mathematics

MDPI

Article

A Flexible Deep BSDE Framework for European Option Pricing: From Theoretical Foundations to High-Dimensional Financial Market Applications

Dongya Deng [1], Kenny Tee [2] and Donghai Xu [1,*]

[1] School of Business, Hebei Normal University, Shijiazhuang 050024, China; dongyadeng@hebtu.edu.cn
[2] JCU Singapore Business School, James Cook University, Singapore 387380, Singapore; kenny.tee@jcu.edu.au
* Correspondence: xu_tony77@hebtu.edu.cn

Abstract

This paper develops a flexible backward stochastic differential equation (BSDE)-based framework for pricing European-style derivatives, with a focus on bridging the theoretical foundations and practical applications in high-dimensional financial markets. We first establish the equivalence between the BSDE representation and the classical Black–Scholes PDE, and derive the explicit form of the risk-neutral driver. Building on this theoretical foundation, we propose a deep learning-based numerical scheme. The forward dynamics are discretized via an Euler scheme. The proposed Deep BSDE solver is tested on one-dimensional European call and put options as well as multi-dimensional (5, 10, 20, and 100 dimension) basket and max options. The numerical results demonstrate high accuracy and consistency with the Monte Carlo and analytic (if available) benchmarks. By integrating the theoretical underpinnings with practical numerical implementation, the proposed flexible Deep BSDE framework provides a reliable and adaptable solution for derivative pricing in both standard and complex multi-asset financial market scenarios.

Keywords: backward stochastic differential equation (BSDE); European option pricing; risk-neutral measure; deep learning; high-dimensional derivatives

MSC: 91G20; 60H35; 65C30; 91G60

1. Introduction

The valuation of financial derivatives, particularly European-style multi-asset options such as basket options, max options, and geometric basket options, plays a central role in modern quantitative finance. Accurate pricing is essential for risk management, hedging, portfolio optimization, and the design of structured financial products. The classical Black–Scholes–Merton (BSM) framework provides closed-form solutions for single-asset European options under idealized assumptions [1,2]. However, when extended to multi-asset contexts, the analytical tractability diminishes due to correlation structures, high-dimensional dynamics, and complex payoff forms [3–5]. Furthermore, the distinction between real-world and risk-neutral measures, the role of market prices of risk, and the transformation between these measures become more intricate in high dimensions. Understanding this transformation is not only mathematically relevant but essential for model calibration, risk premium interpretation, and arbitrage-free pricing.

The traditional approach to option pricing relies on partial differential equations (PDEs) derived from no-arbitrage arguments, such as the seminal BSM PDE. This line

Check for updates

Academic Editor: Jonathan Blackledge

Received: 28 September 2026
Revised: 30 October 2026
Accepted: 11 November 2026
Published: 26 November 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

of research establishes the equivalence between martingale pricing under a risk-neutral measure and solving the associated PDE with terminal conditions. For multi-asset derivatives, the resulting PDE becomes high-dimensional, making analytic solutions rare and numerical solutions computationally demanding. Monte Carlo (MC) simulation is one of the most flexible and widely used numerical methods for multi-asset derivative pricing, particularly due to its dimension-independent convergence rate [6]. It is well suited for high-dimensional and path-dependent contracts, but suffers from slow convergence and low efficiency. Therefore, variance reduction techniques, including control variates, antithetic sampling, and stratification, are adopted to improve efficiency, yet they still cannot overcome the curse of dimensionality. Binomial and trinomial trees, while elegant for one-dimensional settings [7], do not scale well to a multi-asset case and are generally limited to low-dimensional, European or simple American payoffs. Quasi-Monte Carlo methods and advanced discretization schemes partially alleviate the computational burdens, but challenges remain for high-dimensional or strongly path-dependent derivatives, where the existing methods still hit efficiency boundaries.

Backward stochastic differential equations (BSDEs) provide an alternative probabilistic formulation of option pricing, establishing a deep connection with (nonlinear) PDEs through the nonlinear Feynman–Kac formula [8,9]. BSDEs are particularly useful for nonlinear payoffs, path dependence, and early exercise features. The early numerical schemes for BSDEs include random walk-based methods for one-dimensional Brownian motion, with implicit and explicit discretizations and convergence proofs [10].

Traditional methods rely on time discretization schemes, such as explicit and implicit Euler methods, which approximate the backward dynamics by discretizing the time interval and recursively computing the conditional expectations. To evaluate these expectations, Monte Carlo techniques are often employed, sometimes combined with regression-based methods to approximate the conditional expectation of future values. Multi-step schemes, which leverage higher-order discretizations for improved convergence, and branching diffusion approaches, which are suitable for certain nonlinear BSDEs, are widely adopted for solving BSDEs. More recent developments include deep learning-based methods, where neural networks approximate the solution processes Y_t and Z_t, enabling efficient treatment of high-dimensional problems [11].

Deep learning has emerged as a powerful tool for solving high-dimensional partial differential equations (PDEs) and backward stochastic differential equations (BSDEs), stimulated by approaches such as the Deep BSDE method [12–15] and physics-informed neural networks (PINNs) [16]. Neural network-based approximations excel in breaking the curse of dimensionality and efficiently approximating conditional expectations and gradients. Deep-learning based methods have demonstrated strong potential for high-dimensional PDE and BSDE problems [17,18]. The convergence of the Deep BSDE method has been rigorously analyzed for fully coupled FBSDEs, including cases with non-Lipschitz coefficients [19,20].

In the context of derivative pricing, BSDE representations are particularly useful for handling nonlinear payoffs, early exercise features, and path-dependent claims. By discretizing the backward system in time and using neural networks to approximate the solution and its gradient at each time step, conditional expectations can be learned efficiently in high dimensions [21]. Existing Deep BSDE pricing studies differ substantially in their problem settings, network architectures, and benchmark assets [22–27]. Few of these Deep BSDE pricing works systematically examine the linkage between BSDE formulations and risk-neutral PDE representations, nor do they quantify the performance gaps against conventional numerical solvers under identical high-dimensional pricing settings.

While the existing methodological streams have substantially advanced option pricing research, several research gaps remain. First, most studies treat PDE-based and BSDE-based approaches independently, despite the theoretical connections established in foundational works, such as [9,28]. Second, the transformation from real-world dynamics to a risk-neutral measure—although standard in theory—is often stated without rigorous derivation, particularly in multi-asset settings with correlated Brownian motions [29,30]. Third, systematic numerical benchmarks, especially Monte Carlo comparisons against theoretical PDE- or BSDE-based pricing results, are frequently absent, as highlighted in [6] and subsequent BSDE/PDE deep learning studies. Finally, while deep learning methods have been applied to high-dimensional option pricing, their integration into a unified framework alongside classical closed-form solutions (e.g., geometric basket options) and risk-neutral benchmarks remains largely unexplored [23,31,32].

Motivated by these gaps, this paper develops a theoretical and numerical framework for European option pricing under the multivariate Black–Scholes model. Our contributions include both theoretical developments and computational innovations:

1. Theoretically, we complete the derivation of the real-world and risk-neutral dynamics, the explicit transformation using the market price of risk, the BSDE–PDE correspondence, and the identification of the risk-neutral driver, and formulate basket and maximum option pricing problems within this unified measure-consistent framework.
2. Computationally, we develop a tailored Deep BSDE solver; the proposed implementation approximates benchmark prices with good accuracy across the considered examples, including one-dimensional European call and put options and multi-dimensional (5-, 10-, 20-, and 100-dimensional) basket and max options, especially for the 100-dimensional case.

The contributions of this study can be articulated along three major dimensions. First, we rigorously derive the real-world and risk-neutral dynamics for multi-asset systems. Second, we establish a unified BSDE–PDE formulation that yields an explicit representation of the driver function. In contrast to classical treatments, which typically analyze PDE-based and BSDE-based methods separately, our framework reveals that the closed-form results (e.g., geometric basket options) and numerical schemes (e.g., arithmetic basket and max options) arise naturally as special cases of a common structure. This unification extends beyond the formulations considered in [9] and the numerical implementations in [12,13,27]. Third, we present comprehensive Monte Carlo-based benchmarks that validate both the analytical derivations and the resulting pricing algorithms.

The remainder of this paper is organized as follows. Section 2 introduces the multivariate Black–Scholes model, defines the payoff structures, and derives both the real-world and risk-neutral dynamics, including the generator f. Section 3 presents the numerical BSDE discretization, neural network approximation, and optimization framework. Section 4 provides numerical examples, including comparisons with Monte Carlo and analytic (if available) benchmarks in both one- and multi-dimensional settings. Section 5 concludes the paper and discusses potential avenues for future research.

