-Scholes equation, standard European options admit an analytical solution [3]. However, multi-asset options, such as *basket* options, typically do not admit closed-form solutions for arithmetic payoffs and therefore require the use of numerical methods.

Among the classical approaches, Monte Carlo simulation, tree-based methods (binomial or multinomial), and finite difference discretizations are widely used [6]. Monte Carlo methods scale relatively well with dimensionality, but exhibit slow convergence and may require variance reduction techniques to improve efficiency. Finite differences and tree-based methods, in turn, are highly competitive for low-dimensional problems, but suffer from the curse of dimensionality. In grid-based discretizations, the number of degrees of freedom grows exponentially with the number of underlying assets, which quickly renders these methods impractical as dimensionality increases.

Physics-Informed Neural Networks (PINNs) have emerged as an alternative within the field of scientific machine learning, where neural networks are trained to satisfy the governing partial differential equations and the associated boundary and initial conditions through the minimization of a loss function [11]. In contrast to traditional grid-based numerical methods, PINNs approximate the solution using neural networks and therefore do not require the construction of structured meshes in high-dimensional spaces. This characteristic makes them an appealing candidate for multi-asset option pricing problems.

Recent studies have investigated the use of PINNs in financial applications, including option pricing problems in two dimensions [13] and more complex settings, such as Asian options, which are path-dependent [9]. In addition, the literature on learning-based methods for high-dimensional problems highlights the persistent challenge posed by the curse of dimensionality in option pricing and the ongoing search for scalable alternatives [15].

Despite these advances, a systematic comparison between PINNs and classical numerical methods for option pricing across multiple dimensions remains limited in the literature. In particular, the relative accuracy and computational cost of PINNs with respect to established approaches such as Monte Carlo simulation, finite difference schemes, and tree-based methods have not yet been thoroughly assessed in a unified experimental framework.

In this context, the objective of this work is to investigate whether Physics-Informed Neural Networks (PINNs) can provide a competitive alternative to classical numerical and stochastic approaches for option pricing problems of increasing dimensionality. To this end, we perform a consistent experimental comparison, in terms of MAE, RMSE, and total execution time, among PINNs, Monte Carlo simulation, a Crank–Nicolson finite difference scheme, and binomial tree methods in the one-dimensional case, where the analytical solution of the Black–Scholes equation is available as reference. The four methods are also compared in the two-dimensional case for a European *basket* option, using

a high-accuracy Monte Carlo estimate as the benchmark. Finally, a scalability assessment is conducted for three- and five-dimensional problems, comparing PINNs and Monte Carlo methods in order to reflect the practical infeasibility of tree-based and finite difference approaches in these regimes.

The remainder of this paper is organized as follows. Section 2 presents the Black-Scholes model and the considered option pricing problem. Section 3 describes the numerical methods, including the PINN formulation. Section 4 presents the experimental setup and results. Finally, Sect. 5 summarizes the conclusions.

2 The Black-Scholes Model in d Dimensions

Under the classical assumptions of the Black-Scholes model - a frictionless and arbitrage-free market, continuous trading, a constant risk-free rate r, and constant volatilities - consider a vector of d underlying assets $\mathbf{S}_t = (S_t^{(1)}, \ldots, S_t^{(d)})^\top$ each paying a continuous dividend yield (or carrying costs) q_i.

Let $(\Omega, \mathcal{F}, (\mathcal{F}_t)_{t\geq 0}, \mathbb{Q})$ be a filtered probability space and let $\mathbb{Q}$ denote the risk-neutral measure. Under $\mathbb{Q}$, the asset prices follow d-dimensional correlated geometric Brownian motions [8]:

$$\frac{\mathrm{d}S_t^{(i)}}{S_t^{(i)}} = (r - q_i)\,\mathrm{d}t + \sigma_i\,\mathrm{d}W_t^{(i)}, \qquad i = 1, \ldots, d, \tag{1}$$

where $\sigma_i > 0$ denotes the volatility of asset i. The vector $\boldsymbol{W}_t = (W_t^{(1)}, \ldots, W_t^{(d)})^\top$ represents a d-dimensional Brownian motion whose components exhibit instantaneous correlations

$$\mathrm{d}W_t^{(i)}\,\mathrm{d}W_t^{(j)} = \rho_{ij}\,\mathrm{d}t, \qquad \rho_{ii} = 1, \qquad \boldsymbol{\rho} = [\rho_{ij}] \succeq 0, \tag{2}$$

where $\boldsymbol{\rho}$ is a symmetric positive semidefinite correlation matrix, ensuring a valid covariance structure for the driving noise processes [2].

Consider a European option with payoff $g(\mathbf{S}_T)$ at maturity T. Under the risk-neutral valuation principle, its arbitrage-free price is given by the conditional expectation

$$V(t, \mathbf{S}) = \mathbb{E}^{\mathbb{Q}}\left[e^{-r(T-t)}\, g(\mathbf{S}_T) \,\middle|\, \mathbf{S}_t = \mathbf{S}\right], \tag{3}$$

which corresponds to the probabilistic representation of the option value under the risk-neutral measure [12].

