OptionRL: Estimating with Differential Equations (Draft ver.)

Dongsheng Hou*
Department of Computer Science and Engineering
Southern University of Science and Technology
12410421@mail.sustech.edu.cn

Yanqiao Chen*
Department of Computer Science and Engineering
Southern University of Science and Technology
12412115@mail.sustech.edu.cn

February 4, 2026

Contents

1 Introduction

2 Related Works

3 OptionRL

3.1 Black-Scholes-Merton Model

In the field of financial mathematics, the Black-Scholes-Merton (BSM) model is a foundational framework for (European) option pricing. It assumes that the price of the underlying asset follows a geometric Brownian motion with constant volatility and drift. The BSM model provides a closed-form solution for European-style options.

They derived a partial differential equation (PDE) that the option price must satisfy, known as the Black-Scholes equation.

Theorem 3.1 (Black-Scholes Equation)**.** *The price of a European call option $C(S,t)$ on a non-dividend-paying stock satisfies the following PDE:*

$$\frac{\partial C}{\partial t} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 C}{\partial S^2} + rS\frac{\partial C}{\partial S} - rC = 0 \tag{1}$$

where $C(S,t)$ is the price of the option at time t when the underlying asset price is S, σ is the volatility of the underlying asset, and r is the risk-free interest rate.

By applying Ito's Lemma and constructing a riskless portfolio, they eliminated the stochastic component and derived the pricing formula for European call options.

Theorem 3.2 (Black-Scholes-Merton Formula)**.** *The price of a European call option $C(S_t,t)$ on a non-dividend-paying stock is given by:*

$$C(S_t,t) = S_t\Phi(d_1) - Ke^{-r(T-t)}\Phi(d_2) \tag{2}$$

where:

$$d_1 = \frac{\ln(S_t/K) + (r + \frac{1}{2}\sigma^2)(T-t)}{\sigma\sqrt{T-t}}, \quad d_2 = d_1 - \sigma\sqrt{T-t} \tag{3}$$

Here, $C(S_t,t)$ is the price of a European call option at time t, S_t is the current price of the underlying asset, K is the strike price, r is the risk-free interest rate, σ is the volatility of the underlying asset, T is the time to maturity, and $\Phi(\cdot)$ is the cumulative distribution function of the standard normal distribution.

Rationale In Reinforcement Learning, the agent interacts with an environment to maximize cumulative rewards. However, in real-world scenarios, rewards can be sparse and delayed, making it challenging for agents to learn effective policies, leading to high variance in value estimates and slow convergence. To address this, we propose using the BSM model to shape rewards based on the agent's current state and time-to-go, providing more informative feedback and guiding the agent's learning process.

In a RL task, we have to estimate the value function $V(s)$ or action-value function $Q(s,a)$, usually within a certain period of time. For instance, in Monte-Carlo methods, we estimate the

expected return over an episode, while in Temporal-Difference (TD) learning, we estimate the value function based on one-step transitions. This is analogous to option pricing, where the option's value depends on the underlying asset price and time to maturity. These method often suffer from slow convergence rate due to sparse rewards, e.g., in website bug mining, the agent only receives a reward when a bug is found, which may take a long time.

We assume that the noise in the environment follows a log-normal distribution, similar to asset price movements in financial markets. By mapping the agent's state to a proxy asset price and time-to-go to time-to-maturity, we can compute a potential function using the BSM formula. Thus, though the environment may not provide frequent rewards, the agent can still receive continuous feedback through the potential-based shaping rewards derived from the BSM model, which helps reduce variance and accelerates learning.