2. Model Construction

2.1. Multivariate Black–Scholes Dynamics and BSDE

We consider a filtered probability space $(\mathbf{\Omega}, \mathcal{F}, \{\mathcal{F}_t\}_{t\geq 0}, \mathbb{P})$ supporting a $\mathbf{d}$-dimensional Brownian motion $\mathbf{W_t^{\mathbb{P}}} = \left(\mathbf{W_t^{1,\mathbb{P}}}, \ldots, \mathbf{W_t^{d,\mathbb{P}}}\right)^\top$. Let $\mathbf{S_t} = \left(\mathbf{S_t^1}, \ldots, \mathbf{S_t^d}\right)^\top$ denote the vector of underlying asset prices. We adopt the multivariate Black–Scholes dynamics in the real-world measure $\mathbb{P}$:

$$\mathbf{dS_t} = \mathbf{diag}(\mathbf{S_t})\left(\boldsymbol{\mu}(\mathbf{t}, \mathbf{S_t})\,\mathbf{dt} + \boldsymbol{\sigma}(\mathbf{t}, \mathbf{S_t})\,\mathbf{dW_t^{\mathbb{P}}}\right), \tag{1}$$

where $\boldsymbol{\mu}(\mathbf{t}, \mathbf{S_t}) \in \mathbb{R}^{\mathbf{d}}$ is the drift vector (physical drift), $\boldsymbol{\sigma}(\mathbf{t}, \mathbf{S_t}) \in \mathbb{R}^{\mathbf{d}\times\mathbf{d}}$ is the volatility matrix (assumed non-singular), and $\mathbf{diag}(\mathbf{S_t})$ denotes the $\mathbf{d} \times \mathbf{d}$ diagonal matrix with entries $\mathbf{S_t^i}$.

Let $\mathbf{g} : \mathbb{R}^{\mathbf{d}} \to \mathbb{R}$ be a Borel-measurable payoff function payable at time $\mathbf{T}$. We consider the backward stochastic differential equation (BSDE) for $(\mathbf{Y_t},\mathbf{Z_t})$:

$$\begin{aligned} \mathbf{dY_t} = -\mathbf{f}_{\mathbb{P}}(\mathbf{t}, \mathbf{S_t}, \mathbf{Y_t}, \mathbf{Z_t})\,\mathbf{dt} + \mathbf{Z_t^\top}\mathbf{dW_t^{\mathbb{P}}}, \qquad 0 \le \mathbf{t} \le \mathbf{T}, \\ \mathbf{Y_T} = \mathbf{g}(\mathbf{S_T}), \end{aligned} \tag{2}$$

where $\mathbf{Z_t} \in \mathbb{R}^{\mathbf{d}}$ is the control (hedging) process and $\mathbf{f}_{\mathbb{P}}$ is the driver under $\mathbb{P}$. In a complete market with no arbitrage, the price process $\mathbf{V}(\mathbf{t},\mathbf{S_t}) = \mathbf{Y_t}$ is connected to the PDE solved by the pricing function (see Section 2.4).

2.2. Payoff Definitions: Basket, Max

We define three commonly studied multi-asset European payoffs. The basket call falls into one of two categories: arithmetic or geometric basket call.

1. Arithmetic basket call: Let $\mathbf{w} = (\mathbf{w}_1,\ldots,\mathbf{w_d})^\top$ be nonnegative weights with $\sum_{\mathbf{i}=1}^{\mathbf{d}} \mathbf{w_i} = \mathbf{1}$. The arithmetic basket call payoff is

$$\mathbf{g_{ari}}(\mathbf{S_T}) = (\sum_{\mathbf{i}=1}^{\mathbf{d}} \mathbf{w_i}\mathbf{S_T^i} - \mathbf{K})^+. \tag{3}$$

2. Geometric basket call: The geometric basket call payoff is

$$\mathbf{g_{geo}}(\mathbf{S_T}) = (\prod_{\mathbf{i}=1}^{\mathbf{d}} \mathbf{S_T^i})^{\mathbf{w_i}} - \mathbf{K}^+. \tag{4}$$

3. Max call: The max (or best-of) call payoff is given by

$$\mathbf{g_{max}}\,(\mathbf{S_T}) = (\max_{\mathbf{1}\le\mathbf{i}\le\mathbf{d}} \mathbf{S_T^i} - \mathbf{K})^+. \tag{5}$$

Note that the geometric basket admits a lognormal representation under multivariate lognormal dynamics and thus often has a closed-form pricing formula; the other two payoffs generally require numerical evaluation.

2.3. Real-World Measure $\mathbb{P}$ Versus Risk-Neutral Measure $\mathbb{Q}$

We now present the measure transformation from $\mathbb{P}$ to $\mathbb{Q}$ in a multivariate setting, introduce the market price of risk vector, and state a theorem showing equivalence of initial prices for European payoffs.

Assume a constant (or state-dependent) risk-free rate $\mathbf{r} \in \mathbb{R}$. For market consistency (no-arbitrage), there exists a market price of risk vector $\boldsymbol{\lambda}(\mathbf{t}, \mathbf{S_t}) \in \mathbb{R}^{\mathbf{d}}$ such that the Radon–Nikodym derivative

$$\frac{\mathbf{d}\mathbb{Q}}{\mathbf{d}\mathbb{P}}|_{\mathcal{F}_\mathbf{T}} = \mathbf{exp}\left(-\int_0^{\mathbf{T}} \boldsymbol{\lambda}(\mathbf{s}, \mathbf{S_s})^\top \mathbf{dW_s^{\mathbb{P}}} - \frac{1}{2}\int_0^{\mathbf{T}} \|\boldsymbol{\lambda}(\mathbf{s}, \mathbf{S_s})\|^2\,\mathbf{ds}\right) \tag{6}$$

defines an equivalent martingale measure $\mathbb{Q}$ (Girsanov). Under $\mathbb{Q}$, the process

$$\mathbf{W_t^{\mathbb{Q}}} := \mathbf{W_t^{\mathbb{P}}} + \int_0^{\mathbf{t}} \boldsymbol{\lambda}(\mathbf{s}, \mathbf{S_s})\,\mathbf{ds} \tag{7}$$

is a Brownian motion.

Under $\mathbb{Q}$, we obtain

$$\mathbf{dS_t} = \mathbf{diag}(\mathbf{S_t})\left(\tilde{\boldsymbol{\mu}}(\mathbf{t}, \mathbf{S_t})\,\mathbf{dt} + \boldsymbol{\sigma}(\mathbf{t}, \mathbf{S_t})\,\mathbf{dW_t^{\mathbb{Q}}}\right), \tag{8}$$

where

$$\tilde{\boldsymbol{\mu}}(\mathbf{t}, \mathbf{S_t}) = \boldsymbol{\mu}(\mathbf{t}, \mathbf{S_t}) - \boldsymbol{\sigma}(\mathbf{t}, \mathbf{S_t})\,\boldsymbol{\lambda}(\mathbf{t}, \mathbf{S_t}). \tag{9}$$

Absence of arbitrage requires $\tilde{\boldsymbol{\mu}}(\mathbf{t}, \mathbf{S_t}) = \mathbf{r1}$ (the vector with all components equal to $\mathbf{r}$), which determines the market price of risk:

$$\boldsymbol{\lambda}(\mathbf{t}, \mathbf{S_t}) = \boldsymbol{\sigma}(\mathbf{t}, \mathbf{S_t})^{-1}(\boldsymbol{\mu}(\mathbf{t}, \mathbf{S_t}) - \mathbf{r1}), \tag{10}$$

assuming $\boldsymbol{\sigma}$ is invertible.

Under $\mathbb{P}$, the BSDE is (2) with driver $\mathbf{f}_{\mathbb{P}}$. Under $\mathbb{Q}$, the BSDE for the same terminal payoff $\mathbf{g}(\mathbf{S_T})$ reads

$$\mathbf{d}\tilde{\mathbf{Y}}_\mathbf{t} = -\mathbf{f}_{\mathbb{Q}}(\mathbf{t}, \mathbf{S_t}, \tilde{\mathbf{Y}}_\mathbf{t}, \tilde{\mathbf{Z}}_\mathbf{t})\,\mathbf{dt} + \tilde{\mathbf{Z}}_\mathbf{t}^{\top}\,\mathbf{dW}_\mathbf{t}^{\mathbb{Q}}, \qquad \tilde{\mathbf{Y}}_\mathbf{T} = \mathbf{g}(\mathbf{S_T}). \tag{11}$$

We will show that, in the usual linear (European) pricing setup, the two BSDEs imply the same initial value.