By applying Itô's lemma to $V(t, \mathbf{S}_t)$ and invoking the no-arbitrage argument for self-financing portfolios, one obtains that V satisfies the multidimensional parabolic Black–Scholes partial differential equation:

$$\frac{\partial V}{\partial t} + \sum_{i=1}^{d}(r - q_i)S^{(i)}\frac{\partial V}{\partial S^{(i)}} + \frac{1}{2}\sum_{i=1}^{d}\sum_{j=1}^{d}\rho_{ij}\sigma_i\sigma_j S^{(i)}S^{(j)}\frac{\partial^2 V}{\partial S^{(i)}\partial S^{(j)}} - rV = 0, \tag{4}$$

subject to the terminal condition

$$V(T, \mathbf{S}) = g(\mathbf{S}). \tag{5}$$

The diffusive operator in (4) involves the covariance structure given by the matrix $(\rho_{ij}\sigma_i\sigma_j)$ and generates cross-derivative terms whenever correlations between assets are present. The indices i and j in the double summation run over all underlying assets, that is, $i, j = 1, \ldots, d$. Terms with $i = j$ correspond to the usual second-order derivatives with respect to each individual asset price, whereas terms with $i \neq j$ correspond to mixed second-order derivatives between different asset prices. These mixed derivatives arise from the instantaneous correlations ρ_{ij} between the Brownian motions driving the assets. Therefore, the diffusive operator in (4) involves the covariance structure given by the matrix $(\rho_{ij}\sigma_i\sigma_j)$ and generates cross-derivative terms whenever correlations between assets are present. These mixed derivatives significantly increase the numerical complexity of mesh-based discretizations, particularly finite difference schemes in two and higher spatial dimensions [14].

2.1 European Basket Option Payoff

We consider an arithmetic basket call option with payoff

$$g(\mathbf{S}_T) = \max\left\{\sum_{i=1}^{d} w_i S_T^{(i)} - K,\, 0\right\}, \qquad \sum_{i=1}^{d} w_i = 1,\ w_i \geq 0. \tag{6}$$

For $d = 1$ and $w_1 = 1$, this reduces to the standard European call option, for which a closed-form analytical solution is available [3]. For $d > 1$, the arithmetic basket option does not admit a known closed-form analytical solution, which motivates the use of numerical pricing methods [14].

2.2 Analytical Solution in 1D

In the one-dimensional case ($d = 1$), with European call payoff $g(S_T) = \max(S_T - K, 0)$, the Black-Scholes model admits a closed-form analytical solution. Under the risk-neutral measure $\mathbb{Q}$, the asset price follows a geometric Brownian motion, implying that $\ln(S_T)$ is normally distributed. This property allows the option price to be expressed as the expectation of the discounted payoff and leads to an explicit formula involving the cumulative distribution function of the standard normal distribution [8].

The Black-Scholes price of a European call option is given by [3,8]

$$V(S, t) = S\, e^{-q(T-t)}\Phi(d_1) - K\, e^{-r(T-t)}\Phi(d_2), \tag{7}$$

where q is the continuous dividend yield (when $q = 0$, the classical non-dividend-paying case is recovered),

$$d_1 = \frac{\ln(S/K) + (r - q + \frac{1}{2}\sigma^2)(T-t)}{\sigma\sqrt{T-t}}, \qquad d_2 = d_1 - \sigma\sqrt{T-t}, \tag{8}$$

and $\Phi(\cdot)$ denotes the cumulative distribution function of the standard normal distribution.

The quantities $\Phi(d_1)$ and $\Phi(d_2)$ admit useful financial interpretations. In particular, $\Phi(d_2)$ corresponds to the risk-neutral probability that the option expires in the money, that is, $\mathbb{Q}(S_T > K)$. The term $e^{-q(T-t)}\Phi(d_1)$ appears in the expression for the option's *delta*, which measures the sensitivity of the price with respect to changes in the underlying asset [6].

In this work, Eq. (7) is used as the reference solution in the 1D scenario to measure the numerical error of Monte Carlo, binomial tree, finite differences, and PINN approaches.

3 Numerical Methods

Monte Carlo estimates the risk-neutral price of the option by simulating a large number of trajectories for the underlying assets under the measure $\mathbb{Q}$, followed by the computation of the sample average of the discounted payoff. For each simulated path, the asset dynamics follow the stochastic differential equation defined in Eq. (1). The option price is then approximated by

$$V(t, \mathbf{S}) \approx e^{-r(T-t)} \frac{1}{N} \sum_{k=1}^{N} g(\mathbf{S}_T^{(k)}),$$

where N denotes the number of simulated trajectories. Its accuracy improves as the number of paths increases, with statistical error decaying slowly, typically on the order of $1/\sqrt{N}$, which makes the method particularly attractive in higher dimensions due to its good scalability [5].

Tree-based methods approximate the stochastic dynamics of the underlying assets by a discrete-time recombining tree. In the one-dimensional case, the Cox-Ross-Rubinstein (CRR) binomial scheme is used. At each time step, the asset price may move upward or downward by predetermined factors, and the option value is obtained by backward propagation from the terminal payoff, discounting risk-neutral expectations step by step [4,6].

In the two-dimensional case, a correlated extension of the binomial tree is considered, in which joint probabilities are introduced to capture the correlation between the two underlying assets. However, the number of nodes grows rapidly with dimensionality, which makes tree-based methods computationally expensive in higher-dimensional settings.

Finite differences approximate the derivatives of the Black–Scholes PDE on a computational grid defined over the asset price variables and time, thereby transforming the continuous problem into a linear system of algebraic equations that is solved backward in time. In the one-dimensional case, the Crank–Nicolson scheme is applied. This implicit method is second-order accurate in both time and space and provides a good compromise between stability and accuracy when combined with appropriate boundary conditions [14].

