A Deep Neural Network for Pricing American Options under Jump-Diffusion in High Dimension: A Backward Stochastic Differential Equation Approach

by

Alex Starr

A research paper
presented to the University of Waterloo
in fulfillment of the
research paper requirement for the degree of
Master of Mathematics
in
Computational Mathematics

Waterloo, Ontario, Canada, 2025

Author's Declaration

Abstract

Models that incorporate jumps are essential for capturing abrupt price movements observed in financial markets. While jump–diffusion dynamics provide a more realistic description of asset behaviour than pure diffusions, they add computational complexity to accurately price options. This difficulty is especially pronounced for American options, which dominate practical trading but have received comparatively less attention in the academic literature than their European counterparts.

Recent advances in machine learning offer new opportunities for addressing these challenges. In particular, neural methods for solving backward stochastic differential equations provide a simulation-based alternative to classical PDE and PIDE techniques, which struggle to scale to high-dimension from the curse of dimensionality. This thesis develops a deep-learning framework for pricing American options under multi-asset jump–diffusion models. We derive a discrete-time BSDE that incorporates both diffusion and jump components and integrate it into the existing diffusion-only deep neural network framework. The resulting method achieves accurate and stable performance in dimensions up to $d = 100$, demonstrating that modern deep-learning tools can make high-dimensional American option pricing feasible in realistic jump–diffusion settings.

Acknowledgements

I would like to thank my research supervisor, Professor Justin Wan, for his guidance and support throughout this project. I would also like to thank Professor Tony Wirjanto for taking the time to review this paper as its second reader.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Recent advances in artificial intelligence (AI) and deep learning have allowed for substantial advancements in quantitative finance. Tasks that traditionally relied on closed-form solutions, partial differential equation solvers, or large Monte Carlo simulations can now be approached using data-driven methods. In particular, deep neural networks trained on simulated market dynamics have shown strong ability to learn option prices, risk sensitivities, and hedging strategies in settings where classical methods become computationally too slow or too difficult to implement.

A powerful nexus that connects financial modelling with machine learning is the use of backward stochastic differential equations (BSDEs). In finance, a BSDE describes how the price of a financial derivative evolves backward from its payoff at maturity. This representation naturally incorporates a hedging strategy required to replicate the payoff under a risk-neutral model. Neural BSDE solvers take advantage of this structure by using neural networks to approximate the conditional expectations that appear in the backward recursion, allowing them to scale effectively to high dimensions and to handle complex products such as those with path dependence or early-exercise features.

Although BSDE-based approaches work well under continuous market dynamics, real financial markets often exhibit sudden and discrete jumps due to earnings announcements, macroeconomic events, or periods of market stress. Modelling these discontinuities is essential for accurate pricing and hedging, but incorporating jumps into a BSDE creates additional computational challenges. The jump component introduces extra nonlocal terms that reflect the average effect of all possible future jumps. Directly estimating this term at every time step typically requires repeatedly simulating large numbers of jump scenarios, making the required training process computationally slow and unstable numerically.

This thesis introduces a practical and scalable way to overcome the bottleneck repeatedly solving nonlocal terms. The key finding is to treat the nonlocality as another conditional expectation problem. This means that it can be solved for in the same way as the option price itself. Once trained, this compensator network can be inserted directly into the BSDE solver, eliminating the need to sample jumps during the training process. This substantially improves computational efficiency while preserving the intended financial behaviour of the model, such as the martingale property of discounted asset prices.

In addition, the framework developed in this thesis extends naturally to American options, whose early-exercise feature requires determining when it is optimal to exercise rather than continue holding the derivative. Classical numerical methods for American options rely on solving a free-boundary problem or a variational inequality, both of which become extremely difficult when jumps are present or when many underlying assets drive the option. This work extends existing work on American options, and integrates the jump feature into the existing American option framework of (source). By combining the pre-trained compensator with a reflected BSDE formulation, the proposed approach provides a unified neural-network framework for pricing and hedging American options under jump-diffusion dynamics.

A major advantage of BSDE-based methods is that they operate on simulated paths rather than spatial grids. This allows them to scale to dimensions that are inaccessible to finite-difference or PIDE solvers. The combination of a pre-trained compensator, a reflected BSDE formulation, and a neural network architecture tailored to high-dimensional dynamics makes it possible to efficiently model multi-asset derivatives and basket options in settings with realistic jump behaviour.

The remainder of this thesis develops these ideas in detail. Chapter 3 provides the necessary background on European and American options, Monte Carlo simulation, jump-diffusion, and the BSDE formulation used in modern learning approaches. Chapter 4 introduces the compensator network, the integration with reflected BSDEs, and the full neural architecture. Chapter 5 presents numerical results demonstrating accuracy, stability, and scalability. Chapter 6 concludes with limitations and potential extensions of the thesis, including stochastic volatility and multi-factor Lévy processes.

Contributions

This thesis makes several contributions to the study of high-dimensional American option pricing under jump–diffusion dynamics. The principal contributions of this thesis are as follows:

- **A discrete-time BSDE for jump–diffusions that respects the no-arbitrage conditions.** We show a derivation of the discrete jump BSDE with correlated jump structure. Culminating in *Theorem 3.2.3*, we provide the precise backward dynamics satisfied by the option value, the diffusion hedge Z_t, and the jump response vector U_t. This theorem serves as the mathematical foundation for the proposed learning algorithm.

- **A scalable neural architecture for high-dimensional discontinuous dynamics.** We extend the Chen–Wan architecture by introducing a dual-head design with separate time-decay outputs for the continuation value and jump response, enabling stable training in high dimension.

- **Empirical valuation of American Options in dimensions up to $d = 100$.** The experiments in Chapter 4 demonstrate high accuracy across a wide range of jump intensities, basket dimensions, and time discretizations, while maintaining computational stability.

Collectively, these contributions establish a practical, theoretically grounded, and computationally scalable framework for pricing and hedging high-dimensional American options under realistic jump–diffusion dynamics.

Chapter 2

Background

2.1 Overview of Financial Options

In financial markets, a financial derivative can refer to any a class of security whose value depends on, and is thus "derived" from, the performance of another underlying asset, typically denoted S when referring to stocks. An option is a type of financial derivative that grants the holder a contractual right to participate in the returns of the underlying asset, without imposing any obligation on them. For example, a *call* option will specify a strike price K, which grants the holder the right, but not the obligation, to purchase the underlying stock S for K dollars at some time T in the future. If at this time $S > K$, the holder would exercise the option, purchasing the underlying for K dollars, and could then immediately sell it for S dollars. However, if $S < K$, the holder of the option would not exercise their option. Since no transaction has happened, the value of the option is 0 at time T. From this construction, the value of an option at exercise in the future is $\max(S – K, 0)$.

This asymmetric payoff profile, marked by capped downside and unlimited upside, makes options a valuable tool that can be tailored to a wide range of hedging and speculative objectives. Standard expositions of these options may be found in Shreve [19] and Wilmott, Howison, and Dewynne [21]. It is then an important and valuable problem to be able to come up with a fair price for an option, this fair price is known as the premium, or option price.

2.1.1 European Options

There are various styles of options; the simplest case of which is the European option. A European option restricts the exercise of the contract to a single predetermined maturity date T.

Let $K > 0$ denote the strike price at which the transaction may occur. A European *call* option grants the right to purchase the underlying at K, whereas a European *put* option grants the right to sell it. At maturity, the payoff is determined solely by the terminal asset price S_T. The canonical payoff functions are

$$g_{\text{call}}(S_T) = \max(S_T - K, 0), \qquad g_{\text{put}}(S_T) = \max(K - S_T, 0).$$

These payoffs capture the fundamental asymmetry of option contracts: favorable movements of the underlying generate positive returns for the holder, while adverse movements result in no exercise and thus no additional loss beyond the initial premium. Since we do not know the price of S_T, we determine that the value at current time should represent the expected option value over all potential asset prices. This is the standard arbitrage-free framework, in which the price of a European contingent claim is a discounted expected payoff under a risk-neutral probability measure,

$$V(t, S_t) = e^{-r(T-t)}\,\mathbb{E}[g(S_T) \mid S_t]\,,$$

where r denotes the risk-free interest rate. This representation provides a probabilistic foundation for option valuation and serves as a benchmark for numerical and analytical methods introduced later in this chapter.

2.1.2 American Options

American options generalize European options by permitting exercise at any time prior to or at maturity. This additional flexibility increases their economic value but introduces a substantial conceptual and mathematical complication: the holder must determine when it is optimal to exercise. The resulting valuation problem is inherently dynamic and requires comparing the immediate exercise payoff with the value of continuation at every point in time. Formally, an American option with payoff $g(S_t)$ admits the following arbitrage-free value

$$V(t, S_t) = \sup_{\tau \in \mathcal{T}_{t,T}} \mathbb{E}\big[e^{-r(\tau-t)} g(S_\tau) \mid S_t\big]\,,$$

where $\mathcal{T}_{t,T}$ denotes a set of all stopping times with respect to the filtration generated by the underlying asset. The optimal stopping formulation naturally induces two regions:

$$\text{Exercise region: } V(t,S) = g(S), \qquad \text{Continuation region: } V(t,S) > g(S).$$

American options are therefore structurally more complex than their European counterparts, as their valuation depends not only on the terminal payoff but also on the optimal exercise policy. This distinction plays an important role in the subsequent formulation of numerical methods and motivates the use of probabilistic techniques that can accommodate early exercise without explicitly tracking the exercise boundary.