Theorem 1. *Equivalence of initial price under $\mathbb{P}$ and $\mathbb{Q}$. Let the market be complete and $\boldsymbol{\sigma}(t,S_t)$ invertible. Let $g(S_T)$ be a European payoff. Denote (Y_t,Z_t) as the solution under $\mathbb{P}$-BSDE—$dY_t = -f_{\mathbb{P}}(t, S_t, Y_t, Z_t)dt + Z_t^{\top} dW_t^{\mathbb{P}}$, $Y_T = g(S_T)$—and $\left(\tilde{Y}_t, \tilde{Z}_t\right)$ the solution under $\mathbb{Q}$-BSDE—$d\tilde{Y}_t = -f_{\mathbb{Q}}\left(t, S_t, \tilde{Y}_t, \tilde{Z}_t\right)dt + \tilde{Z}_t^{\top} dW_t^{\mathbb{Q}}$, $\tilde{Y}_T = g(S_T)$. If $f_{\mathbb{P}}(t, S, Y, Z) = f_{\mathbb{Q}}(t, S, Y, Z) - \boldsymbol{\lambda}(t, S)^{\top} Z$, then $Y_t = \tilde{Y}_t$ for all $t \in [0,T]$.*

Proof of Theorem 1. Starting from $\mathbb{P}$-BSDE,

$$\mathbf{dY_t} = -\mathbf{f}_{\mathbb{P}}(\mathbf{t}, \mathbf{S_t}, \mathbf{Y_t}, \mathbf{Z_t})\,\mathbf{dt} + \mathbf{Z_t}^{\top}\,\mathbf{dW_t^{\mathbb{P}}}.$$

Substituting $\mathbf{dW_t^{\mathbb{P}}} = \mathbf{dW_t^{\mathbb{Q}}} - \boldsymbol{\lambda}(\mathbf{t}, \mathbf{S_t})\,\mathbf{dt}$,

$$\mathbf{dY_t} = -\mathbf{f}_{\mathbb{P}}(\mathbf{t}, \mathbf{S_t}, \mathbf{Y_t}, \mathbf{Z_t})\,\mathbf{dt} + \mathbf{Z_t}^{\top}\left(\mathbf{dW_t^{\mathbb{Q}}} - \boldsymbol{\lambda}(\mathbf{t}, \mathbf{S_t})\mathbf{dt}\right) = -\left[\mathbf{f}_{\mathbb{P}} + \boldsymbol{\lambda}^{\top}\mathbf{Z_t}\right]\mathbf{dt} + \mathbf{Z_t}^{\top}\mathbf{dW_t^{\mathbb{Q}}}.$$

Using $\mathbf{f}_{\mathbb{P}} = \mathbf{f}_{\mathbb{Q}} - \boldsymbol{\lambda}^{\top}\mathbf{Z}$,

$$\mathbf{dY_t} = -\mathbf{f}_{\mathbb{Q}}(\mathbf{t}, \mathbf{S_t}, \mathbf{Y_t}, \mathbf{Z_t})\,\mathbf{dt} + \mathbf{Z_t}^{\top}\,\mathbf{dW_t^{\mathbb{Q}}}.$$

By uniqueness of BSDE solutions, $\mathbf{Y_t} = \tilde{\mathbf{Y}}_\mathbf{t}$ a.s. for all $\mathbf{t}$. □

2.4. Deriving the Driver f Under the Risk-Neutral Measure and the BSDE–PDE Correspondence

Let $\mathbf{Y_t}$ denote the (undiscounted) price process of a European contingent claim with terminal payoff $\mathbf{g}(\mathbf{S_T})$. Under our BSDE sign convention, the risk-neutral BSDE takes the form

$$\mathbf{dY_t} = -\mathbf{f}_{\mathbb{Q}}(\mathbf{t}, \mathbf{S_t}, \mathbf{Y_t}, \mathbf{Z_t})\,\mathbf{dt} + \mathbf{Z_t}^{\top}\,\mathbf{dW_t^{\mathbb{Q}}}, \qquad \mathbf{Y_T} = \mathbf{g}(\mathbf{S_T}). \tag{12}$$

Define the discounted price process $\bar{\mathbf{Y}}_\mathbf{t} := \mathbf{e^{-rt}Y_t}$. Using Itô's formula and (12), we have

$$\begin{aligned}
\mathbf{d\bar{Y}_t} &= \mathbf{e^{-rt}}(\mathbf{dY_t} - \mathbf{rY_t dt}) \\
&= \mathbf{e^{-rt}}\left(-\mathbf{f}_{\mathbb{Q}}(\mathbf{t}, \mathbf{S_t}, \mathbf{Y_t}, \mathbf{Z_t})\,\mathbf{dt} + \mathbf{Z_t}^{\top}\mathbf{dW_t^{\mathbb{Q}}} - \mathbf{rY_t dt}\right) \\
&= \mathbf{e^{-rt}}\left(-\left(\mathbf{f}_{\mathbb{Q}}(\mathbf{t}, \mathbf{S_t}, \mathbf{Y_t}, \mathbf{Z_t}) + \mathbf{rY_t}\right)\mathbf{dt} + \mathbf{Z_t}^{\top}\mathbf{dW_t^{\mathbb{Q}}}\right).
\end{aligned}$$

By the risk-neutral pricing principle, the discounted price $\bar{\mathbf{Y}}_\mathbf{t}$ is a $\mathbb{Q}$-martingale, hence its drift must vanish. Therefore,

$$\mathbf{f}_{\mathbb{Q}}(\mathbf{t}, \mathbf{S_t}, \mathbf{Y_t}, \mathbf{Z_t}) + \mathbf{rY_t} = 0,$$

which yields the identification

$$\mathbf{f}_{\mathbb{Q}}(\mathbf{t}, \mathbf{S}, \mathbf{Y}, \mathbf{Z}) = -\,\mathbf{r}\,\mathbf{Y}. \tag{13}$$

Proofs of this main result can be seen in Appendix A.

Assume the pricing function is sufficiently smooth and that

$$\mathbf{Y_t} = \mathbf{V}(\mathbf{t}, \mathbf{S_t}), \qquad \mathbf{Z_t} = \boldsymbol{\sigma}^\top \mathbf{diag}(\mathbf{S_t})\, \nabla_{\mathbf{S}} \mathbf{V}(\mathbf{t}, \mathbf{S_t}).$$

It is worth noting that the definition of $\mathbf{Z_t}$ differs from those in the Deep BSDE literature. Specifically, [13,27] define $\mathbf{Z_t}$ simply as $\nabla_{\mathbf{S}} \mathbf{V}(\mathbf{t},\mathbf{S_t})$, thereby omitting the scaling induced by the asset prices and the volatility structure. In contrast, our definition of $\mathbf{Z_t}$, the same as that in [33], is fully consistent with the classical BSDE representation derived from Itô's formula and directly reflects the transmission of Brownian shocks through the volatility matrix and the asset price levels.

From a financial perspective, $\boldsymbol{\nabla_S} \mathbf{V}(\mathbf{t},\mathbf{S_t})$ represents the classical delta hedging ratios with respect to the underlying assets, whereas $\mathbf{Z_t}$ measures the exposure of the option value to the underlying risk factors driving the Brownian motion. This distinction becomes particularly important in high-dimensional multi-asset settings, where the interaction between heterogeneous volatilities, cross-asset correlations, and asset price scales plays a critical role. Incorporating the factor $\boldsymbol{\sigma}^\top \mathbf{diag}(\mathbf{S_t})$ ensures the correct propagation of uncertainties in the stochastic integrals, leading to improved numerical stability and enhanced interpretability of the learned control process.

Under the physical measure $\mathbb{P}$, the BSDE driver generally includes a risk premium term, and one can write (for a suitable market price of risk vector $\boldsymbol{\lambda}$)

$$\mathbf{f}_{\mathbb{P}}(\mathbf{t}, \mathbf{S}, \mathbf{Y}, \mathbf{Z}) = \mathbf{f}_{\mathbb{Q}}(\mathbf{t}, \mathbf{S}, \mathbf{Y}, \mathbf{Z}) - \boldsymbol{\lambda}\big(\mathbf{t}, \mathbf{S}\big)^\top \mathbf{Z} = -\mathbf{r}\mathbf{Y} - \boldsymbol{\lambda}\big(\mathbf{t}, \mathbf{S}\big)^\top \mathbf{Z}.$$

Equivalently,

$$\mathbf{dY_t} = -\Big(-\mathbf{r}\mathbf{Y_t} - \boldsymbol{\lambda}^\top \mathbf{Z_t}\Big)\, \mathbf{dt} + \mathbf{Z_t^\top} \mathbf{dW_t^{\mathbb{P}}}.$$

After Girsanov transformation of $\mathbf{W}^{\mathbb{P}}$ to $\mathbf{W}^{\mathbb{Q}} = \mathbf{W}^{\mathbb{P}} + \int \boldsymbol{\lambda} \mathbf{dt}$, the extra $\boldsymbol{\lambda}^\top \mathbf{Z}$ term is absorbed by the measure change, and one recovers the risk-neutral BSDE with driver $-\mathbf{r}\mathbf{Y}$.

3. Deep BSDE Solver

This section describes the deep learning-based solver for the forward–backward stochastic differential equations (FBSDEs) used in this paper. The implementation follows the Deep BSDE paradigm [13,27] and is carried out in conda 23.3.1, python 3.7.11, pytorch 1.10.2. A key distinction from those works is that they employ TensorFlow as the computational framework, whereas our implementation uses PyTorch. In addition, our neural network architecture differs in several aspects (cf. [19]), including the design of hidden layers, activation functions, and training procedures, all of which are adapted to enhance efficiency in high-dimensional multi-asset settings. We present the mathematical discretization and show how it maps to the code structure provided in the previous section.