In the two-dimensional case, an ADI (Alternating Direction Implicit) type scheme is used to reduce the computational cost associated with multidimensional grids. The domain is truncated and approximate boundary conditions are imposed to ensure numerical stability. However, in higher dimensions (3D and 5D), direct grid-based discretizations become impractical due to the exponential growth in the number of grid points and the associated memory and computational cost. For this reason, finite difference methods are not considered in these regimes [14].

3.1 PINN Formulation

Physics-Informed Neural Networks (PINNs) approximate the solution of a PDE by means of a neural network $u_\theta(t, \mathbf{S})$, parameterized by θ. The network is trained to satisfy simultaneously (i) the differential equation and (ii) the problem conditions, including the terminal condition and the boundary conditions, by incorporating such constraints directly into the loss function [11]. The main conceptual advantage of this approach is its mesh-free character: instead of explicitly discretizing the domain on a regular grid, as in finite differences, points are sampled in the continuous domain and violations of the PDE at those points are penalized during training.

Given the multidimensional Black-Scholes equation defined in Eq. (4), the differential operator $\mathcal{N}[\cdot]$ is defined as

$$\mathcal{N}[V](t, \mathbf{S}) = \frac{\partial V}{\partial t} + \sum_{i=1}^{d} (r - q_i) S^{(i)} \frac{\partial V}{\partial S^{(i)}} + \frac{1}{2} \sum_{i=1}^{d} \sum_{j=1}^{d} \rho_{ij} \sigma_i \sigma_j S^{(i)} S^{(j)} \frac{\partial^2 V}{\partial S^{(i)} \partial S^{(j)}} - rV. \tag{9}$$

The PINN seeks a function u_θ such that $\mathcal{N}[u_\theta](t, \mathbf{S}) \approx 0$ for $(t, \mathbf{S})$ in the interior of the domain. For this purpose, the residual function is defined as

$$f_\theta(t, \mathbf{S}) = \mathcal{N}[u_\theta](t, \mathbf{S}), \tag{10}$$

computed through automatic differentiation. This technique enables the consistent evaluation of first- and second-order derivatives of the neural network output with respect to the inputs $(t, \mathbf{S})$, avoiding the need for finite difference approximations [11].

For European options, the terminal condition (payoff) is imposed at $t = T$:

$$u_\theta(T, \mathbf{S}) \approx g(\mathbf{S}) \equiv \max\left\{ \sum_{i=1}^{d} w_i S^{(i)} - K,\, 0 \right\}. \tag{11}$$

In addition, since the domain is truncated in practice, the asset prices are restricted to $S^{(i)} \in [0, S^{(i)}_{\max}]$. Approximate boundary conditions are imposed on the truncated domain. In particular, in the case of a call, low values of $S^{(i)}$ drive the price toward values close to zero, whereas high values make the price grow approximately linearly with the basket value, due to the dominance of the

payoff. In practice, boundary constraints consistent with the truncated computational domain are incorporated into the loss function.

Training is performed by minimizing a composite loss function that enforces the PDE residual, the terminal condition, and the boundary conditions:

$$\begin{aligned}\mathcal{L}(\theta) = \lambda_{\text{pde}} \frac{1}{N_f} \sum_{k=1}^{N_f} \left| f_\theta(\tau_k, \mathbf{S}_k) \right|^2 \\ + \lambda_{\text{ic}} \frac{1}{N_0} \sum_{m=1}^{N_0} \left| u_\theta(0, \mathbf{S}_m) - g(\mathbf{S}_m) \right|^2 \\ + \lambda_{\text{bc}} \frac{1}{N_b} \sum_{n=1}^{N_b} \left| u_\theta(\tau_n, \mathbf{S}_n) - b(\tau_n, \mathbf{S}_n) \right|^2 .\end{aligned} \tag{12}$$

where $\{(\tau_k, \mathbf{S}_k)\}$ denote collocation points sampled in the interior of the space-time domain, $\{(0, \mathbf{S}_m)\}$ correspond to points enforcing the payoff condition at $\tau = 0$, and $\{(\tau_n, \mathbf{S}_n)\}$ represent boundary points. The term with N_0 is referred to as the initial-condition loss because the original terminal condition $V(T, \mathbf{S}) = g(\mathbf{S})$ becomes $V(0, \mathbf{S}) = g(\mathbf{S})$ after introducing the time-to-maturity variable $\tau = T - t$. The weights $\lambda_{(\cdot)}$ control the balance between enforcing the PDE and imposing the terminal and boundary conditions, and adjustments are often required to ensure stability and good accuracy, especially in problems with different magnitude scales among the terms [11]. All loss components in Eq. (12) were assigned unit weights, i.e., $\lambda_{\text{pde}} = \lambda_{\text{ic}} = \lambda_{\text{bc}} = 1$, including each boundary-condition term in the multidimensional cases.

Unlike grid-based methods, the performance of PINN depends critically on the sampling of collocation points. Random sampling strategies, such as uniform sampling or Latin Hypercube sampling, are usually employed, and the density of points may be increased in regions where the solution is more difficult to approximate, such as near payoff nonsmoothness. Optimization is performed by gradient-based methods such as Adam and may be refined by quasi-Newton optimizers, such as L-BFGS, to accelerate convergence and reduce the final residual [11].