2.2 Asset Price Dynamics

A central component of option valuation is the choice of a mathematical model for the underlying asset price. Such a model must balance realism with analytical and numerical tractability. It should be rich enough to capture essential features of financial markets, notably randomness, volatility, and occasional abrupt movements while remaining amenable to simulation and analysis.

In this section, we introduce two widely used classes of asset price models: the geometric Brownian motion (GBM) model and jump–diffusion models. These serve as fundamental building blocks for the more general frameworks discussed later in the chapter. Comprehensive treatments of these models may be found in Shreve [19] and Øksendal [15].

2.2.1 Geometric Brownian Motion

The classical starting point for modelling asset prices is the geometric Brownian motion (GBM). Under GBM, the asset price process S_t evolves according to the following stochastic differential equation (SDE)

$$\frac{dS_t}{S_t} = \mu\, dt + \sigma\, dW_t,$$

where μ denotes the drift, $\sigma > 0$ the volatility, and W_t a standard Brownian motion. This model assumes that price changes occur continuously and are driven by a combination of deterministic growth and random fluctuations. Solving the SDE yields the following explicit representation

$$S_t = S_0 \exp\left(\left(\mu - \frac{1}{2}\sigma^2\right) t + \sigma W_t\right),$$

demonstrating that GBM produces strictly positive asset prices and lognormally distributed returns. These properties contribute to GBM's popularity and underpin the Black–Scholes option pricing formula.

Although GBM is mathematically convenient, real financial markets often exhibit features that it cannot capture. In particular, price trajectories may experience sudden jumps due to news announcements, earnings reports, or broader market events. Academic literature has also shown that real-world asset returns exhibit heavier tails than are produced by GBM (source). Such discontinuities motivate the extension of GBM to jump–diffusion models.

2.2.2 Jump–Diffusion Models

Jump–diffusion models extend GBM by incorporating discrete, possibly large price changes alongside the continuous fluctuations generated by Brownian motion. Formally, the asset price follows

$$\frac{dS_t}{S_{t^-}} = \mu\, dt + \sigma\, dW_t + (J_t - 1)\, dN_t,$$

where N_t is a Poisson process with intensity $\lambda > 0$, governing the arrival of jump events, and J_t denotes the random multiplicative jump size. The notation S_{t^-} denotes the left limit of the price just before the jump, reflecting the fact that the process may be discontinuous.

This framework preserves the continuity of GBM between jump times while allowing the price to undergo sudden shifts. It is capable of generating heavier tails and greater return variability than the standard diffusion models alone and can better reflect market behaviour during periods of heightened uncertainty.

A widely used specification is the Merton jump–diffusion model, in which the jump sizes are lognormally distributed. Under this model,

$$J_t = e^{Y_t}, \qquad Y_t \sim \mathcal{N}(\mu_J, \sigma_J^2),$$

and each jump arrival results in a proportional change of the form $S_t = S_{t^-} J_t$. The jump intensity λ controls the expected number of jumps per unit of time, while the parameters μ_J and σ_J control their average magnitude and variability.

Jump–diffusion models form a natural bridge between classical diffusion models and more general Lévy processes. Although they remain analytically tractable in some settings, they introduce nonlocal features into the valuation problems for derivatives, necessitating more sophisticated probabilistic and numerical tools.

2.3 Standard Option Pricing Techniques

2.3.1 Monte Carlo Simulation

Once a stochastic model for the underlying asset price has been specified, the pricing problem can be expressed in terms of the distribution of the terminal asset value. Monte Carlo methods approximate this distribution by generating a large number of sample paths of the underlying process and computing discounted payoffs evaluated along those paths. This pathwise interpretation makes Monte Carlo applicable to a broad range of models, including those involving jumps or other features that complicate analytic treatments. A detailed exposition of this topic appears in Glasserman [11].

Path Simulation

Given the discrete-time approximations introduced in Section 3.2, one constructs a time grid $0 = t_0 < t_1 < \cdots < t_N = T$, and simulates the asset price sequentially along this grid. For GBM, a Euler–Maruyama discretization scheme yields

$$S_{t_{n+1}} = S_{t_n} \left(1 + \mu\,\Delta t + \sigma\sqrt{\Delta t}\,Z_n\right), \qquad Z_n \sim \mathcal{N}(0,1),$$

while for jump–diffusion models, a simulated increment incorporates jump arrivals and jump magnitudes,

$$S_{t_{n+1}} = S_{t_n} \left(1 + \mu\,\Delta t + \sigma\sqrt{\Delta t}\,Z_n\right) \prod_{k=1}^{N_n} J_{n,k}, \qquad N_n \sim \text{Poisson}(\lambda\Delta t).$$

Repeating this construction M times produces an ensemble of independent trajectories $\{S_t^{(m)}\}_{m=1}^{M}$, which serves as a numerical proxy for the model's distributional dynamics.

Valuation

For a European option with payoff $g(S_T)$, the Monte Carlo estimator of the time-t value is

$$\widehat{V}(t, S_t) = e^{-r(T-t)} \frac{1}{M} \sum_{m=1}^{M} g\left(S_T^{(m)}\right).$$

This estimator converges to the true price by the law of large numbers, with a standard error that scales as $M^{-1/2}$. Its accuracy therefore depends on both the number of sample paths and the time-step size used in discretizing the underlying process. Reducing either source of error improves precision but increases computational cost. In practice, both discretization bias and statistical variance must be controlled carefully when using Monte Carlo estimates as benchmarks or as training data for learning-based methods.

Variance Reduction

As the convergence rate of Monte Carlo is slow relative to many deterministic methods, variance reduction techniques are often employed to improve efficiency. Without going into excess detail on each method, standard approaches include antithetic variates, control variates, and importance sampling; each method reduces estimator variance without increasing the number of simulated paths, we refer the reader to Glasserman [11] for an in-depth discussion of these techniques. These ideas are particularly useful in models with jumps, where the distribution of S_T may be skewed or heavy-tailed and naive sampling may lead to high-variance estimates.

Role in Simulation-Based Methods

Beyond providing a standalone valuation method, Monte Carlo simulation forms the basis of many modern numerical approaches to option pricing. Methods based on regression, backward iteration, or BSDE formulations all rely on the availability of simulated asset paths. In the context of this thesis, Monte Carlo plays an additional role: it is used to generate high-quality estimates of certain conditional expectations, such as the jump compensator, which serve as training targets for neural networks. The overall accuracy and stability of the learning framework therefore depend critically on the quality of the simulated data.

Monte Carlo methods thus serve both as a flexible valuation tool and as a fundamental computational component underlying the simulation-based techniques developed in the following sections.

2.3.2 Classical PDE and PIDE Approaches

Option valuation problems can often be reformulated as partial differential equations (PDEs). In models without jumps, the price $V(t, S)$ of a European option typically satisfies a parabolic PDE derived from the infinitesimal generator of the underlying asset process. For example, under geometric Brownian motion, the classical Black–Scholes PDE takes the form

$$\frac{\partial V}{\partial t} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2} + rS\frac{\partial V}{\partial S} - rV = 0,$$

with terminal condition $V(T, S) = g(S)$. A detailed derivation of this result may be found in Wilmott, Howison, and Dewynne [21].

When jumps are incorporated into the model, the valuation equation becomes a partial integro–differential equation (PIDE), since the infinitesimal generator now contains a nonlocal integral term representing the expected impact of jumps. In the Merton jump–diffusion model, where jump sizes are lognormally distributed, the option price satisfies the PIDE

$$\frac{\partial V}{\partial t} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2} + (r - \lambda\kappa)S\frac{\partial V}{\partial S} - rV + \lambda \int_{\mathbb{R}} \Big[V(t, Se^y) - V(t, S)\Big] f_Y(y)\, dy = 0,$$

where λ is the jump intensity, f_Y is the density of the normal jump-size distribution Y, and $\kappa = \mathbb{E}[e^Y - 1]$. Derivations of this PIDE and its variants may be found in Merton [14] and in the monograph of Cont and Tankov [7].

PDE and PIDE methods have been extensively developed and are effective in low dimensions, where the spatial domain is small and the integral terms can be evaluated with sufficient accuracy. However, the computational cost of grid-based schemes grows exponentially with the number of underlying assets. Each added dimension introduces an extra spatial axis in the grid, rapidly making the discretization intractable. This phenomenon, often referred to as the curse of dimensionality, severely limits the applicability of PDE and PIDE methods to multi-asset options or models with high-dimensional state variables.