We discretize $[0,T]$ uniformly with N steps, $\Delta t = T/N$, and denote $S_n \approx S_{t_n}$, $Y_n \approx Y_{t_n}$, $Z_n \approx Z_{t_n}$. The Deep BSDE solver proceeds by:

1. Parameterizing the unknown processes $\left(Y_0, Z_{t_0}, \ldots, Z_{t_{N-1}}\right)$ using neural networks. In practice, a feed-forward network ϕ_θ is used to produce Z_n as a function of (t_n,S_n), $Z_n = \phi_\theta(t_n,\mathrm{S}_n)$.
2. Employing a fully connected feed-forward network ϕ_θ. The input layer has dimension $\dim_s + 1$, and the network contains three hidden layers, each with $\dim_h$ neurons activated by the hyperbolic tangent function. The hyperbolic tangent is adopted here owing to its zero-centered and globally smooth features: its zero-centering stabilizes gradient flow in the backward BSDE recursion, while its smoothness avoids numerical artifacts caused by non-smooth points in activations such as ReLU when approximating

the BSDE solution and its gradient process. Denoting the input as s, the hidden layers are computed as

$$h_1 = tanh(W_1 s + b_1), \quad h_2 = tanh(W_2 h_1 + b_2), \quad h_3 = tanh(W_3 h_2 + b_3),$$

and the output layer maps h_3 linearly to $\dim_y \times \dim_d$ and reshapes it to match the Z_t tensor. W_1, b_1, W_2, b_2, W_3, and b_3 denote the parameters of the neural network. This architecture provides a flexible nonlinear approximation for high-dimensional FBSDEs.

3. Simulating the sample paths of the forward SDE using Monte Carlo.
4. Propagating S_n and Y_n forward in time using the discretized dynamics

$$S_{n+1} = S_n + \mu(t_n, S_n, Y_n)\Delta t + \sigma(t_n, S_n)\Delta W_n,$$
$$Y_{n+1} = Y_n - f(t_n, S_n, Y_n, Z_n)\Delta t + Z_n^\top \Delta W_n,$$

where ΔW_n is the Brownian increment.

5. Comparing Y_N with the terminal condition $g(S_N)$ and minimizing the squared error over Monte Carlo samples to learn θ and the initial parameter Y_0.
6. Performs training using Adam. Training minimizes

$$\mathcal{L}(\theta, Y_0) = \mathbb{E}\left[\| Y_N^\theta - g(S_N) \|^2\right],$$

approximated by the batch empirical MSE. We adopt the Adam optimizer with PyTorch default hyperparameters: learning rate $\alpha = 0.001$, moment decay coefficients $\beta_1 = 0.9$ and $\beta_2 = 0.999$, and stability constant $\epsilon = 10^{-8}$.

4. Numerical Examples

In this section we present numerical experiments that validate the proposed Deep BSDE solver against classical benchmarks. We consider two families of European options: one-dimensional vanilla calls and puts (Section 4.1) and multi-dimensional options, including arithmetic basket calls, geometric basket calls and max (best-of) calls (Section 4.2). For each experiment we describe the parameter settings, implementation details, and numerical results, including a comparison with Monte Carlo (MC) and analytic benchmarks.

4.1. One-Dimensional European Options

We consider a single underlying asset St. The terminal payoffs of the European call and put with strike K and maturity T are

$$\boldsymbol{\Phi_{call}}(\boldsymbol{S_T}) = \left(\boldsymbol{S_T} - \boldsymbol{K}\right)^+, \qquad \boldsymbol{\Phi_{put}}(\boldsymbol{S_T}) = (\boldsymbol{K} - \boldsymbol{S_T})^+.$$

4.1.1. Analytic Benchmark (Black–Scholes Formula)

For the one-dimensional case, the closed-form Black–Scholes formulas provide the ground truth:

$$C_{BS}(S_0, K, T, r, \sigma) = S_0\Phi(d_1) - Ke^{-rT}\Phi(d_2),$$
$$P_{BS}(S_0, K, T, r, \sigma) = Ke^{-rT}\Phi(-d_2) - S_0\Phi(-d_1),$$

where

$$d_{1,2} = \frac{ln(S_0/K) + \left(r \pm 1/2\sigma^2\right)T}{\sigma\sqrt{T}},$$

and $\Phi(\cdot)$ denotes the standard normal CDF.

4.1.2. Experimental Setup

We train the Deep BSDE solver as described in Section 3 to estimate Y_0. The MC benchmark is computed under $\mathbb{Q}$ with the standard Euler sampling of the lognormal terminal distribution.

The parameters are: risk-free rate $r = 0.03$; drift $\mu \in \{0.04, 0.03, 0.02\}$; volatility $\sigma = 0.2$}; initial price $S_0 = 100$; strike $K \in \{95, 100, 105\}$; and maturities $T = 1.0\}$ (years). The deep BSDE hyperparameters are: time steps $N = 50$; batch size $M_{\text{batch}} = 64$; iterations = 8000; and hidden size $\dim_h = 32$. The MC benchmark is: number of paths $M = 1 \times 10^5$ and number of time steps $Nsteps = 50$.

4.1.3. Numerical Results

Table 1 reports the BS analytic price Y_0^{BS}, the MC estimate $\hat{Y}_0^{MC,\mathbb{Q}}$ with 95% confidence interval, and the Deep BSDE estimates $Y_0^{DB,\mathbb{Q}}$ and $Y_0^{DB,\mathbb{P}}$. We design a test to verify the theoretical invariance property: we vary the physical drift μ under the $\mathbb{P}$ measure and set the corresponding market price of risk λ consistently. For each μ, we solve the BSDE twice: (i) under the physical measure $\mathbb{P}$ with drift μ and calibrated λ; (ii) directly under the risk-neutral measure $\mathbb{Q}$. The BS price is invariant with respect to the physical drift μ. Our Deep BSDE estimates are close to both the BS and MC benchmarks, and the measured $|Y_0^{DB,\mathbb{Q}} - Y_0^{DB,\mathbb{P}}|$ stays at the level of numerical error without dependence on μ.

Table 1. One-dimensional European option results.

Option	K	μ	λ	Y_0^{BS}	$\hat{Y}_0^{MC,\mathbb{Q}}$ (CIs)	$Y_0^{DB,\mathbb{Q}}$	$Y_0^{DB,\mathbb{P}}$	$\lvert Y_0^{DB,\mathbb{Q}} - Y_0^{DB,\mathbb{P}} \rvert$
Call	95	0.04	0.05	12.1797	12.2094 (12.1122, 12.3066)	12.1856	12.1755	0.0101
		0.03	0				12.1825	0.0031
		0.02	−0.05				12.1752	0.0104
	100	0.04	0.05	9.4134	9.4099 (9.3220, 9.4978)	9.4251	9.4214	0.0037
		0.03	0				9.4178	0.0073
		0.02	−0.05				9.4086	0.0165
	105	0.04	0.05	7.1281	7.1533 (7.0753, 7.2313)	7.1136	7.1200	0.0064
		0.03	0				7.1153	0.0017
		0.02	−0.05				7.1165	0.0029
Put	95	0.04	0.05	4.372	4.3641 (4.3170, 4.4111)	4.3762	4.3746	0.0016
		0.03	0				4.3771	0.0009
		0.02	−0.05				4.3843	0.0081
	100	0.04	0.05	6.458	6.4754 (6.4174, 6.5334)	6.4448	6.4728	0.028
		0.03	0				6.4499	0.0051
		0.02	−0.05				6.4190	0.0258
	105	0.04	0.05	9.0248	9.0167 (8.9481, 9.0854)	9.0294	9.0287	0.0007
		0.03	0				9.0309	0.0015
		0.02	−0.05				9.0190	0.0104

Note: Confidence intervals (CIs) are constructed at the 95% confidence level, consistent with standard practice in the literature.

For example, when the strike price of a call option is 100, the BS analytical solution is 9.4134. With different values of the drift term—namely $\mu = 0.02$, 0.03, and 0.04—the simulated value from the MC method is 9.4099, while the solutions obtained by the Deep BSDE method are 9.4086, 9.4178, and 9.4214 under the P measure and 9.4251 under the Q measure, which are consistent with the analytical solution. The pairwise absolute differences between the P-based and Q-based Deep BSDE solutions are small and compatible with neural discretization and sampling error (as shown in Table 2).

Table 2. Sensitivity analysis with respect to selected hyperparameters.