4 Numerical Experiments

We solve the Black-Scholes PDE introduced in Sect. 2 and evaluate the accuracy of the numerical solutions on a test set within the computational domain. The metrics considered are the mean absolute error (MAE) and the root mean square error (RMSE), defined by:

$$\text{MAE} = \frac{1}{N_{\text{test}}} \sum_{i=1}^{N_{\text{test}}} \left| V_i^{\text{num}} - V_i^{\text{ref}} \right|, \text{RMSE} = \left(\frac{1}{N_{\text{test}}} \sum_{i=1}^{N_{\text{test}}} \left(V_i^{\text{num}} - V_i^{\text{ref}} \right)^2 \right)^{1/2}, \tag{13}$$

where N_{test} is the evaluation points, V_i^{num} denotes the numerical approximation produced by a given method and V_i^{ref} denotes the corresponding reference value. Lower values of MAE and RMSE indicate better agreement with the reference solution. Computational cost is reported as the total execution time, including training for learning-based methods and evaluation on the test set.

In this work, the PINNs were implemented using the DeepXDE library with the `tensorflow.compat.v1` backend, which provides: (i) declarative definition of the PDE and the conditions, (ii) generation of collocation, boundary, and terminal points, (iii) automatic differentiation, and (iv) training and evaluation routines [7]. This enables a uniform implementation across different dimensions ($d = 1, 2, 3, 5$), preserving the same general loss formulation and changing only the dimension of the input vector and the corresponding differential operator.

The PINN is modeled as a fully connected feedforward neural network (FNN) that approximates the option price $V(S, \tau)$ on the computational domain $\Omega = [0, S_{\max}] \times [0, T]$, where $\tau = T - t$ denotes the time to maturity. The network takes (S, τ) as input and outputs a scalar approximation of V, the option value. The architecture $[2, 64, 64, 64, 1]$ is adopted, consisting of three hidden layers with 64 neurons each, using tanh activation and Glorot normal initialization. The choice of three hidden layers with 64 neurons each was motivated by a compromise between approximation capacity and computational cost. Since the Black–Scholes equation involves first- and second-order derivatives of the network output, excessively shallow networks may lack sufficient expressive power to represent the option price surface and its derivatives accurately. Conversely, substantially larger networks increase the cost of automatic differentiation and training, especially in higher-dimensional settings. Therefore, the architecture $[d+1, 64, 64, 64, 1]$ was adopted as a moderate-size fully connected network, which is consistent with moderate-size architectures previously used in PINN applications [1]. The same architectural pattern was maintained across all dimensions in order to ensure a consistent comparison framework. Input normalization is performed through feature scaling, with S normalized by $S_{\max}$ and τ by T, to improve numerical stability and facilitate optimization.

Training minimizes a composite loss function composed of the PDE residual evaluated at interior collocation points, together with constraints imposed by the initial condition in the transformed time variable, $V(S, 0) = \max(S - K, 0)$, and Dirichlet boundary conditions $V(0, \tau) = 0$ and the asymptotic condition $V(S_{\max}, \tau) \approx S_{\max} - Ke^{-r\tau}$. The training dataset consists of $N_f = 6000$ interior points, $N_b = 1000$ boundary points, and $N_0 = 1000$ initial points, automatically generated by the framework. Optimization is performed using the Adam method with learning rate 10^{-3} for 50,000 iterations.

All experiments were conducted on a machine with an Intel Xeon CPU (2.20 GHz), 12.7 GB RAM, and an NVIDIA Tesla T4 GPU (16 GB VRAM), running Python 3.10 and DeepXDE 1.6.1. This configuration was maintained consistently for all experiments to ensure fair comparisons of execution times.

Four experimental scenarios are considered, varying the dimensionality and the payoff structure. In all cases, maturity $T = 1$ and risk-free rate $r = 0.05$ are

fixed. For each scenario, we define: (i) a test set used exclusively for computing MAE and RMSE, and (ii) a reference solution, obtained analytically when available or via high-accuracy Monte Carlo simulations otherwise.

In the 1D case, a European call option is considered with $S_{\max} = 200$, $K = 50$, $T = 1$, $r = 0.05$, and volatility $\sigma = 0.20$. Evaluation is performed on a regular 40×40 grid over the computational domain, Ω. The analytical Black-Scholes formula is used as reference, providing a deterministic ground truth and eliminating statistical uncertainty in the error metrics [3].

In higher dimensions, European basket options are considered with nonnegative and normalized weights ($\sum_i w_i = 1$), strike $K = 50$, $T = 1$, and $r = 0.05$. In the 2D case, equal weights $w_1 = w_2 = 0.5$, volatilities $(\sigma_1, \sigma_2) = (0.20, 0.25)$, and correlation $\rho = 0.30$ are adopted. Since no closed-form solution exists, the reference is computed via Monte Carlo with 600,000 paths. The test set consists of 400 randomly sampled points.

Similarly, in the 3D case, equal weights $w_1 = w_2 = w_3 = \frac{1}{3}$, volatilities $\sigma = (0.20, 0.25, 0.22)$, and correlations $(\rho_{12}, \rho_{13}, \rho_{23}) = (0.30, 0.20, 0.25)$ are used, with the same Monte Carlo configuration and test set size. Finally, in the 5D case, equal weights $w_i = \frac{1}{5}$, $i = 1, 2, \cdots, 5$, volatilities $\sigma = (0.20, 0.25, 0.22, 0.18, 0.24)$, and uniform correlation $\rho_{ij} = 0.25$, for $i \neq j$, are considered. The reference solution is again obtained via Monte Carlo with 600,000 paths, and the test set contains 400 random points. This last scenario highlights the regime in which deterministic grid-based methods become impractical, emphasizing the scalability of PINNs and Monte Carlo methods under a common evaluation protocol.