For this reason, although these classical methods provide valuable theoretical insight and serve as effective tools in one- and two-dimensional settings, their computational limitations motivate the search for alternative approaches capable of scaling to higher dimensions.

2.4 American Options and Optimal Stopping

American options introduce an early-exercise feature, requiring the valuation problem to account for the holder's decision of when to exercise. In the PDE framework, this leads to a variational inequality that couples the pricing equation with the constraint

$$V(t,S) \geq g(S),$$

together with a complementary condition that characterizes the exercise boundary. The resulting free-boundary problem has been studied extensively and is well understood in low-dimensional diffusion models; see, for example, Broadie and Detemple [5].

When jumps are present, the valuation equation becomes a PIDE complementarity problem, and the exercise boundary interacts with the nonlocal integral term. This substantially increases the numerical burden: in addition to discretizing the spatial domain, one must handle the jump integral and ensure consistency with the early-exercise constraint. As the dimension of the underlying state increases, grid-based discretizations become infeasible, and computing the exercise region becomes prohibitively expensive.

Consequently, while PDE and PIDE formulations provide important conceptual foundations for American option pricing, their scalability is limited in scope. High-dimensional problems or models with complex jump structures require alternative frameworks that avoid explicit grid construction and can accommodate optimal stopping without relying on free-boundary identification.

2.5 Backward Stochastic Differential Equations

The limitations of grid-based PDE and PIDE methods in high-dimensional settings have motivated the development of probabilistic techniques for the option valuation. A central tool in this direction is the theory of backward stochastic differential equations (BSDEs), first introduced by Pardoux and Peng [16], who established existence and uniqueness results for a broad class of nonlinear backward equations. Since then, BSDEs have played an important role in mathematical finance, particularly in the pricing and hedging of contingent claims; see, for example, El Karoui, Peng, and Quenez [10], Zhang [22], and Pham [17].

In its basic form, a BSDE consists of a pair of adapted processes (Y_t, Z_t) satisfying

$$Y_t = \xi + \int_t^T f(s, Y_s, Z_s)\, ds - \int_t^T Z_s\, dW_s, \qquad t \in [0, T],$$

for a given terminal condition ξ and a generator f. The solution Y_t represents the value of the contingent claim, while Z_t corresponds to the hedging integrand associated with the driving Brownian motion. In a risk-neutral setting, f arises from the drift adjustment of the underlying asset dynamics, and the terminal condition ξ is the option payoff.

For European claims, choosing $\xi = g(S_T)$ and an appropriate generator, which is GBM, as discussed above, yields a probabilistic representation of the arbitrage-free price,

$$Y_t = V(t, S_t),$$

where V denotes the option value function. Since the BSDE framework relies on simulated trajectories of the underlying process rather than spatial grids, it avoids the curse of dimensionality that limits PDE and PIDE methods. Numerical schemes based on Monte Carlo regression—such as those developed by Bouchard and Touzi [4] and Gobet, Lemor, and Warin [12]—are well suited for approximating (Y_t, Z_t) in high-dimensional settings.

The BSDE formalism also provides a representation of American options. In this case, the value process must satisfy the early-exercise constraint

$$Y_t \geq g(S_t),$$

which leads to a *reflected* BSDE. The reflection is enforced through an increasing process that forces the solution to remain above the payoff. This approach captures both the option value and the optimal stopping strategy without explicitly solving a free-boundary problem. Foundational results for reflected BSDEs and their applications to optimal stopping appear in El Karoui et al. [10] and Touzi [20].

From a computational standpoint, solving a BSDE typically involves simulating paths of the underlying process and performing a backward-time approximation of conditional expectations. This simulation-based structure extends naturally to models with more complex features, such as state dependence or high-dimensional dynamics, and forms the basis of the learning-based BSDE methods, such as in Chen & Wan [6], which form the basis for the neural BSDE solver methods discussed in the next chapter.

Chapter 3

Model Formulation

3.1 Modelling Dependence in Multi-Asset Jump Processes

Before setting up the model for pricing such options in the multi-dimensional case, we first need to define how we model jump dependence among underlying assets. Some jumps reflect events that are specific to a single asset, while others correspond to market-wide shocks that move many assets simultaneously. Capturing these different forms of dependence is essential for a realistic model of discontinuities. The frameworks described below provide several complementary ways to introduce such a structure.

Independent jump arrivals. The most direct approach assigns each asset its own Poisson clock, so that jumps arrive independently in each component. A jump in asset i occurs when its clock rings, and its size is drawn from the marginal jump-size distribution specified for that asset. This construction is simple and flexible, but it cannot generate simultaneous jumps or coordinated movements across assets.

Common shock models. When several assets are exposed to the same underlying sources of risk, it is natural to model them as sharing a common jump clock. Here a single Poisson process represents system-wide shock events, and whenever this clock rings, all assets in the affected group experience a jump. The magnitudes of the jumps may differ, but their timing is shared, creating instantaneous co-movement. Common-shock constructions appear widely in actuarial science and financial modelling; see Platen and Heath [18] or Dassios and Zhao [8]. Additional asset-specific clocks may be included to retain idiosyncratic behaviour. Literature has demonstrated that using a small number of common shocks with a random participation structure can provide an accurate representation of the complex jump dependence observed in real-world financial data [13, 1, 2, 9].

Dependence through copula-based jump sizes. Another way assets can move together is through the *sizes* of their jumps rather than their timing. In this approach, all jumps are driven by a single arrival process, but the jump vector

$$X = (X^{(1)}, \dots, X^{(d)})$$

is sampled from a multivariate distribution which dependence structure is specified by a copula.

Definition 3.1.1 (Copula)**.** A copula is a multivariate distribution function $C : [0,1]^d \to [0,1]$ with uniform marginals. Given marginal distribution functions $F_{X^{(i)}}$ for the components of a random vector, the joint distribution can be written in the form

$$F_X(x_1, \dots, x_d) = C(F_{X^{(1)}}(x_1), \dots, F_{X^{(d)}}(x_d)) \,.$$

This representation separates marginal behaviour from dependence and is foundational in the modelling of multivariate jump processes; see Cont and Tankov [7, Ch. 4].

Incorporating the copula framework introduces a dependence structure among jump sizes while retaining full freedom to specify the one-dimensional distributions. For example, assets may exhibit symmetric dependence, asymmetric dependence, or tail clustering, all depending on the chosen copula family.

A common choice of copula is the Gaussian copula. One begins by sampling a Gaussian vector with a chosen correlation matrix and mapping each component through the standard normal distribution function. The resulting uniform vector is then pushed through the inverse marginals to obtain a joint jump-size vector. In this way, the correlation matrix directly determines how strongly the jump sizes tend to move together.

The copula approach can be combined with a common shock model as needed, enforcing correlation in jump arrival time and jump size. A model may include both a common jump clock affecting all assets and individual clocks for idiosyncratic shocks, while the sizes of the common jumps may follow a copula-based multivariate distribution. This layered structure provides substantial flexibility for reproducing a wide range of dependence patterns observed in practice, while not having to model a seperate Poisson arrival process for every subset of assets. For use in the BSDE framework, all we need is a Poisson arrival process for each possible jump case.

3.2 Jump–Diffusion Dynamics and the BSDE Representation

By construction, the BSDE solver enforces the solution to an SDE over a set of sampled paths. Thus, we begin our extension of Chen & Wan [6] solver by deriving a new BSDE formulation that incorporates this extra feature. We begin by describing the stochastic model used for the underlying state process. As mentioned previously, we choose S to follow a jump process, however we now extend it to the multi-dimensional case. Throughout, $S_t \in \mathbb{R}_+^d$ denotes the vector of underlying asset levels at time t. The process evolves under the risk–neutral measure $\mathbb{Q}$ and is driven by both continuous fluctuations and discrete jumps. Continuous movements are generated by Brownian noise, while jumps arrive according to a family of Poisson processes. This combination allows the model to capture sudden discontinuities, common shocks, and idiosyncratic movements within a unified framework.

The mathematical link between jump–diffusions, BSDEs, and the corresponding PIDEs is now well established. For a full and complete derivation of the link between BSDEs and PIDEs, we refer the reader to Barles, Buckdahn, and Pardoux [3], where they extend the classical Feynman–Kac correspondence to systems driven jointly by Brownian motion and Poisson random measures. Their analysis shows that, under suitable regularity conditions, the solution of a BSDE with jumps is precisely the viscosity solution of the associated PIDE, and conversely that such PIDEs admit a probabilistic representation through BSDEs. This result justifies treating the option price process as the solution of a backward equation which martingale components capture the sensitivities to both continuous diffusion risks and discontinuous jump risks.