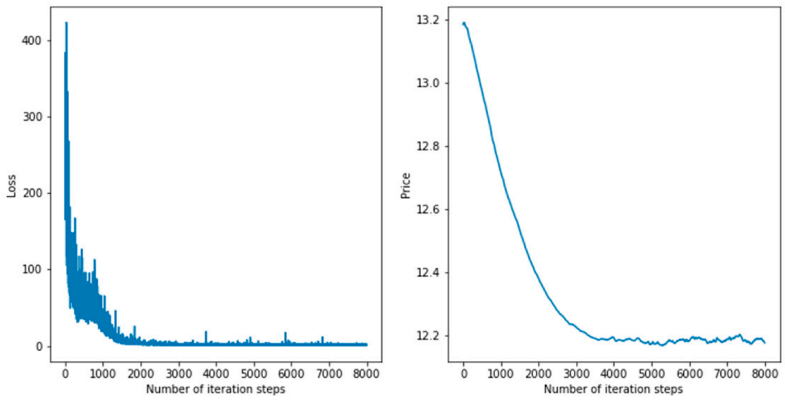
Parameter	Tested Values	Baseline	Mean of Deep BSDE Price	Std of Deep BSDE Price
Number of time steps N	10, 20, 30, 50, 100	50	9.4184, 9.4112, 9.4161, 9.420, 9.4150	0.0040, 0.0068, 0.0090, 0.0034, 0.0043
Batch size	32, 64, 128, 256	64	9.4107, 9.4197, 9.4169, 9.4129	0.0137, 0.0034, 0.0028, 0.0011
Optimization iterations	1000, 3000, 5000, 8000, 10000	8000	8.8667, 9.3826, 9.4197, 9.420	0.0159, 0.0043, 0.0029, 0.0034
Hidden dimension	16, 32, 48, 64	32	9.4204, 9.4197, 9.4221, 9.4219	0.0035, 0.0034, 0.0027, 0.0028
Network depth	2, 3, 4, 5	2	9.4218, 9.4198, 9.4198, 9.4201	0.0027, 0.0034, 0.0046, 0.0083
Learning rate	0.0001, 0.0005, 0.001, 0.005	0.001	8.8074, 9.4172, 9.4218, 9.4164	0.0052, 0.0026, 0.0027, 0.0475

Note: The computed call option prices (S = K = 100; μ = r = 0.03) are generated under the risk-neutral measure. Each reported price corresponds to the mean or standard deviation obtained across five distinct random seeds (0, 1, 2, 3, and 4).

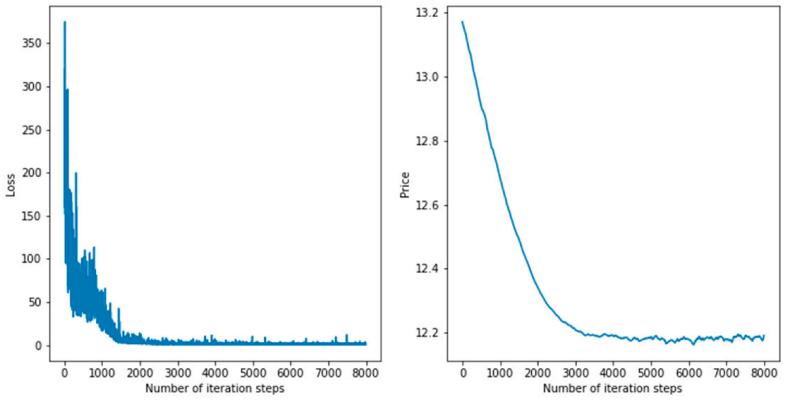
Figure 1 illustrates the loss trajectories and the convergence behavior of $Y_0^{DB,\mathbb{P}}$ caculated using the deep learning method for $K = 95$ and three drift values: $\mu = 0.04$ (a), 0.03 (b), and 0.02 (c). The initial value of Y_0^{DB} is set to the Monte Carlo pricing result plus 1. As shown in the figure, the estimates converge to highly accurate values by approximately 4000 iterations.

Figure 2 illustrates the loss trajectories and the convergence behavior of Y_0^{DB} obtained using the deep learning method for the call option with $K = 100$ and $\mu = 0.03$, under three different initial values of Y for the BSDE forward process: the Monte Carlo pricing result plus 1 (a), the Monte Carlo result itself (b), and the Monte Carlo result minus 1 (c). It can be observed that the deep learning method converges to a stable solution regardless of the chosen initial value. Furthermore, when we directly set the initial value close to the analytical solution (as shown in the (b)-right figure, approximately 9.41), the solution obtained by the Deep BSDE method maintains small and stable fluctuations throughout.

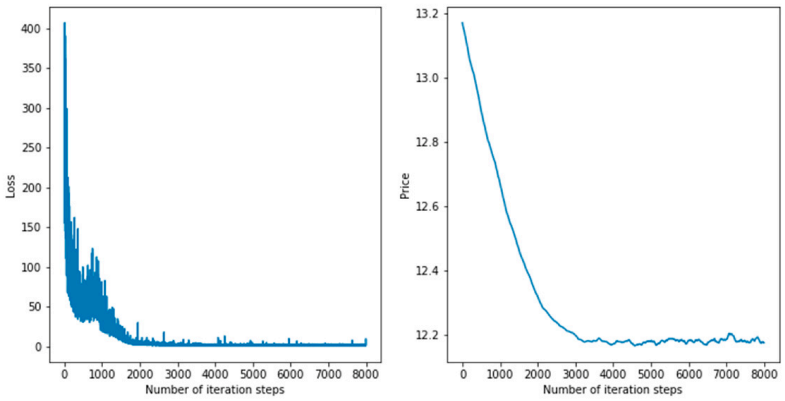
We perform a one-at-a-time sensitivity analysis to examine the sensitivity of the proposed Deep BSDE solver. We alter one hyperparameter each time while holding all other configurations fixed at the baseline values. For each hyperparameter setting, we repeat the experiment with five different random seeds (0, 1, 2, 3, and 4) to report the sample mean and standard deviation of the Deep BSDE estimated call option price with $S = K = 100$ and $\mu = r = 0.03$. The hyperparameters under investigation are the number of time steps N, batch size, number of optimization iterations, network hidden dimension, network depth, and learning rate. Table 2 summarizes the tested configurations and statistical results. The results across different random seeds demonstrate the stochastic nature of the Deep BSDE estimator; small standard deviations indicate reasonable stability under our baseline setting. Taken together, our baseline hyperparameters achieve a satisfactory balance between numerical accuracy and computational burden.



(**a**) $\mu = 0.04$, one-dimensional European call option.

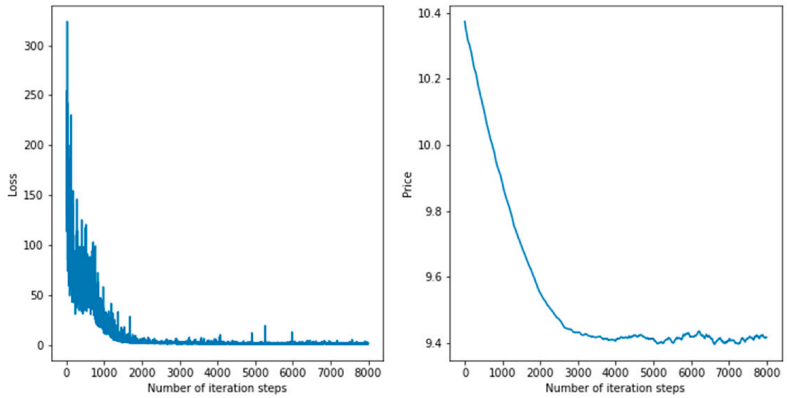


(**b**) $\mu = 0.03$, one-dimensional European call option.

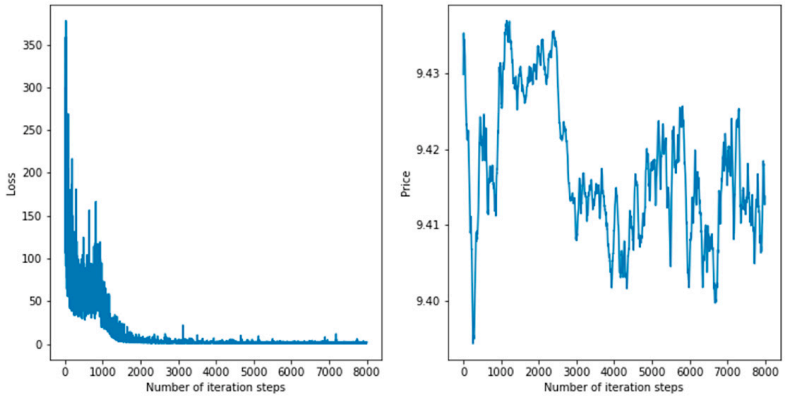


(**c**) $\mu = 0.02$, one-dimensional European call option.

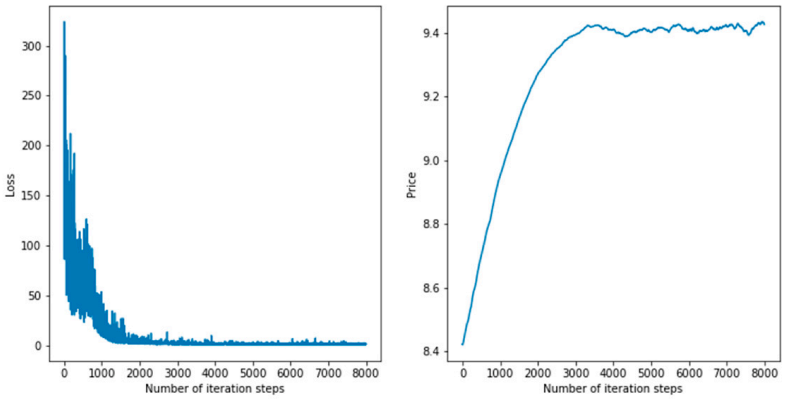
Figure 1. The loss trajectories and the convergence behavior of Y_0^{DB} for K = 95 and three drift values: $\mu = 0.04$ (**a**), 0.03 (**b**), and 0.02 (**c**).