4.1 1D Case: Four Methods Versus Analytical Solution

In the one-dimensional case, the comparison is performed against the closed-form Black-Scholes formula for the European call option, allowing the numerical error to be assessed directly throughout the computational domain. To mitigate oscillations and numerical noise induced by the nonsmooth payoff near $\tau \approx 0$, the finite difference scheme uses the Crank-Nicolson (CN) method with Rannacher smoothing, consisting of two half-steps of implicit Euler prior to CN, following the original proposal [10] and its well-established use in option pricing with nonsmooth payoffs.

PINN training in this experiment required 161.765 s, a cost that was incurred only during the offline phase. After training, the network can be reused for multiple price and Greeks queries with reduced inference cost. Similarly, the finite difference method requires a PDE solution stage (0.499 s) to construct the surface on the grid; subsequent queries are dominated by interpolation over the precomputed grid. Consequently, Table 1 reports only the prediction time on the test set (*pred_time*), highlighting the cost per query after any offline computation.

Table 1 highlights the trade-off between accuracy and cost per query. The binomial tree achieves the smallest global error, but exhibits the highest prediction time, since each query involves a backward evaluation over the tree structure. With Rannacher smoothing, the finite difference method combines

Table 1. 1D case (European call). Errors on the test set with respect to the analytical price and prediction time (*pred_time*) on the test set.

Method	MAE	RMSE	*pred_time* [s]
Finite Difference	3.67×10^{-4}	1.07×10^{-3}	$\mathbf{2.66 \times 10^{-3}}$
Binomial Tree	$\mathbf{9.50 \times 10^{-5}}$	$\mathbf{3.12 \times 10^{-4}}$	7.01
Monte Carlo	7.25×10^{-3}	1.21×10^{-2}	6.13
PINN	1.39×10^{-1}	1.57×10^{-1}	2.22×10^{-1}

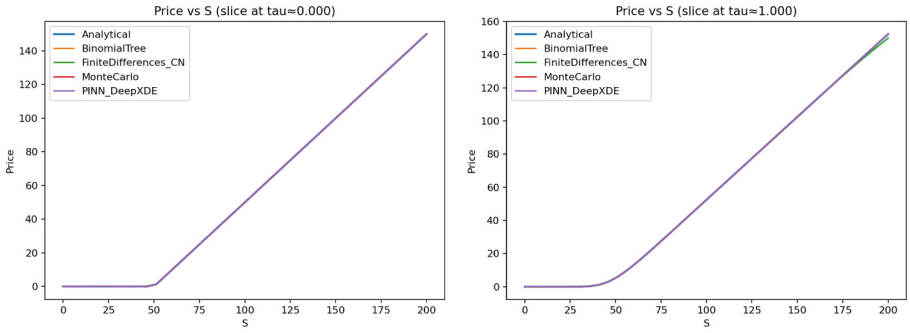
extremely fast prediction, via interpolation over the precomputed surface, with low aggregate error, rendering it competitive with the binomial tree in terms of accuracy while being significantly faster in prediction. Monte Carlo provides moderate accuracy, but at a high prediction cost, as each price requires a large number of simulated paths to reduce variance. The PINN, in this configuration, presents significantly higher error despite its low inference cost after training. This behavior reflects approximation bias and potentially the need for better sampling and better balancing of the loss weights under the adopted computational budget.

To complement the quantitative analysis, Fig. 1 presents price profiles $C(S, \tau)$ as functions of S at two representative stages of the contract: near maturity ($\tau \approx 0$) and at the initial time ($\tau \approx T$). A close agreement is observed between the analytical solution and the binomial tree and finite difference approximations. Near maturity ($\tau \approx 0$), all methods accurately capture the payoff structure, despite its nonsmoothness, due to the explicit inclusion of training points enforcing the initial condition. For larger values of τ, small deviations become more noticeable, as the solution is governed by the PDE dynamics and must be approximated through the learned representation.

Figure 2 summarizes the distribution of the absolute error across the test set on a logarithmic scale. The binomial tree exhibits the smallest errors, reaching values close to machine precision, although with a wider spread that indicates localized larger errors. The finite difference method exhibits slightly higher error with moderate dispersion, reflecting a more uniform accuracy across the domain, which can be attributed to the use of Rannacher smoothing. Monte Carlo shows consistently larger errors, but with a relatively narrow distribution, indicating a more uniform error pattern dominated by sampling variability, whereas the PINN displays higher errors throughout most of the domain, consistent with the aggregate MAE and RMSE values reported in Table 1.

4.2 2D Case: Four Methods (Monte Carlo Reference)

In the two-dimensional case, corresponding to a European basket option, no general closed-form formula exists to serve as a direct reference throughout the domain. Therefore, a high-accuracy Monte Carlo simulation with 600,000 paths is used as a reference, exclusively for error computation. In this experiment,



(a) Maturity: $\tau \approx 0.000$ (b) Initial time: $\tau \approx 1.000$

Fig. 1. 1D case. Profiles of $C(S, \tau)$ at maturity and at the initial time ($\tau = T - t$), comparing the analytical solution and the numerical approximations.