Definition 3.2.1 (General jump–diffusion model)**.** Let $\{W_t\}_{t\geq 0}$ be an d–dimensional Brownian motion, and let $\{N_t^{(m)}\}_{m=1}^M$ be independent Poisson processes with constant intensities $\{\lambda_m\}_{m=1}^M$. For each jump source m, let $J_t^{(m)}$ denote a random mark describing the size or direction of the jump, and let

$$\Gamma^{(m)}(t, S_{t-}, J_t^{(m)})$$

represent the instantaneous change in the state when jump type m occurs. The process S_t satisfies the d–dimensional jump–diffusion SDE

$$\mathrm{d}S_t = b(t, S_t)\,\mathrm{d}t + \Sigma(t, S_t)\,\mathrm{d}W_t + \sum_{m=1}^{M} \Gamma^{(m)}\left(t, S_{t^-}, J_t^{(m)}\right)\,\mathrm{d}N_t^{(m)}, \tag{3.1}$$

Here:

- $b(t, S)$ is the drift vector determining the average rate of change of each asset,
- $\Sigma(t, S)$ is the $d \times d$ diffusion matrix defined by

$$\Sigma(t, S) = \operatorname{diag}\big(\sigma_1(t, S), \dots, \sigma_d(t, S)\big)\, \rho^{1/2},$$

  where $\sigma_i(t, S)$ is the instantaneous volatility of asset i and ρ is the correlation matrix governing the co-movement of the Brownian drivers.
- $dN_t^{(m)}$ equals 1 if jump type m occurs at time t and 0 otherwise,
- $\Gamma^{(m)}$ determines how the state moves in response to a jump.

This formulation encompasses a wide class of stochastic models used in finance and insurance.

Example: Merton Jump–Diffusion In the one–dimensional Merton model, jumps occur according to a single Poisson process ($M = 1$). A jump multiplies the asset price by a random factor J (often lognormally distributed). In this case,

$$\Gamma^{(1)}(t, S_{t-}, J) = S_{t-}(J - 1),$$

and the SDE becomes

$$dS_t = (r - \lambda\kappa) S_t \, dt + \sigma S_t \, dW_t + S_{t-}(J - 1)\, dN_t,$$

where $\kappa = \mathbb{E}[J - 1]$. Thus, the Merton model appears as a special instance of the general form (3.1).

To derive the backward equation satisfied by an option price, we use two classical results: Itô's lemma for jump processes and the risk–neutral martingale property of discounted asset prices. For a full BSDE derivation, we refer readers to Chen & Wan [6], where the extension for the jump case is using the *Itô's lemma with jumps* instead of the continuous variant of *Itô's lemma.*

Theorem 3.2.1 (Itô's lemma with jumps)**.** *Let X_t be a càdlàg semimartingale with continuous part X_t^c and jump measure $\mu(dt, dx)$ with compensator $\nu(dx)\, dt$. For any sufficiently smooth function $f(t, x)$,*

$$\begin{aligned} df(t, X_t) &= f_t(t, X_t)\, dt + \nabla_x f(t, X_t) \cdot dX_t^c + \tfrac{1}{2} \operatorname{Tr}\left(D_{xx}^2 f(t, X_t)\, d[X^c]_t\right) \\ &\quad + \int_{\mathbb{R}^d} \big(f(t, X_{t-} + x) - f(t, X_{t-})\big)\, (\mu(dt, dx) - \nu(dx)\, dt). \end{aligned} \tag{3.1}$$

This formula shows that the differential of $f(t, X_t)$ has three components: a drift contribution from the predictable part of the motion, a Brownian term from continuous fluctuations, and a jump term accounting for the instantaneous jump increments. A full definition and proof of Itô's lemma with jumps can be found in Cont and Tankov [7, Ch. 8].

Theorem 3.2.2 (Risk–neutral pricing condition)**.** *Let V_t denote the value of a claim paying $g(S_T)$ at time T. Under the risk–neutral measure,*

$$e^{-rt}V_t = \mathbb{E}^{\mathbb{Q}}\left[e^{-rT}g(S_T)\right], \tag{3.2}$$

and thus the discounted price process $e^{-rt}V_t$ is a martingale.

3.2.1 Derivation of the Jump BSDE

Apply Itô's Lemma Let $V_t = v(t, S_t)$ and note that $V_T = g(S_T)$. Applying Itô's lemma for jumps (Theorem 3.2.1) to $v(t, S_t)$ under the dynamics (3.1) yields

$$\begin{aligned} dV_t = \Big[& v_t(t, S_t) + \nabla_x v(t, S_t) \cdot b(t, S_t) + \tfrac{1}{2}\operatorname{Tr}\big(\Sigma\Sigma^\top(t, S_t)\, v_{xx}(t, S_t)\big) \\ & + \sum_{m=1}^{M} \lambda_m \mathbb{E}\Big[v\big(t, S_{t-} + \Gamma^{(m)}(t, S_{t-}, J_t^{(m)})\big) - v(t, S_{t-})\Big]\Big] dt \\ & + Z_t\, dW_t + \sum_{m=1}^{M} U_t^{(m)}\, d\widetilde{N}_t^{(m)}, \end{aligned} \tag{3.3}$$

where

$$Z_t := \nabla_x v(t, S_t)\, \Sigma(t, S_t), \qquad U_t^{(m)} := v\Big(t, S_{t-} + \Gamma^{(m)}(t, S_{t-}, J_t^{(m)})\Big) - v(t, S_{t-}),$$

and

$$d\widetilde{N}_t^{(m)} := dN_t^{(m)} - \lambda_m\, dt$$

are the compensated Poisson increments. Thus $U_t^{(m)}$ represents the instantaneous change in option value if a jump of type m occurs at time t.

Enforce the No Arbitrage Condition To obtain the backward representation, we consider the discounted value process $M_t = e^{-rt}V_t$. Under the risk–neutral measure this process must be a martingale, reflecting the fundamental financial requirement of no arbitrage. If M_t were to carry a predictable drift term, say $\mathrm{d}M_t = \alpha_t\,\mathrm{d}t +$ martingale terms with $\alpha_t \neq 0$, then an investor could exploit this drift through a self–financing strategy. In particular, holding one unit of the claim and financing the position at the risk–free rate produces a wealth process whose discounted dynamics inherit the same drift $\alpha_t \, \mathrm{d}t$. A positive drift ($\alpha_t > 0$) would yield a strictly increasing expected discounted gain, while a negative drift would permit riskless profit by shorting the claim and investing the proceeds. In either case, the strategy requires no external cash flows—hence "self–financing"—and generates a predictable excess return, contradicting the fundamental theorem of asset pricing. Therefore, the drift of M_t must vanish. When the Itô–jump expansion of $V_t = v(t, S_t)$ is computed, all drift contributions must therefore combine exactly to $rV_t \, \mathrm{d}t$. Enforcing this identity removes the drift from the discounted process and yields the jump–augmented BSDE in which the processes Z_t and $U_t^{(m)}$ represent the instantaneous sensitivities to diffusion and jump risks.

To obtain a backward representation of the price process, we start from the Itô–jump expansion of $V_t = v(t, S_t)$ obtained in (3.3). In particular, V_t satisfies

$$dV_t = A_t \, dt + Z_t \, dW_t + \sum_{m=1}^{M} U_t^{(m)} \, d\widetilde{N}_t^{(m)},$$

where the coefficient A_t collects all drift contributions arising from Itô's formula:

$$A_t = v_t(t, S_t) + \nabla_x v(t, S_t) \cdot b(t, S_t) + \tfrac{1}{2} \operatorname{Tr} \left(\Sigma\Sigma^\top (t, S_t) \, v_{xx}(t, S_t) \right)$$

$$+ \sum_{m=1}^{M} \lambda_m \, \mathbb{E}\big[v(t, S_{t-} + \Gamma^{(m)}) - v(t, S_{t-}) \big].$$

To determine the form of this drift, we examine the discounted process $\widetilde{V}_t := e^{-rt} V_t$. Notably, by (Theorem 3.2.2), this is martingale and we will be able to use the fact that it's drift is 0. Using the product rule,

$$d\widetilde{V}_t = e^{-rt} \, dV_t \; - \; re^{-rt} V_t \, dt,$$

and substituting the expression for dV_t into expression for dM_t yields

$$d\widetilde{V}_t = e^{-rt} \Big[(A_t - rV_t) \, dt + Z_t \, dW_t + \sum_{m=1}^{M} U_t^{(m)} \, d\widetilde{N}_t^{(m)} \Big].$$

Under the risk–neutral measure, the discounted price of any tradable asset must be a martingale. Thus the drift of M_t (the coefficient of dt) must vanish. Since $e^{-rt} > 0$, this condition is equivalent to

$$A_t - rV_t = 0, \qquad \text{that is,} \qquad A_t = rV_t.$$

In other words, the collection of drift terms produced by Itô's formula must match the instantaneous discounting rate r in order for the pricing process to be arbitrage–free. With this identification, the dynamics of V_t reduce to

$$dV_t = rV_t\,dt + Z_t\,dW_t + \sum_{m=1}^{M} U_t^{(m)}\,d\widetilde{N}_t^{(m)}.$$

Finally, rewriting this in backward form gives the BSDE

Theorem 3.2.3 (Jump–Diffusion BSDE Representation)**.** *Let $(S_t)_{t\geq 0}$ follow the jump–diffusion dynamics in (3.1), and let the terminal payoff be $V_T = g(S_T)$. Under the risk–neutral measure, and imposing the no–arbitrage condition that the discounted price process $e^{-rt}V_t$ is a martingale, the option value process $(V_t)_{t\geq 0}$ satisfies the backward stochastic differential equation*

$$-\mathrm{d}V_t = -rV_t\,\mathrm{d}t - Z_t\,\mathrm{d}W_t - \sum_{m=1}^{M} U_t^{(m)}\,\mathrm{d}\widetilde{N}_t^{(m)}, \qquad V_T = g(S_T). \tag{3.4}$$

This equation describes the backward evolution of the option price in the presence of both continuous and jump–driven sources of uncertainty. The terms Z_t and $U_t^{(m)}$ quantify the sensitivity of the option value to Brownian shocks and jump arrivals, respectively, and will play a central role in the numerical scheme developed in later chapters.