(**a**) Initial values = the Monte Carlo pricing result plus 1.



(**b**) Initial values = the Monte Carlo pricing result.



(**c**) Initial values = the Monte Carlo pricing result minus 1.

Figure 2. The loss trajectories and the convergence behavior of Y_0^{DB} for call option with K = 100 and $\mu = 0.03$, under three different initial values: the Monte Carlo pricing result plus 1 (**a**), the Monte Carlo result itself (**b**), and the Monte Carlo result minus 1 (**c**).

The average computation time for the deep learning method was approximately 350 s. All experiments were conducted on a computer equipped with an Intel Core i5-11320H processor (4 physical cores, 8 logical cores, base frequency 3.20 GHz, Dell Inc., Shijiazhuang, China). The system had 16 GB of RAM (15.74 GB available) and ran Windows 10. The software environment included Python 3.7.11, PyTorch 1.10.2, and NumPy 1.21.5. Note that no GPU acceleration was used in this experiment; all computations were performed on the

CPU. All reported runtime values correspond to the wall-clock time, measured from the start to the completion of each numerical experiment.

4.2. Multi-Dimensional European Options

We price three payoff types: arithmetic basket call, max call, and geometric basket call.

4.2.1. Parameter Settings

In our high-dimensional European option pricing experiments, we adopt a simplified correlation structure in which the pairwise asset correlation ρ is taken to be constant across all assets. This choice allows us to control the complexity of the covariance matrix and to focus on the performance and stability of the pricing algorithms across different dimensions. All subsequent option pricing is conducted under the risk-neutral measure. The parameters are set as follows:

Dimension $d \in \{5,10,20,100\}$; strike price $K = 100$; basket weight vector has uniform weights $w_i = 1/d$; maturity $T = 1.0$; and $r = 0.03$.

For the initial prices, the $5-$dimensional $S_0 = [100.0,105.0,95.0,102.0,98.0]$, $10-$dimensional $S_0 = [100.0,105.0,95.0,102.0,98.0,103.0,97.0,101.0,99.0,104.0]$, $20-$dimensional $S_0 = [100.0, 105.0, 95.0, 102.0, 98.0, 103.0, 97.0, 101.0, 99.0, 104.0, 96.0, 107.0, 93.0, 100.5, 99.5, 101.5, 98.5, 103.5, 96.5, 104.5]$,

And the 100-dimensional S_0 is evenly spaced over $[95, 95 + 10/(d-1), \ldots, 105]$, i.e., $S_0 = [95, 95 + 10/(d-1), \ldots, 105]$.

For the volatility and correlation structure,

$$5-\text{dimensional } \boldsymbol{\sigma} = [\mathbf{0.20, 0.25, 0.18, 0.22, 0.19}], \quad \boldsymbol{\rho} = \mathbf{0.30};$$

$$10-\text{dimensional} \quad \sigma = [0.20, 0.25, 0.18, 0.22, 0.19, 0.21, 0.23, 0.17, 0.24, 0.16], \ \rho = 0.25;$$

$$20-\text{dimensional } \sigma = [0.20, 0.25, 0.18, 0.22, 0.19, 0.21, 0.23, 0.17, 0.24, 0.16, 0.26, 0.15, 0.27, 0.14, 0.28, 0.13, 0.29, 0.12, 0.30, 0.11], \quad \rho = 0.20; \text{ and}$$

$$100-\text{dimensional } \sigma = [\mathbf{0.1, 0.1 + 0.2/d - 1, \ldots, 0.3}], \quad \boldsymbol{\rho} = \mathbf{0.05}.$$

For the Deep BSDE hyperparameters, $N = 50$, batch size $= 64$, iterations $= 8000$, and hidden dimension $\dim_h = 32$. For the Monte Carlo benchmark, $M = 1 \times 10^5$ sample paths for $d = 5$, $M = 2 \times 10^5$ for $d = 10$, $M = 4 \times 10^5$ for $d = 20$, and $M = 1 \times 10^6$ for $d = 100$; the number of time steps is set to $Nsteps = 50$, identical to the Deep BSDE baseline to ensure a fair comparison.

4.2.2. Numerical Results

Table 3 presents a comparative analysis of three option types (arithmetic, max, and geometric) across the different dimensions (5D, 10D, 20D, and 100D). The Monte Carlo method serves as the benchmark, providing price estimates along with 95% confidence intervals. The Deep BSDE method provides results obtained through forward–backward stochastic differential equation solving with a deep learning solver. For the geometric average options, the analytical solutions are provided as reference values. All computation times are measured in seconds.

Table 3. Comparison of option pricing results for different methods and dimensions.

Dim (Paths)	Monte Carlo (CIs)	Time(s)	Deep BSDE	Time(s)	Analytical
Arithmetic					
5D (100,000)	7.0644 [7.0214, 7.1073]	3.04	7.0482	376.76	-
10D (200,000)	6.4694 [6.4390, 6.4997]	9.54	6.4723	374.59	-
20D (400,000)	5.7320 [5.7093, 5.7546]	21.03	5.7346	375.53	-
100D (1,000,000)	3.7738 [3.7661, 3.7815]	240.77	3.7728	1904.53	-
Max					
5D (100,000)	26.2652 [26.1352, 26.3951]	0.98	26.1410	372.1	-
10D (200,000)	34.7818 [34.6908, 34.8728]	4.03	34.7677	377.67	-
20D (400,000)	45.3687 [45.2964, 45.4409]	21.81	45.1725	375.6	-
100D (1,000,000)	80.1328 [80.0834, 80.1822]	247.35	79.7840	1895.23	-
Geometric					
5D (100,000)	6.2749 [6.2179, 6.3319]	1.48	6.2783	383.67	6.2702
10D (200,000)	5.5460 [5.5116, 5.5803]	6.14	5.5517	384.62	5.5645
20D (400,000)	4.6655 [4.6450, 4.6860]	20.36	4.6639	386.34	4.6637
100D (1,000,000)	2.3867 [2.3804, 2.3929]	240.83	2.3925	1764.03	2.3851

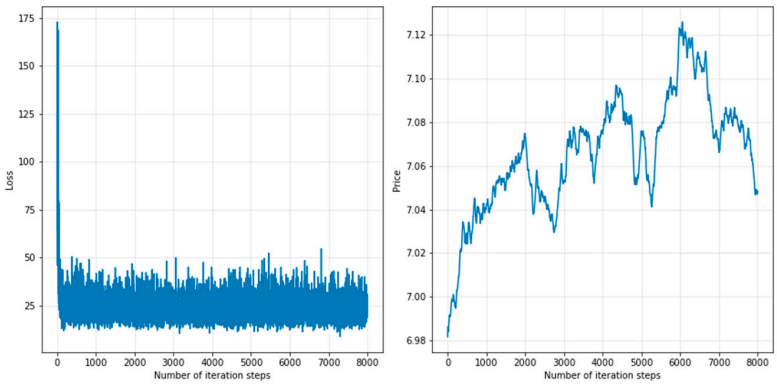
Note: Confidence intervals (CIs) are constructed at the 95% confidence level, consistent with standard practice in the literature. The entries in the column "Dim (Paths)" denote the option dimension and the total number of Monte Carlo sample paths. For instance, 5D (100,000) corresponds to a 5-dimensional asset with 100,000 simulated paths.

The results demonstrate good consistency between the Monte Carlo and Deep BSDE methods. For the arithmetic options, the relative price deviations between the two methods remain small across dimensions, with the largest relative difference of only 0.23% in the 5D case. For the max options, the relative difference reaches 0.44% at 100D. For the geometric options, the Deep BSDE method yields prices very close to the analytical values, with relative errors of 0.13% (5D), −0.23% (10D), 0.004% (20D), and 0.31% (100D). The Monte Carlo estimates also perform well, and all reported 95% confidence intervals contain the corresponding analytical prices.

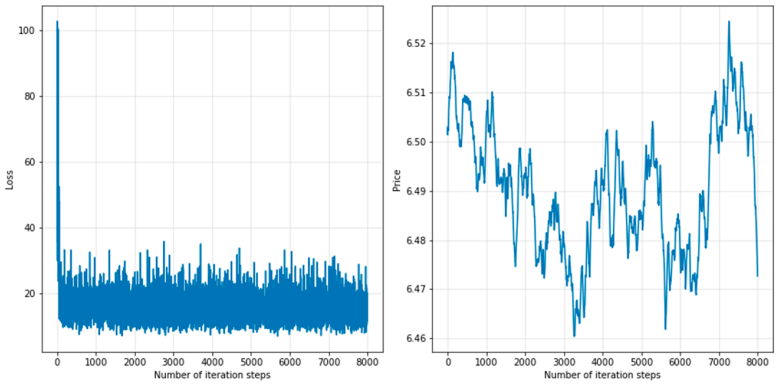
The close agreement between the Monte Carlo and Deep BSDE methods across all option types and dimensions suggests that deep learning approaches can effectively solve high-dimensional option pricing problems. However, the computational efficiency trade-offs should be considered: while Deep BSDE methods offer the advantage of solving the underlying differential equations directly, they require significant computational resources for training neural networks.