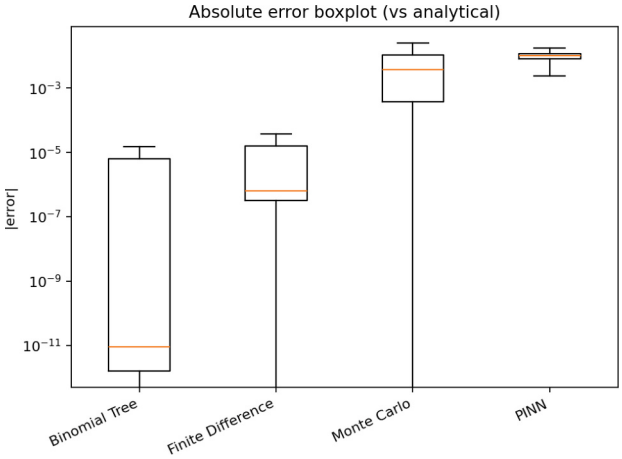


Fig. 2. 1D case. Boxplot of the absolute error (log scale) with respect to the analytical solution, aggregating errors over the test set.

the PINN was trained for 50,000 iterations with periodic resampling of PDE-points, resulting in 185.694 s of training in the offline phase. After this one-time cost, inference becomes significantly faster. Similarly, the two-dimensional finite difference solution was obtained using a fully implicit Crank-Nicolson scheme, requiring 0.136 s to construct the solution surface; subsequent predictions are dominated by interpolation. To emphasize the cost per query after offline stages, Table 2 reports the prediction time on the test set (*pred_ time*) along with aggregate errors relative to the reference.

To complement the aggregate results in Table 2, Fig. 3 presents the distribution of the absolute error over the test set in logarithmic scale, highlighting differences in dispersion and the presence of outliers.

Table 2. 2D case (European basket option). Errors with respect to the high-accuracy Monte Carlo reference and prediction time (*pred_ time*) on the test set.

Method	MAE	RMSE	*pred_ time* [s]
Finite Difference	5.08×10^{-3}	7.09×10^{-3}	$\mathbf{3.85 \times 10^{-4}}$
Binomial Tree	$\mathbf{4.19 \times 10^{-3}}$	$\mathbf{5.33 \times 10^{-3}}$	2.89×10^{02}
Monte Carlo	9.18×10^{-3}	1.19×10^{-2}	1.25
PINN	1.06×10^{-2}	1.95×10^{-2}	2.53×10^{-1}

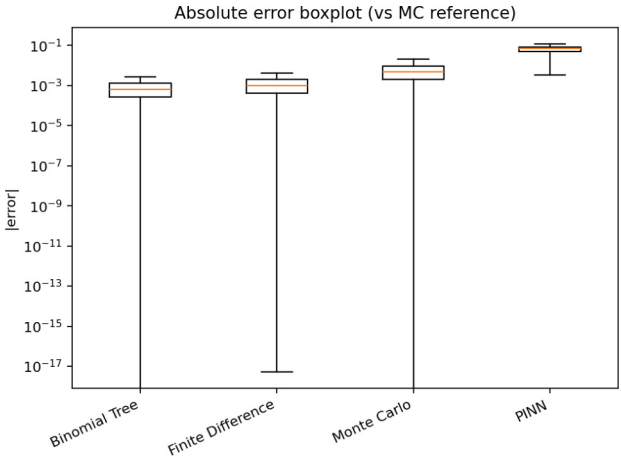


Fig. 3. 2D case. Boxplot of the absolute error (log scale) with respect to the Monte Carlo reference, aggregating errors over the test set.

Table 2 and Fig. 3 reveal four distinct performance regimes. The two-dimensional binomial tree, considering N=120, achieves the smallest aggregate error, but at an extremely high prediction time, reflecting the combinatorial growth in the number of nodes. The finite difference method with Crank–Nicolson exhibits low error and negligible *pred_ time*, as queries reduce to interpolation over the precomputed grid. Monte Carlo with 60,000 paths provides a robust balance between cost and accuracy, with error consistent with sampling variance and controllable through the number of paths. The PINN, while mesh-free and capable of fast inference after training, remains less accurate than Monte Carlo in this experiment, particularly in terms of RMSE, indicating the presence of approximation bias under the adopted training budget and sampling strategy.

4.3 3D Case: PINN Versus Monte Carlo

In the three-dimensional case, corresponding to a European basket option, a high-accuracy Monte Carlo with 600,000 paths is adopted as a reference solution. This reference is used exclusively to compute the errors of both the Monte Carlo

method with 80,000 paths and the PINN predictions. Table 3 summarizes the results for the two evaluated methods, reporting aggregate errors relative to this reference and the prediction time (*pred_time*) on the test set. For the PINN, it is important to highlight the existence of an offline training cost, which in this experiment amounts to 241.972955 s, while pointwise inference remains fast once training is completed.

Table 3. 3D case (European basket option). Errors with respect to the high-accuracy Monte Carlo reference and prediction time (*pred_time*) on the test set.

Method	MAE	RMSE	*pred_time* [s]
Monte Carlo	$\mathbf{3.89 \times 10^{-3}}$	$\mathbf{5.68 \times 10^{-3}}$	1.31
PINN	6.57×10^{-2}	2.06×10^{-1}	$\mathbf{3.95 \times 10^{-1}}$

In this three-dimensional setting, the analysis is restricted to Monte Carlo and PINNs, since deterministic grid-based methods rapidly become computationally prohibitive in terms of memory and execution time as dimensionality increases. Figure 4 illustrates the distribution of the pointwise absolute error over the test set in logarithmic scale.