3.3 Discrete-Time Approximation of the BSDE

The continuous-time BSDE from Theorem 3.2.3 characterizes how the option value evolves when both diffusion and jump risks are present. To make this relation usable in numerical work, it must be translated into a form that acts on a discrete set of time points. The purpose of this section is to show how the continuous equation naturally gives rise to a backward recursion that links one time step to the next.

Discretization Scheme We divide the horizon $[0, T]$ into discrete times

$$0 = t_0 < t_1 < \cdots < t_N = T, \qquad \Delta t_n = t_{n+1} - t_n.$$

On each short interval, the value process experiences only a small amount of drift, a small Brownian fluctuation, and possibly a jump. Because these effects accumulate gradually, the BSDE can be integrated over the interval $[t_n, t_{n+1}]$ to relate the option value at the two ends of the step.

Integrating the BSDE over a small step shows that the change in value over that interval is made up of three contributions: the drift term involving rV_t, the Brownian term scaled by Z_t, and the jump terms scaled by the $U_t^{(m)}$. Approximating each quantity by its value at the left endpoint t_n gives the discrete relation

$$V_{n+1} \approx V_n + rV_n\Delta t_n + Z_n\Delta W_n + \sum_{m=1}^{M} U_n^{(m)}\Delta\widetilde{N}_n^{(m)}. \tag{3.5}$$

In this expression, the random increments ΔW_n and $\Delta\widetilde{N}_n^{(m)}$ represent the diffusion and jump shocks that occur between the two time points.

Rewriting Backward from Final Time Since the BSDE describes a backward evolution starting from the terminal payoff, it is helpful to solve (3.5) for V_n rather than V_{n+1}. Doing so yields

$$V_n \approx e^{-r\Delta t_n}\left(V_{n+1} - Z_n\Delta W_n - \sum_{m=1}^{M} U_n^{(m)}\Delta\widetilde{N}_n^{(m)}\right). \tag{3.6}$$

This backward form expresses the idea that the value today should equal the discounted expected value of the option at the next time step, after adjusting for the shocks that occur during the interval.

At maturity, the value is known: $V_N = g(S_N)$. The discrete BSDE then provides a way to work backward from t_N to t_0, step by step, using (3.6). Each application of the backward step replaces the continuous evolution of the BSDE over the interval by a finite update that accounts for discounting, diffusion shocks, and the possibility of jumps.

The discrete approximation captures the essential structure of the continuous-time BSDE. The term involving Z_n represents the effect of small continuous fluctuations, while the terms involving $U_n^{(m)}$ capture the instantaneous impact of jumps. Discounting appears because the BSDE contains the term $-rV_t\,dt$, ensuring that the recursion remains consistent with risk-neutral valuation. Thus, the backward relation (3.6) mirrors the logic of the continuous BSDE but in a form that operates directly on discrete time points.

3.4 Neural Parameterization and Training Algorithm

The discrete BSDE yields a backward relation connecting the option value at two successive time points. Rather than storing V_n and the jump coefficients U_n on a grid, we represent both quantities with a neural network depending on the current state. At each time step t_n and state S_n, the neural network outputs

$$\mathcal{N}_\theta(t_n, S_n) = \big(V_\theta(t_n, S_n),\ U_\theta(t_n, S_n)\big),$$

where V_θ approximates the continuation value and $U_\theta(t_n, S_n) \in \mathbb{R}^M$ represents the jump sensitivities associated with the M Poisson clocks driving the jump component of the model.

Backpropogate Future Predictions The discrete BSDE takes the form, with the jump term written in matrix form,

$$V_n \approx e^{-r\Delta t_n}\Big(V_{n+1} - Z_n \Delta W_n - U_n^\top \Delta \widetilde{N}_n\Big),$$

where $\Delta \widetilde{N}_n \in \mathbb{R}^M$ collects the compensated Poisson increments on $[t_n, t_{n+1}]$. In the neural approximation we replace

$$V_n \approx V_\theta(t_n, S_n), \qquad V_{n+1} \approx V_\theta(t_{n+1}, S_{n+1}), \qquad U_n \approx U_\theta(t_n, S_n).$$

The backward step therefore uses the network's prediction at the next time step to form a training target for the current time step. To incorporate the early-exercise feature of the

American option, the continuation value predicted by the network is compared with the immediate exercise payoff. Denote the continuation value by

$$V_n^{\text{cont}} = V_\theta(t_n, S_n).$$

If the immediate exercise is optimal at time t_n, then the option value should satisfy

$$V_n = g(S_n),$$

where $g(\cdot)$ is the exercise payoff. Thus, after evaluating the network at (t_n, S_n), we set the training label to

$$V_n^{\text{label}} = \max\bigl(V_n^{\text{cont}},\, g(S_n)\bigr).$$

This substitution ensures that the backward recursion respects the early-exercise constraint at every step: if exercising is optimal on a given path and time, the label is replaced by the payoff; otherwise, the continuation value is retained. In this way, the American option logic is embedded directly into the training procedure, without the need to introduce an explicit reflection process.

The BSDE Residual as the Loss Function To enforce the discrete BSDE, all terms are moved to the left-hand side, producing the residual

$$\text{Res}_V = V_\theta(t_n, S_n) - e^{-r\Delta t_n}\Bigl(V_\theta(t_{n+1}, S_{n+1}) - Z_n \Delta W_n - U_\theta(t_n, S_n)^\top \Delta \widetilde{N}_n\Bigr).$$

If the model satisfies the BSDE perfectly, this quantity is met with perfect equality for every path. Therefore, we consider this quantity as loss to minimize over, defined as a mean squared residual,

$$\mathcal{L}(\theta) = \mathbb{E}\bigl[\text{Res}_V^2\bigr].$$

Thus the loss is exactly the discrete BSDE written with all terms on one side; minimizing $\mathcal{L}(\theta)$ drives the network output toward a solution of the backward dynamics.

Algorithm Summary Training follows a backward-in-time procedure that mirrors the numerical evaluation of the BSDE.

Algorithm 1 Training Algorithm for the Discrete Jump–Diffusion BSDE

```
1: Simulate $M$ sample paths of $(S_n, \Delta W_n, \Delta \widetilde{N}_n)$ on the grid $\{t_0, \dots, t_N\}$.
2: Initialize terminal values:
       $$V_N^{\text{label}} = g(S_N)$$
3: for $n = N - 1$ down to 0 do
4:     Evaluate the network at the current state:
           $$(V_\theta, U_\theta) = \mathcal{N}_\theta(t_n, S_n).$$
5:     Form the discrete BSDE residual:
           $$\text{Res}_V = V_\theta(t_n, S_n) - e^{-r\Delta t_n}\Big(V_{n+1}^{\text{label}} - Z_n \Delta W_n - U_\theta(t_n, S_n)^\top \Delta \widetilde{N}_n\Big).$$
6:     Update network parameters by minimizing
           $$\mathcal{L}(\theta) = \mathbb{E}\big[\,\text{Res}_V^2\,\big].$$
7:     After optimization at step $n$, evaluate the network again to obtain the continuation
       value:
           $$V_n^{\text{cont}} = V_\theta(t_n, S_n).$$
8:     Apply American early-exercise logic by comparing continuation and payoff:
           $$V_n^{\text{label}} = \max\big(V_n^{\text{cont}},\ g(S_n)\big).$$
9: end for
```

In this way, the neural approximation (V_θ, U_θ) is trained to satisfy the discrete-time BSDE by directly minimizing its residual. Because the same network predicts both the continuation value and all M jump sensitivities, the method retains efficiency while capturing the full structure of the jump component.

3.5 Neural Network Architecture

To approximate the mappings

$$(t_n, S_n) \longmapsto \big(V_n,\ U_n\big)$$

required by the discrete BSDE introduced above, we employ a deep neural network whose structure follows the ideas of Chen and Wan [6], but extends their formulation by adding a dedicated output head for the jump–sensitivity vector. The model predicts the continuation value $V_\theta(t_n, S_n)$ and the M-dimensional jump vector $U_\theta(t_n, S_n)$ that appear in the discrete backward relation derived in the previous subsection.