Figure 3 presents the loss trajectories and the convergence behavior of Y_0^{DB} for the arithmetic basket option across different dimensions: 5 dimensions (a), 10 dimensions (b), 20 dimensions (c), and 100 dimensions (d).

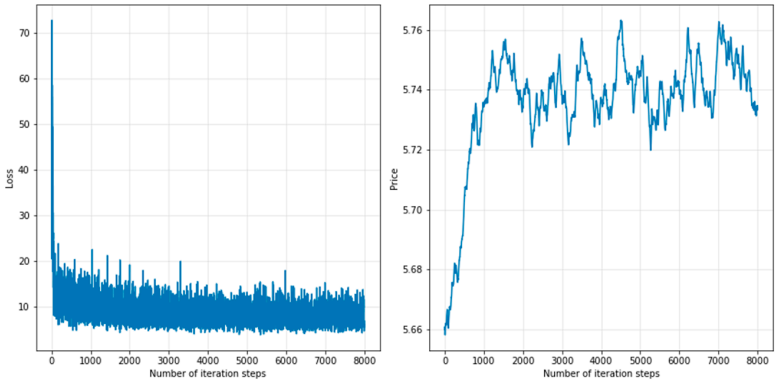
All numerical implementations in this paper are fully reproducible. The key computational procedures are summarized in the form of pseudocode, which can be found in Appendix B.



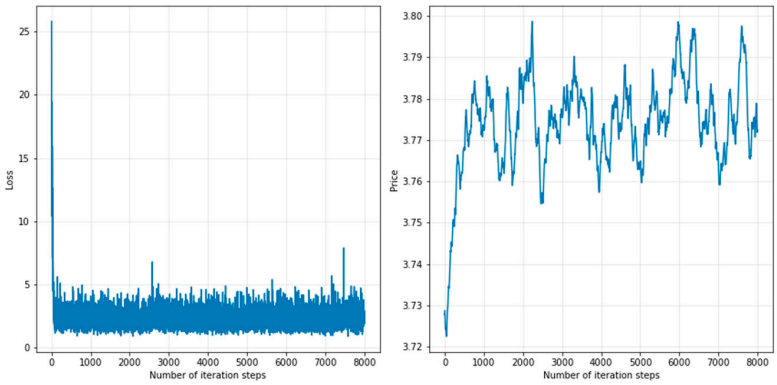
(**a**) The 5-dimensional arithmetic basket option.



(**b**) The 10-dimensional arithmetic basket option.



(**c**) The 20-dimensional arithmetic basket option.



(**d**) The 100-dimensional arithmetic basket option.

Figure 3. The loss trajectories and the convergence behavior of Y_0^{DB} for the arithmetic basket option across different dimensions: 5 dimensions (**a**), 10 dimensions (**b**), 20 dimensions (**c**), and 100 dimensions (**d**).

5. Conclusions and Future Work

This paper develops a unified BSDE-based framework for pricing European-style derivatives under both the real-world measure $\mathbb{P}$ and the risk-neutral measure $\mathbb{Q}$. We construct a Deep BSDE solver tailored to linear (pricing) BSDEs. The numerical discretization scheme, combined with neural network approximations for the control process Z_t, provides a flexible approach for solving high-dimensional pricing problems. The proposed method is evaluated on a suite of experiments, ranging from one-dimensional European calls and puts to multi-asset options.

The numerical results, which are cross-validated against the Monte Carlo and analytical solutions where available, demonstrate that our Deep BSDE method delivers accurate price estimates consistent with the Monte Carlo and analytical benchmarks. In terms of runtime, Monte Carlo is more efficient for the linear European examples considered up to 100 dimensions; nevertheless, the Deep BSDE exhibits promising advantages for nonlinear pricing problems.

Although the Deep BSDE method performs well for linear pricing BSDEs under the assumptions of complete markets and invertible volatility matrices adopted in Theorem 1, several important extensions deserve further investigation. The equivalence between BSDE solutions under the real-world measure and the risk-neutral measure derived in Theorem 1 may no longer hold when the volatility matrix becomes singular or the market is incomplete. In such settings, multiple equivalent martingale measures exist, and a unique arbitrage-free price cannot be obtained, which breaks the one-to-one correspondence between the two measures. Beyond the measure equivalence issue, nonlinear BSDEs (e.g., involving funding costs, credit valuation adjustment, or utility-based pricing) may require more complex network structures and training regimes. The computational cost grows with dimensionality, and practical high-dimensional applications may require parallel GPU strategies or variance reduction techniques to improve efficiency.

Our framework also suggests several promising directions for future research. For instance, extending the analysis to nonlinear BSDEs—such as those arising from XVA considerations or portfolio constraints—would further generalize the option pricing problem and showcase the computational strengths of Deep BSDE solvers in non-Markovian and path-dependent settings.

Author Contributions: Conceptualization, D.D. and D.X.; methodology, D.D.; software, D.D.; validation, D.D., K.T. and D.X.; formal analysis, D.D.; writing—original draft preparation, D.D.; writing—review and editing, D.X.; visualization, K.T.; supervision, D.X. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The original contributions presented in this study are included in the article. Further inquiries can be directed to the corresponding author. The implementation code for the proposed Deep BSDE solver is hosted at https://github.com/dongyadeng/dongyadeng (accessed on 10 November 2026). The repository will be made publicly available upon acceptance of this manuscript for publication.

Conflicts of Interest: The authors declare no conflicts of interest.

Appendix A

In this appendix, we provide detailed proofs of the main theoretical results used in the paper, including the equivalence between the pricing BSDE (Appendix A.1) and the classical Black–Scholes PDE, and the derivation of the risk-neutral driver (Appendix A.2).

Appendix A.1

Consider the standard pricing BSDE under the risk-neutral measure $\mathbb{Q}$:

$$\mathbf{dY_t} = -\mathbf{f}(\mathbf{t}, \mathbf{Y_t}, \mathbf{Z_t})\mathbf{dt} + \mathbf{Z_t^\top} \mathbf{dW_t^\mathbb{Q}}, \qquad \mathbf{Y_T} = \mathbf{g}(\mathbf{X_T}),$$

where X_t follows

$$\mathbf{dX_t} = \boldsymbol{\mu}(\mathbf{X_t}, \mathbf{t})\mathbf{dt} + \boldsymbol{\sigma}(\mathbf{X_t}, \mathbf{t})\mathbf{dW_t^\mathbb{Q}}, \qquad \mathbf{X_0} = \mathbf{x_0}.$$

Assume that the solution $Y_t = u(t,X_t)$ is sufficiently smooth. Applying Itô's lemma,

$$\mathbf{du}(\mathbf{t}, \mathbf{X_t}) = \left(\partial_\mathbf{t}\mathbf{u} + (\boldsymbol{\mu} \cdot \nabla_\mathbf{x})\mathbf{u} + 1/2\mathbf{Tr}\left[\boldsymbol{\sigma}\boldsymbol{\sigma}^\top \nabla_\mathbf{x}^2 \mathbf{u}\right]\right)\mathbf{dt} + \left(\nabla_\mathbf{x}\mathbf{u}\right)^\top \boldsymbol{\sigma}\mathbf{dW_t^\mathbb{Q}}.$$

Matching the diffusion term with the BSDE, we identify

$$\mathbf{Z_t} = \left(\nabla_\mathbf{x}\mathbf{u}\right)^\top \boldsymbol{\sigma}.$$

Matching the drift term yields the PDE:

$$\partial_\mathbf{t}\mathbf{u} + (\boldsymbol{\mu} \cdot \nabla_\mathbf{x})\mathbf{u} + \frac{1}{2}\mathbf{Tr}\left[\boldsymbol{\sigma}\boldsymbol{\sigma}^\top \nabla_\mathbf{x}^2 \mathbf{u}\right] + \mathbf{f}(\mathbf{t}, \mathbf{u}, \mathbf{Z_t}) = 0, \qquad \mathbf{u}(\mathbf{T}, \mathbf{x}) = \mathbf{g}(\mathbf{x}). \tag{A1}$$

Appendix A.2

For a standard European option on a single asset under the risk-neutral measure,

$$\mathbf{dS_t} = \mathbf{rS_t dt} + \boldsymbol{\sigma}\mathbf{S_t dW_t^\mathbb{Q}}, \qquad \mathbf{Y_T} = \mathbf{g}(\mathbf{S_T}),$$

the PDE (A1) reduces to the classical Black–Scholes PDE:

$$\partial_\mathbf{t}\mathbf{u} + \mathbf{rS}\partial_\mathbf{S}\mathbf{u} + \frac{1}{2}\boldsymbol{\sigma}^2\mathbf{S}^2\partial_\mathbf{S}^2\mathbf{u} - \mathbf{ru} = 0, \qquad \mathbf{u}(\mathbf{T}, \mathbf{S}) = \mathbf{g}(\mathbf{S}).$$

Comparing with (A1), we immediately identify that the BSDE driver must satisfy

$$\mathbf{f}(\mathbf{t}, \mathbf{Y_t}, \mathbf{Z_t}) = -\mathbf{rY_t}.$$

Appendix B

We summarize the Deep BSDE solver used in this paper in algorithmic form. The notation is consistent with Sections 3 and 4.