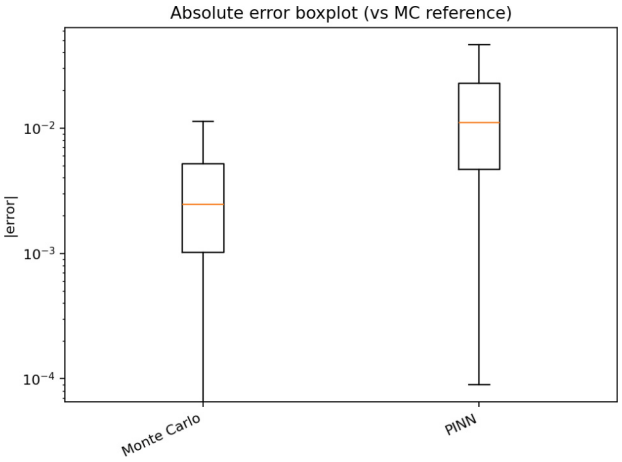


Fig. 4. 3D case. Boxplot of the absolute error (log scale) with respect to the Monte Carlo reference, aggregating errors over the test set.

Table 3 shows that Monte Carlo maintains low aggregate error at moderate computational cost under the adopted sampling budget, constituting a strong baseline in three dimensions when accuracy can be controlled via the number of simulated paths. The PINN, in contrast, retains the advantage of being mesh-free

and achieves faster inference on the test set, but at the expense of substantially higher aggregate error. Figure 4 reinforces this observation by showing that the absolute error distribution of the PINN is shifted toward higher values compared to Monte Carlo. This behavior suggests the predominance of approximation bias, in addition to variance effects, which is expected when learning solutions associated with a nonsmooth payoff. Overall, in the evaluated configuration, Monte Carlo remains superior in terms of accuracy, whereas the PINN provides a viable alternative in higher dimensions, characterized by a high offline cost but fast and reusable inference once trained. This result highlights the increasing difficulty of balancing approximation accuracy and computational cost for PINNs as dimensionality grows, reinforcing the need for improved training strategies in high-dimensional settings.

4.4 5D Case: PINN Versus Monte Carlo

In the five-dimensional case, corresponding to a European basket option, a high-accuracy Monte Carlo simulation with 600,000 paths is adopted as a reference solution. This reference is used exclusively to compute the errors of both the Monte Carlo method with 80,000 paths and the PINN predictions on the test set. Table 4 reports the aggregate errors, MAE and RMSE, relative to this reference, along with the prediction time (*pred_ time*) on the test set. For the PINN, the existence of an offline training cost must be emphasized, amounting in this experiment to 416.840 s, while inference remains fast once training is completed.

Table 4. 5D case (European basket option). Errors with respect to the high-accuracy Monte Carlo reference and prediction time (*pred_ time*) on the test set.

Method	MAE	RMSE	*pred_ time* [s]
Monte Carlo	$\mathbf{3.68 \times 10^{-3}}$	$\mathbf{4.86 \times 10^{-3}}$	1.90
PINN	2.57×10^{-2}	8.17×10^{-2}	$\mathbf{4.21 \times 10^{-1}}$

As in the three-dimensional case, Fig. 5 presents the distribution of the absolute error on the test set in logarithmic scale.

The five-dimensional results further highlight the role of PINNs as mesh-free approximators in regimes where grid-based discretizations become computationally prohibitive. Table 4 shows that Monte Carlo maintains an excellent trade-off between accuracy and computational cost under the adopted sampling budget, providing a strong and reliable baseline. The PINN, in contrast, remains applicable without suffering from the combinatorial explosion associated with grid-based methods and achieves faster inference on the test set. However, this comes at the expense of higher aggregate error and a computational cost dominated by the offline training phase. Figure 5 shows that the error distribution of the PINN is shifted toward higher values compared to Monte Carlo, indicating that performance in this configuration is primarily limited by approximation bias, as well

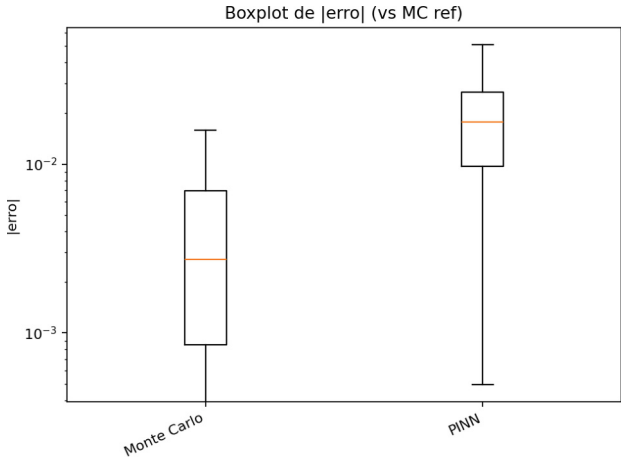


Fig. 5. 5D case. Boxplot of the absolute error (log scale) with respect to the Monte Carlo reference, aggregating errors over the test set.

as by the difficulty of simultaneously enforcing the PDE residual and the nonsmooth terminal condition in higher dimensions. Nevertheless, the PINN remains attractive as a reusable global surrogate, particularly in scenarios requiring multiple pricing evaluations and the efficient computation of Greeks after training.

5 Conclusion

This work investigated the use of Physics-Informed Neural Networks (PINNs) for pricing European options under the Black–Scholes equation, including arithmetic basket options, and compared their performance with classical numerical methods such as Monte Carlo, finite differences, and binomial trees. The analysis spanned multiple levels of dimensionality, enabling direct comparisons in 1D and 2D, as well as an assessment of scalability in 3D and 5D, where deterministic mesh-based methods become computationally prohibitive.