Shared trunk with two output heads We choose a model depth of 7 fully connected layers with a relu activation function. This follows the deep learning framework in Chen and Wan [6] with a refinement where we split the last layer into two separate heads. The first head with the task of learning the scalar option value, and the second head for the vector of jump sensitivities. We choose the shared head network for its smaller size as we do not need to duplicate shared behaviour. The network thus consists of a shared trunk of six fully connected layers, followed by two independent output layers:

$$\mathcal{N}_\theta(t_n, S_n) = \big(V_\theta(t_n, S_n),\ U_\theta(t_n, S_n)\big), \qquad U_\theta(t_n, S_n) \in \mathbb{R}^M.$$

The shared trunk encodes the common nonlinear dependence on the state, while the two heads specialize to the diffusion and jump components respectively. The hidden-layer width is chosen as

$$H = \max\{\, d + 5,\ M\,\}$$

where d is the number of underlying assets in S. This ensures s sufficiently expressive capacity even when the number of jump clocks exceeds the number of underlying assets, which may occur in the multi-asset jump constructions discussed earlier.

Input features Consistent with the BSDE discretization, the network takes as input the tuple

$$\big(S_n,\ V_{\text{prestep}},\ g(S_n)\big),$$

where V_{prestep} denotes the value produced at the most recent forward point along the same simulated path, and $g(S_n)$ is the payoff feature as introduced in the background chapter. Chen and Wan [6] showed that such features stabilize training near maturity, and our formulation follows the same principle.

Time–decay outputs In order to increase stability of the learned quantities, we reformulate the output of the model to be the difference from the prestep value

$$\tau_V(t_n, S_n) \in R, \qquad \tau_U(t_n, S_n) \in R^M,$$

which are the incremental updates for V and U respectively.

Value prediction as an incremental update Rather than predicting $V_\theta(t_n, S_n)$ directly, we use the incremental update formulation of Chen and Wan [6]:

$$V_\theta(t_n, S_n) = \alpha_V \Big(V_{\text{prestep}} + \Delta t_n \, \delta \; \tau_V(t_n, S_n) \Big),$$

where α_V is a learnable scalar, Δt_n is the time step, and δ is a fixed scaling constant (not trained). In practice, this update is only applied every few time steps, and batch normalization is incorporated in the shared trunk; see [6] for implementation details.
The jump–response vector is obtained through an analogous formula:

$$U_\theta(t_n, S_n) = \alpha_U \Big(U_{\text{prestep}} + \Delta t_n \, \delta \; \tau_U(t_n, S_n) \Big),$$

where α_U is a separate learnable scalar. The temporal scaling Δt_n and the factor δ match the value update to maintain consistent time normalization across the diffusion and jump components, while allowing the two outputs to learn independently through their distinct time–decay outputs τ_V and τ_U.

The architecture mirrors the structure of the BSDE developed in the preceding subsections: the network does not learn V_n and U_n directly, but instead learns local updates designed to satisfy the backward relation. The shared representation captures the essential state dependence, while the two output heads and their associated time–decay outputs produce the distinct components required by the diffusion and jump terms of the model.

Chapter 4

Results

4.1 Training Data

All numerical experiments in this chapter are trained on simulated trajectories of the multi-asset Merton jump–diffusion described in Chapter 3. For each configuration (d, λ, K, M), we generate $P = 240{,}000$ independent sample paths on a uniform time grid with N discrete points, drawn from sampling paths based on our specified correlated jumps structure. Each path contains the full set of Brownian increments, compensated Poisson increments, jump indicators, and post-jump states required by the discrete BSDE relation. For each point in time. To ensure consistency across methods, the Monte Carlo benchmarks are evaluated on exactly the same simulated paths and time grid as the neural network. The resulting dataset provides both the forward trajectories for evaluating the BSDE residual and the backward labels for enforcing the American early-exercise condition. This unified simulation framework ensures that any pricing differences observed in the results arise from model quality rather than differences in underlying training data.

4.2 High-Dimensional Pricing Accuracy

A primary objective of this work is to demonstrate that the jump-enhanced BSDE framework can price American-style options accurately and stably in high-dimensional settings. Thus, this is a feasibility study. Unless otherwise specified, all experiments in this section use a geometric basket call with maturity $T = 1.0$, $N = 25$ exercise dates, risk-free rate $r = 0$, diffusion volatility $\sigma = 0.2$, equicorrelation $\rho = 0.75$, and a Merton jump–diffusion with intensity λ and common jump clock $J \sim \log\mathcal{N}(0.005, 0.1^2)$. Wall-clock times were recorded on a single NVIDIA A100 GPU. We focus on the geometric call due to the stability of its PDE solution; however, all results also apply to other basket options whose payoff is determined solely at a given point in time, for example, arithmetic average or maximum options.

Pricing results produced by the neural BSDE solver are compared against two benchmarks. The primary benchmark is a high-precision Monte Carlo dynamic programming estimate evaluated on the same time grid and the same state projections used by the BSDE method. This is done to allow for the same fixed grid on which American early exercise is possible. A secondary benchmark, used for consistency checks, is obtained from an American PIDE solver applied to the one-dimensional geometric reduction of the basket, choosing to benchmark with the geometric basket option allows us to compare against PIDE solver. Agreement between the Monte Carlo and PIDE prices confirms the validity of both reference calculations.

To assess performance in moderate dimension, we evaluate configurations with $d \in \{3, 5\}$ and jump intensities $\lambda \in \{0.5, 1.0, 1.5, 2.0\}$ for an at-the-money strike $K = 1.0$. Table 4.1 presents the neural prices, Monte Carlo benchmarks, absolute pricing errors, and wall-clock times for all completed runs. The errors remain on the order of 10^{-4} across all tested values of λ and show no indication of deterioration as dimension increases.

Table 4.1: Neural vs. Monte Carlo Pricing Across Dimensions for $K = 1.0$.

d	λ	V_{NN}	V_{MC}	$\lvert V_{\text{NN}} - V_{\text{MC}}\rvert$	Wall Time (s)
3	0.5	0.073975	0.073794	1.81e-04	873.3
	1.0	0.076526	0.076173	3.53e-04	841.8
	1.5	0.078978	0.078634	3.44e-04	861.8
	2.0	0.081235	0.080894	3.41e-04	860.8
5	0.5	0.072248	0.072174	7.40e-05	822.3
	1.0	0.074812	0.074631	1.81e-04	868.4
	1.5	0.077394	0.077136	2.58e-04	857.2
	2.0	0.079751	0.079360	3.91e-04	850.4

To evaluate performance in very high dimension, we next consider fixed jump intensity $\lambda = 1.0$ and vary the dimension from $d = 10$ up to $d = 100$. All model parameters remain as above. The results appear in Table 4.2. The neural prices remain within a few 10^{-4} of the Monte Carlo benchmarks across all dimensions. In addition, the wall-clock time grows smoothly with d and exhibits no signs of instability or explosive computational cost.

Table 4.2: High-dimensional Pricing Accuracy for a Geometric Basket Call

d	V_{NN}	V_{MC}	$\lvert V_{\mathrm{NN}} - V_{\mathrm{MC}}\rvert$	Wall-clock (s)
10	0.073605	0.073489	0.000116	833.30
25	0.072870	0.072776	0.000094	880.56
50	0.072671	0.072446	0.000225	1007.24
100	0.072706	0.072579	0.000127	1661.94

The results in Tables 4.1 and 4.2 demonstrate that pricing accuracy does not degrade even as the dimension reaches $d = 100$. The absolute error is consistently comparable to the inherent Monte Carlo uncertainty present in the benchmark. Runtime increases with dimension as expected, but the growth is mild and free of numerical instabilities.

Taken together, these experiments show that the addition of jumps to the BSDE framework achieves accurate and numerically stable American option pricing across a wide range of dimensions. The ability to maintain accuracy in settings such as $d = 50$ and $d = 100$ highlights the effectiveness of the neural network in controlling the nonlocal jump contribution and enabling scalability that would be inaccessible to classical PIDE or Monte Carlo–BSDE methods.

4.3 Delta Accuracy in High Dimension

Accurate deltas are essential for hedging and risk management, particularly in high-dimensional settings where even small gradient errors can propagate into substantial hedging discrepancies. This section evaluates the quality of the deltas produced by the jump BSDE solver across dimensions ranging from $d = 3$ to $d = 100$. The experimental setting is identical to that of Section 4.2: we consider a geometric basket call with maturity $T = 1.0$, $N = 25$ exercise dates, risk-free rate $r = 0$, diffusion volatility $\sigma = 0.2$, equicorrelation $\rho = 0.75$, and a Merton jump–diffusion with intensity $\lambda = 1.0$ and common jump clock $J \sim \log \mathcal{N}(0.005, 0.1^2)$. The strike is fixed at $K = 1.0$.