Deep BSDE Algorithm (Pseudocode): Numerical Scheme for Solving BSDEs Using Deep Neural Networks

1: Input:
Initial condition: $X_0 = x_0$, initial guess for $Y_0 = y_0$.
Drift function: μ
Volatility: σ.
Driver of the BSDE: f
Terminal condition: $Y_T = g(X_T)$.
Maturity T and number of time steps N.
2: Neural network: Fully connected layers

$$(x, t) \mapsto \phi(x, t) \in \mathbb{R}^{\dim_y \times \dim_d}$$

with three hidden layers and tanh activation.
3: Set $\Delta t = T/N$.

4: Generate Brownian increments:

$$\Delta W_i \sim \mathcal{N}(0, \Delta t), \qquad i = 0, \ldots, N-1.$$

5:Initialize:

$$X_0 = x_0, \qquad Y_0 = y_0.$$

6: Current time $t = i\Delta t$.

7: Approximate Z_t using the neural network:

$$Z_t = \phi(X_t, t).$$

8: Forward SDE update:

$$X_{t+\Delta t} = X_t + \mu(t, X_t, Y_t)\Delta t + \sigma(t, X_t)\Delta W_i.$$

9: Backward SDE update:

$$Y_{t+\Delta t} = Y_t - f(t, X_t, Y_t, Z_t)\Delta t + Z_t \Delta W_i.$$

10: Output: (X_T, Y_T).

Notes: ϕ denotes the neural network mapping from (t, X_t) to Z_t. Forward and backward updates follow the standard Euler discretization. The loss function enforces terminal condition $Y_T = g(X_T)$ for all batch paths.

References

1. Black, F.; Scholes, M. The pricing of options and corporate liabilities. *J. Political Econ.* **1973**, *81*, 637–654. [CrossRef] [PubMed]
2. Merton, R.C. Theory of rational option pricing. *Bell J. Econ. Manag. Sci.* **1973**, *4*, 141–183. [CrossRef]
3. Stulz, R.M. Options on the minimum or the maximum of two risky assets: Analysis and applications. *J. Financ. Econ.* **1982**, *10*, 161–185. [CrossRef]
4. Johnson, H. Option pricing when the variance is changing. *J. Financ. Quant. Anal.* **1987**, *22*, 143–151. [CrossRef]
5. Rogers, L.C.G.; Shi, Z. The value of an asian option. *J. Appl. Probab.* **1995**, *32*, 1077–1088. [CrossRef]
6. Glasserman, P. *Monte Carlo Methods in Financial Engineering*; Springer: Berlin/Heidelberg, Germany, 2004. [CrossRef]
7. Cox, J.; Ross, S.; Rubinstein, M. Option pricing: A simplified approach. *J. Financ. Econ.* **1979**, *7*, 229–263. [CrossRef]
8. Pardoux, E.; Peng, S. Adapted solution of a backward stochastic differential equation. *Syst. Control Lett.* **1990**, *14*, 55–61. [CrossRef]
9. El Karoui, N.; Peng, S.; Quenez, M.C. Backward stochastic differential equations in finance. *Math. Financ.* **1997**, *7*, 1–71. [CrossRef]
10. Peng, S.; Xu, M. Numerical algorithms for backward stochastic differential equations with 1-d brownian motion: Convergence and simulations. *ESAIM Math. Model. Numer. Anal.* **2011**, *45*, 335–360. [CrossRef]
11. Chessari, J.; Kawai, R.; Shinozaki, Y.; Yamada, T. Numerical methods for backward stochastic differential equations: A survey. *Probab. Surv.* **2023**, *20*, 486–567. [CrossRef]
12. E, W.; Han, J.; Jentzen, A. Deep learning-based numerical methods for high-dimensional parabolic PDEs and BSDEs. *Commun. Math. Stat.* **2017**, *5*, 349–380. [CrossRef]
13. Han, J.; Jentzen, A.; E, W. Solving high-dimensional PDEs using deep learning. *Proc. Natl. Acad. Sci. USA* **2018**, *115*, 8505–8510. [CrossRef] [PubMed]
14. Beck, C.; E, W.; Jentzen, A. Machine learning approximation algorithms for high-dimensional fully nonlinear PDEs and second-order BSDEs. *J. Nonlinear Sci.* **2019**, *29*, 1563–1619. [CrossRef]
15. E, W.; Han, J.; Jentzen, A. Algorithms for solving high-dimensional PDEs: From nonlinear monte carlo to machine learning. *Nonlinearity* **2021**, *35*, 278–310. [CrossRef]
16. Raissi, M.; Perdikaris, P.; Karniadakis, G.E. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear PDEs. *J. Comput. Phys.* **2019**, *378*, 686–707. [CrossRef]
17. Beck, C.; Hutzenthaler, M.; Jentzen, A.; Kuckuck, B. An overview on deep learning-based approximation methods for partial differential equations. *Discret. Contin. Dyn. Syst. Ser. B* **2023**, *28*, 3697–3746. [CrossRef]
18. Kapllani, L.; Teng, L. Deep learning algorithms for solving high-dimensional nonlinear backward stochastic differential equations. *Discret. Contin. Dyn. Syst. Ser. B* **2024**, *29*, 1695–1729. [CrossRef]
19. Jiang, Y.; Li, J. Convergence of the deep BSDE method for FBSDEs with non-lipschitz coefficients. *Probab. Uncertain. Quant. Risk* **2021**, *6*, 391. [CrossRef]
20. Andersson, K.; Andersson, A.; Oosterlee, C.W. Convergence of a robust deep FBSDE method for stochastic control. *SIAM J. Sci. Comput.* **2023**, *45*, A226–A255. [CrossRef]
21. Sirignano, J.; Spiliopoulos, K. DGM: A deep learning algorithm for solving partial differential equations. *J. Comput. Phys.* **2018**, *375*, 1339–1364. [CrossRef]
22. Wang, H.; Chen, H.; Sudjianto, A.; Liu, R.; Shen, Q. Deep learning-based BSDE solver for LIBOR market model with application to bermudan swaption pricing and hedging. *arXiv* **2018**, arXiv:1807.06622.

23. Becker, S.; Cheridito, P.; Jentzen, A. Pricing and hedging american-style options with deep learning. *J. Risk Financ. Manag.* **2020**, *13*, 158. [CrossRef]
24. Liang, J.; Xu, Z.; Li, P. Deep learning-based least squares forward-backward stochastic differential equation solver for high-dimensional derivative pricing. *Quant. Financ.* **2021**, *21*, 1309–1323. [CrossRef]
25. Yu, Y.; Ganesan, N.; Hientzsch, B. Backward deep BSDE methods and applications to nonlinear problems. *Risks* **2023**, *11*, 54. [CrossRef]
26. Gnoatto, A.; Picarelli, A.; Reisinger, C. Deep XVA solver: A neural network–based counterparty credit risk management framework. *SIAM J. Financ. Math.* **2023**, *14*, 314–352. [CrossRef]
27. Huré, C.; Pham, H.; Warin, X. Deep backward schemes for high-dimensional nonlinear PDEs. *arXiv* **2020**, arXiv:1902.01599.
28. Pardoux, E.; Peng, S. Backward stochastic differential equations and quasilinear parabolic partial differential equations. In *Stochastic Partial Differential Equations and Their Applications*; Springer: Berlin/Heidelberg, Germany, 1992; Volume 176, pp. 200–217. [CrossRef]
29. Shreve, S. *Stochastic Calculus for Finance II: Continuous-Time Models*; Springer: Berlin/Heidelberg, Germany, 2004. [CrossRef]
30. Musiela, M.; Rutkowski, M. *Martingale Methods in Financial Modelling*; Springer: Berlin/Heidelberg, Germany, 2005. [CrossRef]
31. Becker, S.; Cheridito, P.; Jentzen, A.; Welti, T. Solving high-dimensional optimal stopping problems using deep learning. *Eur. J. Appl. Math.* **2021**, *32*, 470–514. [CrossRef]
32. Wang, X.; Li, J.; Li, J. A deep learning based numerical PDE method for option pricing. *Comput. Econ.* **2023**, *62*, 149–164. [CrossRef]
33. Shi, Y.; Teng, B.; Wang, S. Option pricing mechanisms driven by backward stochastic differential equations. *Financ. Innov.* **2025**, *11*, 90. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.