The results indicate that, in low-dimensions settings, classical methods remain the most competitive choices in terms of accuracy and computational cost. In 1D, the binomial trees and the finite differences achieved the smallest errors relatives to the analytical solution, while Monte Carlo provided consistent results at a higher per-query cost. In 2D, binomial trees again yielded the highest accuracy, but with prohibitive prediction time, whereas finite differences offered an excellent balance between accuracy and efficiency. In these regimes, PINNs proved to be functional and stable, but less competitive in accuracy under the adopted computational budget.

In contrast, the experiments in 3D and 5D highlight the main potential of PINNs: their applicability in high-dimensional settings, where grid-based methods suffer from the curse of dimensionality. Although PINNs exhibited higher

aggregate errors than Monte Carlo in the tested configurations, they maintained approximation capability without the combinatorial explosion associated with multidimensional discretization. This reinforces their relevance as a mesh-free alternative for high-dimensional problems.

An important advantage of PINNs is that the computational cost is largely concentrated in the offline phase. Once trained, the model can be reused for multiple pricing and sensitivity (Greeks) evaluations, with low inference cost. Moreover, the continuous nature of the approximation enables the direct computation of sensitivities via automatic differentiation, which is particularly attractive in applications involving repeated queries, calibration, or optimization.

At the same time, the experiments reveal important limitations. The PINN exhibited significant approximation bias, especially in higher dimensions and in regions influenced by payoff nonsmoothness. This behavior highlights the strong dependence of performance on factors such as sampling strategy, loss balancing, network architecture, variable scaling, and training procedures. Therefore, the results presented here should be interpreted as representative of a specific implementation, rather than as a general assessment of all PINN-based approaches.

Overall, PINNs should not be regarded as universal replacements for classical methods, particularly in low-dimensional problems, where finite differences and binomial trees remain superior. However, they emerge as a promising alternative for high-dimensional regimes, where scalability becomes a critical concern. Their ability to provide a reusable global approximation, combined with efficient inference and direct access to Greeks, makes them especially appealing when a continuous surrogate model is required.

Future work includes improving collocation sampling strategies, particularly near nonsmooth regions; developing adaptive loss-balancing techniques; exploring deeper or problem-specific network architectures; incorporating hard constraints to enforce boundary and terminal conditions; and extending the framework to more complex derivatives, such as American, Asian, and path-dependent options. In addition, hybrid approaches combining PINNs with Monte Carlo or other numerical methods represent a promising direction for leveraging both statistical robustness and approximation flexibility. These findings contribute to a clearer understanding of the practical role of PINNs in computational finance, particularly in relation to the trade-off between scalability and accuracy in high-dimensional problems.

Declaration of Generative AI and AI-assisted technologies in the writing process

During the preparation of this work the authors used ChatGPT (OpenAI) in order to assist with language revision and text improvement. After using this tool, the authors reviewed and edited the content as needed and take full responsibility for the content of the publication.

Acknowledgments. This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001 - and

by the Fundação de Amparo à Pesquisa e Inovação do Espírito Santo (FAPES), Grant Notice N° 02/2026.

References

1. Arap, O., et al.: Gabor-enhanced physics-informed neural networks for fast simulations of acoustic wavefields. arXiv preprint arXiv:2502.17134 (2025). https://arxiv.org/abs/2502.17134
2. Björk, T.: Arbitrage Theory in Continuous Time, 3rd edn. Oxford University Press, Oxford (2009)
3. Black, F., Scholes, M.: The pricing of options and corporate liabilities. J. Polit. Econ. **81**(3), 637–654 (1973)
4. Cox, J.C., Ross, S.A., Rubinstein, M.: Option pricing: a simplified approach. J. Financ. Econ. **7**(3), 229–263 (1979)
5. Glasserman, P.: Monte Carlo Methods in Financial Engineering. Springer, New York (2003)
6. Hull, J.: Options, Futures, and Other Derivatives, 10th edn. Pearson, Harlow (2017)
7. Lu, L., Meng, X., Mao, Z., Karniadakis, G.E.: Deepxde: a deep learning library for solving differential equations. J. Comput. Phys. **429**, 109926 (2021)
8. Merton, R.C.: Theory of rational option pricing. Bell J. Econ. Manag. Sci. **4**(1), 141–183 (1973)
9. Park, S., Moon, K.S., Kim, H.: Asian option pricing using the physics-informed neural networks method. Econom. Comput. Econom. Cybernet. Stud. Res. **59**(1), 5–20 (2025). https://doi.org/10.24818/18423264/59.1.25.01
10. Pooley, D.M., Vetzal, K.R., Forsyth, P.A.: Convergence remedies for non-smooth payoffs in option pricing. University of Waterloo, Tech. rep. (2002)
11. Raissi, M., Perdikaris, P., Karniadakis, G.E.: Physics-informed neural networks: a deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. J. Comput. Phys. **378**, 686–707 (2019)
12. Shreve, S.E.: Stochastic Calculus for Finance II: Continuous-Time Models. Springer, New York (2004)
13. Tavakoli, et al.: European option pricing with two underlying assets using Pinn (2024), publication details to be completed
14. Tavella, D., Randall, C.: Pricing Financial Instruments: The Finite Difference Method. Wiley, New York (2000)
15. Zhuang, J., Ding, D., Lu, W., Wu, X., Yuan, G.: A gaussian process based method with deep kernel learning for pricing high-dimensional American options (2024). arXiv preprint arXiv:2311.07211