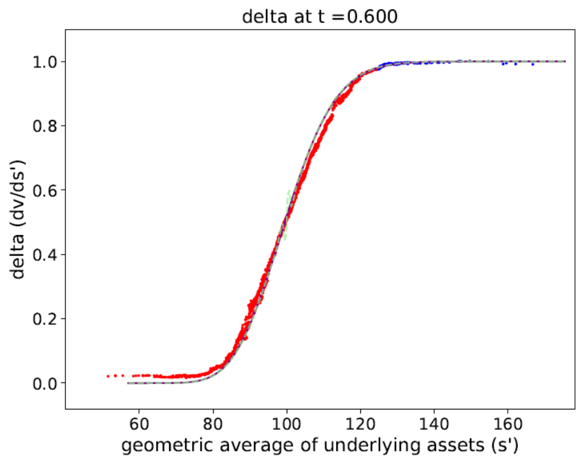
Deltas are benchmarked against Monte Carlo pathwise estimates evaluated on the same time grid and under the same jump–diffusion dynamics. For symmetric geometric baskets, the benchmark delta is effectively the same across all components at points where the basket is near the money, and serves as a scalar reference value Δ_{MC} for all coordinates. The neural-network deltas $\Delta_{\mathrm{NN},i}$ obtained from the BSDE solver can therefore be summarized by their mean $\bar{\Delta}_{\mathrm{NN}}$ across coordinates, together with a mean absolute componentwise deviation Err_{Δ} relative to the Monte Carlo benchmark.

Table 4.3 reports these quantities for dimensions $d \in \{3, 5, 10, 25, 50, 100\}$. The deltas scale as expected with dimension—approaching $1/d$ in the moderately in-the-money region—and the neural estimates are consistently close to the Monte Carlo benchmark. The componentwise error remains below 10^{-3} across all dimensions, and does not deteriorate as d increases.

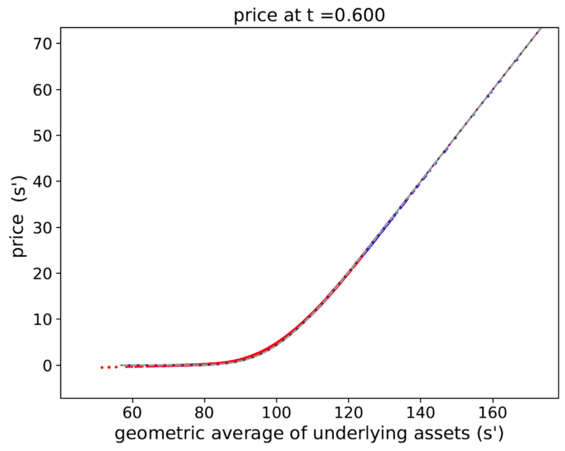
Table 4.3: Delta Accuracy for the Geometric Basket Call with $\lambda = 1.0$ and $K = 1.0$

d	$\bar{\Delta}_{\mathrm{NN}}$	Δ_{MC}	Err_{Δ}
3	0.333378	0.333310	0.000068
5	0.104756	0.103198	0.001558
10	0.052167	0.051330	0.000837
25	0.020788	0.020532	0.000256
50	0.010382	0.010264	0.000118
100	0.005232	0.005145	0.000087

For reference, an example plot of delta and price for a 100-dimensional call option is shown below, with a PDE computed solution overlaid.



(a) Neural delta vs. geometric average at $t = 0.6$.



(b) Value function vs. geometric average at $t = 0.6$.

Figure 4.1: Pricing and Delta Estimation for $d = 100$

4.4 Error Behaviour Across Time and Dimension

To understand how approximation error propagates through the backward recursion, we examine the evolution of pricing error across timesteps as well as its dependence on the state dimension. All statistics reported here are averaged across the full set of strikes $K \in \{0.5, 0.75, 1.0, 1.25, 1.5\}$ and jump intensities $\lambda \in \{0.5, 1.0, 1.5, 2.0\}$, and therefore represent the general behaviour of the solver rather than sensitivity to any particular configuration.

The average RMSE between the neural BSDE price and the Merton American reference at each timestep is shown in Table 1. Note that since we are in the *backward* setting, times here represent the time backwards starting at maturity. The error is largest near maturity, where the payoff introduces nonsmooth features and steep curvature that are challenging for any regression-based method. As one moves backward in time, the RMSE decreases and settles into a stable regime over most of the time horizon. This pattern reflects the smoothing effect of conditional expectation in the BSDE recursion and indicates that the method does not accumulate numerical instability as the recursion progresses.

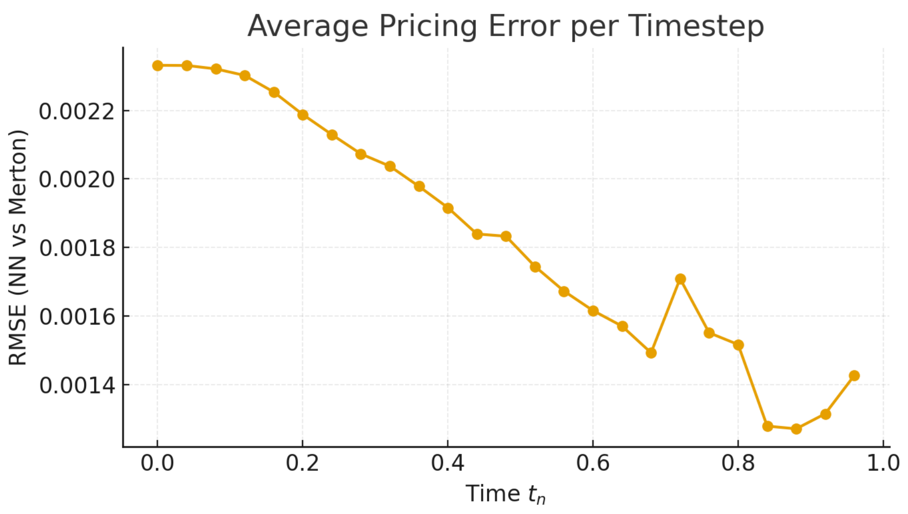


Figure 4.2: Average RMSE between the neural BSDE estimate and the Merton American reference as a function of timestep t_n, averaged over all strikes and jump intensities.

To assess how accuracy varies with dimension, we compute the mean RMSE over all strikes and jump intensities for each dimension d. Table 4.4 summarizes the results. The error remains within the 10^{-3} range from $d = 3$ through $d = 100$, with no indication of a monotone increase. The absence of deterioration in high dimension is notable given the typical variance amplification associated with jumps in classical BSDE schemes.

Table 4.4: Average pricing RMSE as a function of dimension d, aggregated over all strikes and jump intensities.

Dimension d	Mean RMSE
3	0.00160
5	0.00177
7	0.00184
10	0.00137
25	0.00453
50	0.00283
100	0.00390

Together, the timestep- and dimension-wise analyses show that the pricing errors remain small and well structured throughout the backward induction. The backward recursion does not amplify noise, and high-dimensional configurations behave in a manner that is consistent with both theoretical expectations and the empirical behaviour observed in the pricing and delta results of earlier sections.

4.5 Interaction Between Jump Intensity and Time Discretization

The final set of experiments examines how accuracy depends jointly on the jump intensity λ and the time discretization $\Delta t = T/N$. This interaction is fundamental in jump–diffusion models: larger values of λ correspond to more jumps being sampled, while smaller values of Δt reduce discretization error in the BSDE residual. If $\lambda\Delta t$ is too large then numerical instability may ensue. Understanding their combined influence helps identify stable operating regimes for the solver.

We fix the geometric basket model at $d = 5$ with strike $K = 1.0$, maturity $T = 1.0$, equicorrelation $\rho = 0.75$, and Merton jump–diffusion dynamics with common lognormal jump factor $J \sim \log\mathcal{N}(0.005, 0.1^2)$. For time-step counts we choose $N \in \{5, 10, 20, 30\}$ and for jump intensities we consider $\lambda \in \{1, 2, 4, 8\}$. All other parameters and training configurations remain consistent with earlier experiments. Table 4.5 presents the neural and Monte Carlo prices, absolute errors, and wall-clock times for all configurations.