3.2 Merton Jump Model

In financial markets, asset prices often exhibit sudden and significant changes, known as jumps, which cannot be captured by the standard Black-Scholes model. To address this limitation, Robert C. Merton extended the Black-Scholes framework by incorporating jump processes into the asset price dynamics, leading to the Merton Jump-Diffusion Model. The Merton Jump-Diffusion Model assumes that the underlying asset price follows a stochastic process that combines both continuous diffusion and discrete jumps. The asset price dynamics under the Merton model can be described by the following stochastic differential equation (SDE):

$$dS_t = \mu S_t dt + \sigma S_t dW_t + J_t S_t dN_t \tag{4}$$

where:

- S_t is the asset price at time t.
- μ is the drift rate of the asset price.
- σ is the volatility of the continuous component.
- W_t is a standard Brownian motion.
- N_t is a Poisson process with intensity λ, representing the number of jumps up to time t.
- J_t is the jump size, typically modeled as a log-normal random variable.

Remark 3.1. *Such model can be also viewed as a Levy process, which generalizes Brownian motion by allowing for jumps.*

The Merton Jump-Diffusion Model leads to a modified option pricing formula that accounts for the possibility of jumps in the underlying asset price.

Theorem 3.3 (Merton Jump-Diffusion Option Pricing Formula)**.** *The price of a European call option $C(S_t, t)$ under the Merton Jump-Diffusion Model is given by:*

$$C(S_t, t) = \sum_{n=0}^{\infty} \frac{e^{-\lambda(T-t)}(\lambda(T-t))^n}{n!} C_{BS}(S_t, t; \sigma_n) \tag{5}$$

where:

- $C_{BS}(S_t, t; \sigma_n)$ *is the Black-Scholes price of the option with adjusted volatility* $\sigma_n = \sqrt{\sigma^2 + \frac{n\delta^2}{T-t}}$, *where* δ *is the standard deviation of the jump size.*
- λ *is the jump intensity.*
- T *is the time to maturity.*
- n *is the number of jumps.*

Rationale While the Black-Scholes model assumes continuous price movements, real-world environments often exhibit sudden changes or jumps, leading to fat-tailed reward distributions.

To better capture these dynamics, we propose using the Merton Jump-Diffusion Model for reward shaping in Reinforcement Learning. By incorporating jump processes, the Merton model provides a more accurate representation of environments with abrupt changes

3.3 Algorithm Framework

The overall algorithm framework for OptionRL using Merton Potential Shaping is outlined in Algorithm 1.

Algorithm 1 OptionRL via Merton Potential Shaping

Require: State space $\mathcal{S}$, Action space $\mathcal{A}$, Goal Threshold K
Require: Hyperparameters: Volatility σ, Jump Intensity λ, Risk-free rate r
Ensure: Optimal Policy π^*

Initialize $Q(s, a)$ arbitrarily for all $s \in \mathcal{S}, a \in \mathcal{A}$
Initialize Potential $\Phi(s) = 0$
function GetMertonPotential(s)
    Map state s to proxy asset price S_t (e.g., semantic proximity)
    Map state s to time-to-go T
    $d_1 \leftarrow \frac{\ln(S_t/K)+(r+\frac{1}{2}\sigma^2)T}{\sigma\sqrt{T}}$
    $P_{BS} \leftarrow S_t\Phi_{norm}(d_1) - Ke^{-rT}\Phi_{norm}(d_1 - \sigma\sqrt{T})$ ▷ Vanilla Black-Scholes
    $P_{Merton} \leftarrow P_{BS} \cdot (1 + \lambda T)$ ▷ Approx. Jump Premium
    return P_{Merton}
end function
for each episode $1 \dots M$ **do**
    Initialize state s
    repeat
        Choose action a from s using ϵ-greedy policy derived from Q
        Take action a, observe reward r_{env} and next state s'
        $\Phi_t \leftarrow$ GetMertonPotential(s)
        $\Phi_{t+1} \leftarrow$ GetMertonPotential(s')
        $F_t \leftarrow \gamma\Phi_{t+1} - \Phi_t$ ▷ Calculate Shaping Reward
        $r_{total} \leftarrow r_{env} + F_t$
        $Q(s, a) \leftarrow Q(s, a) + \alpha[r_{total} + \gamma \max_{a'} Q(s', a') - Q(s, a)]$
        $s \leftarrow s'$
    until s is terminal
end for