Table 4.5: Pricing accuracy for a $d = 5$ geometric basket call with $K = 1.0$

λ	N	V_{NN}	V_{MC}	$\lvert V_{\text{NN}} - V_{\text{MC}} \rvert$	Wall-clock (s)
1.0	5	0.079754	0.079506	2.48e-4	147.3
1.0	10	0.079447	0.079269	1.78e-4	337.5
1.0	20	0.079398	0.079137	2.61e-4	654.0
1.0	30	0.079342	0.079152	1.90e-4	1009.3
2.0	5	0.088921	0.088625	2.96e-4	145.9
2.0	10	0.088410	0.088379	3.10e-5	318.4
2.0	20	0.088495	0.087984	5.11e-4	670.8
2.0	30	0.088340	0.088001	3.39e-4	983.8
4.0	5	0.105118	0.104919	1.99e-4	148.8
4.0	10	0.104431	0.104341	9.00e-5	340.3
4.0	20	0.104221	0.103887	3.34e-4	663.6
4.0	30	0.104034	0.103603	4.31e-4	994.4
8.0	5	0.133045	0.112110	2.09e-2	145.5
8.0	10	0.130990	0.131056	6.60e-5	329.7
8.0	20	0.130592	0.130276	3.16e-4	655.5
8.0	30	0.130034	0.129760	2.74e-4	1028.4

For fixed λ, the absolute pricing error decreases as the time grid is refined. This behaviour is consistent with the improved approximation of the BSDE residual at smaller Δt. For fixed N, the error increases with λ, reflecting the greater difficulty of representing the more frequent jumps. Even for the high-intensity case $\lambda = 8$, accurate pricing is recovered once N is sufficiently large. The accuracy across all cases aligns with the qualitative prediction that the effective resolution parameter $\lambda\Delta t$ governs the difficulty of the problem: when $\lambda\Delta t$ is small, the jump component is captured smoothly by the network and the solver enters a stable regime.

Taken together, these experiments show that the BSDE framework is robust across a wide range of jump intensities and time discretizations. Accuracy improves systematically with refinement of the time grid, and the method remains stable even in regimes where the jump activity is very high.

Chapter 5

Conclusion

This thesis presented a scalable framework for pricing American-style options under multi-asset jump–diffusion dynamics using a BSDE formulation designed to remain numerically stable even in very high dimensions. Extensive numerical experiments demonstrated that the approach produces accurate American option values and stable deltas across a range of jump intensities, time discretizations, and dimensions up to $d = 100$, while maintaining agreement with both Monte Carlo and PIDE benchmarks. These results show that high-dimensional American options with realistic jump components can be addressed effectively within this BSDE-based framework.

With regard to future extensions of this work, one natural extension is the development of *explicit jump-hedging strategies* within the BSDE setting. Although jump risk is only partially hedgeable in incomplete markets, trading environments often include liquid instruments that provide partial protection against discontinuous movements, such as short-dated options, crash-protection derivatives, or variance-linked products. Incorporating these instruments into the backward recursion, instead a general jump hedge, would allow the solver to learn hedging strategies that manage both diffusive and discontinuous risk, and would enable a quantitative assessment of the marginal value of jump hedging under different model specifications.

A second major direction is the extension of the methodology to *infinite-activity Lévy models*, such as CGMY, Variance Gamma, and other processes exhibiting heavy tails or infinite variation. These models capture fine-scale structure in asset returns that finite-activity jump models cannot represent. In such settings, the jump contribution becomes an integral over the full Lévy measure rather than a finite sum of compound Poisson events, requiring new representations for the jump operator and new training strategies to ensure numerical stability. The results of this thesis suggest that learning-based BSDE methods have the potential to scale to these more demanding regimes, provided that the Lévy integral is approximated in a controlled and computationally efficient manner.

Further extensions may incorporate market frictions, more general early-exercise features, or hybrid payoff structures in which discontinuities interact with path-dependent elements such as barriers or callable schedules. Each of these settings introduces additional mathematical challenges, but they also broaden the practical relevance of the method for applications in equity, energy, credit, and insurance markets.

Overall, this thesis demonstrates that high-dimensional American option pricing with jumps can be carried out accurately and efficiently within a modern learning-based BSDE framework. By establishing numerical stability, scalability, and robust hedging behaviour across a wide range of model configurations, the work lays the foundation for a broader class of high-dimensional derivatives models in which discontinuities and complex state dynamics must be treated simultaneously. Continued development in these directions—especially toward more realistic jump laws and the hedging of discontinuous risks—offers considerable potential for advancing computational methods in quantitative finance.

References

[1] Benjamin Avanzi, Luke C. Cassar, and Bernard Wong. Modelling dependence in insurance claims processes with lévy copulas. *ASTIN Bulletin*, 41(2):575–609, 2011.

[2] Benjamin Avanzi, Greg Taylor, and Bernard Wong. Common shock models for claim arrays. *ASTIN Bulletin*, 48(3):1109–1136, 2018.

[3] Guy Barles, Rainer Buckdahn, and Etienne Pardoux. Backward stochastic differential equations and integral-partial differential equations. *Stochastics and Stochastics Reports*, 60(1–2):57–83, 1997.

[4] Bruno Bouchard and Nizar Touzi. Discrete-time approximation and monte carlo simulation of backward stochastic differential equations. *Stochastic Processes and their Applications*, 111(2):175–206, 2004.

[5] Mark Broadie and Jerome Detemple. American option valuation: New bounds, approximations, and a comparison of existing methods. *The Review of Financial Studies*, 9(4):1211–1250, 1996.

[6] Yangang Chen and Justin W. L. Wan. Deep neural network framework based on backward stochastic differential equations for pricing and hedging american options in high dimensions. *Quantitative Finance*, 21(2):217–246, 2021.

[7] Rama Cont and Peter Tankov. *Financial Modelling with Jump Processes*. Chapman & Hall/CRC, 2004.

[8] Angelos Dassios and Hongbiao Zhao. Exact simulation of hawkes process paths. *Methodology and Computing in Applied Probability*, 15(3):673–685, 2013.

[9] Thibaut Deschatre and Xavier Warin. A common shock model for multidimensional electricity intraday price modelling with application to battery valuation. *Quantitative Finance*, 2024.

[10] Nicole El Karoui, Shige Peng, and Marie-Claire Quenez. Backward stochastic differential equations in finance. *Mathematical Finance*, 7(1):1–71, 1997.

[11] Paul Glasserman. *Monte Carlo Methods in Financial Engineering*, volume 53 of *Stochastic Modelling and Applied Probability*. Springer, 2004.

[12] Emmanuel Gobet, Jean-Philippe Lemor, and Xavier Warin. A regression-based monte carlo method to solve backward stochastic differential equations. *The Annals of Applied Probability*, 15(3):2172–2202, 2005.

[13] Filip Lindskog and Alexander J. McNeil. Common poisson shock models: Applications to insurance and credit risk modelling. *ASTIN Bulletin*, 33(2):209–238, 2003.

[14] Robert C. Merton. Option pricing when underlying stock returns are discontinuous. *Journal of Financial Economics*, 3(1–2):125–144, 1976.

[15] Bernt Øksendal. *Stochastic Differential Equations: An Introduction with Applications.* Springer, 6 edition, 2003.

[16] Etienne Pardoux and Shige Peng. Adapted solution of a backward stochastic differential equation. *Systems & Control Letters*, 14(1):55–61, 1990.

[17] Huyên Pham. *Continuous-Time Stochastic Control and Optimization with Financial Applications*, volume 61 of *Stochastic Modelling and Applied Probability.* Springer, 2009.

[18] Eckhard Platen and David Heath. *A Benchmark Approach to Quantitative Finance.* Springer, 2006.

[19] Steven E. Shreve. *Stochastic Calculus for Finance II: Continuous-Time Models.* Springer, 2004.

[20] Nizar Touzi. *Optimal Stochastic Control, Stochastic Target Problems, and Backward SDEs.* Springer, 2012.

[21] Paul Wilmott, Sam Howison, and Jeff Dewynne. *The Mathematics of Financial Derivatives.* Cambridge University Press, 1995.

[22] Jianfeng Zhang. A numerical scheme for backward stochastic differential equations. *The Annals of Applied Probability*, 14(1):459–488, 2004.

APPENDICES

Table 1: Average pricing error per timestep across all strikes and jump intensities. RMSE and maximum absolute error are reported between the neural BSDE solution and the Merton American benchmark.

Step n	t_n	RMSE_n	$\max_n \lvert V_{\text{NN}} - V_{\text{MC}} \rvert$
0	0.00	6.82e-04	1.78e-02
1	0.04	7.38e-04	1.56e-02
2	0.08	7.04e-04	1.69e-02
3	0.12	7.96e-04	1.67e-02
4	0.16	8.47e-04	1.72e-02
5	0.20	8.20e-04	1.80e-02
6	0.24	8.87e-04	1.86e-02
7	0.28	9.46e-04	2.00e-02
8	0.32	9.17e-04	1.93e-02
9	0.36	9.86e-04	2.00e-02
10	0.40	9.92e-04	1.98e-02
11	0.44	1.07e-03	2.03e-02
12	0.48	1.06e-03	1.97e-02
13	0.52	1.09e-03	2.09e-02
14	0.56	1.04e-03	1.97e-02
15	0.60	1.09e-03	2.01e-02
16	0.64	1.16e-03	2.02e-02
17	0.68	1.17e-03	1.97e-02
18	0.72	1.19e-03	2.03e-02
19	0.76	1.27e-03	2.04e-02
20	0.80	1.52e-03	2.08e-02
21	0.84	1.28e-03	1.77e-02
22	0.88	1.27e-03	2.02e-02
23	0.92	1.32e-03	2.27e-02
24	0.96	1.43e-03	3.16e-02