A Reinforcement Learning Based Approach to Pricing and Hedging Financial Derivatives

The Usage of Discretization to Absorb Mishedging Risk into a Q-Learner Algorithm

Russell Kim

Advisor: Mete Soner

Submitted in partial fulfillment
of the requirements for the degree of
Bachelor of Science in Engineering

Department of Operations Research and Financial Engineering
Princeton University
United States of America
June 2020

Acknowledgements

This thesis was a huge undertaking, and was completed under unusual circumstances, as Covid-19 spread across the world. First and foremost, I would like to thank my advisor, Prof Mete Soner, as his guidance and insightful questions have been an invaluable resource. Most of this thesis would not be possible without his wisdom. I'd also like to give many thanks to my close friends who have also provided valuable discussion - you know who you are. I also couldn't have done any of this without my family. Throughout my four years at Princeton, I have been pushed to my limit and experienced more than I thought I could ever experience, and my family has been the rock that has kept me grounded all this time. I hope I made you proud.
Finally, I'd like to thank Stack Overflow. If you know you know.

Abstract

Options pricing is a notoriously tricky subject to maneuver, both mathematically and empirically. However, the rise of reinforcement learning in recent times has opened up a new avenue to pricing options through machine learning. As a result, this paper explores and extends the discrete-time option pricing model by Halperin [11]. More specifically, we focus on the off-policy algorithm well known as the *Q-Learning* algorithm. In order to transform the original problem of pricing a Black Scholes based derivative, we stray from the academic continuous-time limit and use a discretized version of the classic Black Scholes model in order to construct a risk-neutral Markov Decision Process. Here, the option price of the MDP is an optimal Q-function. Pricing and hedging an option when the stock dynamics - such as the volatility - are unknown is now possible through the use of the Q-Learning algorithm. The model itself does not make any guesses or assumptions about the structure of the data it is given, and is guaranteed to asymptotically converge to the optimal solution given enough data and time. Empirical results show that the pricing of European options is very close to the classical Black-Scholes results, although there are some notable differences and error deviations when controlling for different factors. This model can also be easily extended to practically any type of financial option, given an easily tractable or estimated payoff function. Extensive numerical testing is done, and specific recommendations of further research are given.

Contents

1 Introduction

Reinforcement learning is a subfield of machine learning that has exploded in popularity over the past decade, not unlike the similarly meteoric rise of neural networks. Despite all the interest from academia, however, literature on various applications is sparse, especially in mathematical finance. What this model provides is an environment in which to test different RL algorithms for financial applications, most notably options hedging and pricing. As a result, this implementation can be thought of as a "guinea pig" of sorts, much like the well known cartpole problem in physics as examined by Peng, Jing and Williams [15]. The hope is that somewhere down the line, a future work will implement this environment into a package much like OpenAI Gym [4].

This paper implements the data-driven Q-Learning model(QLBS) suggested by Halperin [11] for derivatives pricing and and extends it by introducing ideas of global risk hedging and control of learning parameters. Furthermore, it attempts to compare the QLBS model with other, well established options pricing models in use and lays the groundwork to extend the types of options to American, Asian, Barrier, and other higher dimensional asset classes.

The classical method of pricing BSM models in real world applications uses the well known Black Scholes equation [1] to price options and derivatives. The core idea of this model is to represent stock prices as functions of geometric brownian motion - and then, to represent option portfolios as continuous rehedges between risk-free cash and purchases of stock. As a result, every single portfolio can theoretically be represented as a combination of stock and cash in a bank account that is rebalanced continuously by either purchasing more stock or selling more stock *with only the cash in the bank account* (this is called the self-financing assumption), giving the strange impression that options are *redundant*, a fact clearly refuted by the existence of a multi-billion dollar options industry. However, there are several important assumptions made in this model that can be wildly unrealistic in comparison to how markets work in practice, such as

- Volatility Risk
- Bid-ask spreads
- Transactional costs
- Mis-hedging risk

In this paper, we focus mainly on the last item - mis-hedging risk - mostly because this slippage occurs most prominently when the most unrealistic assumption, *continuous rehedging*, is dropped. This obviously reflects real life markets more accurately, as rebalancing of option replication portfolios happens with a finite frequency. Rehedging a portfolio frequently may not be worth it in some cases, as transactional costs can eat up a big portion of any potential profit. In fact, a formal continuous-time limit may not exist when transaction costs are added as it leads to an infinite set of possible option prices due to an infinite number of rebalances. As a result, this finite rehedging frequency often leads to *mis-hedging risk*, or *slippage risk*, which the option buyer or seller should be compensated for appropriately. Since this risk premium is obviously dependent on the individual seller/buyer's risk preferences, we see that the option price depends on this risk preference, et ceteris paribus. (Transactional costs can easily be applied to this model by simply penalizing a reward function every time the algorithm rehedges.)

Therefore, the shift from continuous-time to discrete-time finance leads to a problem where we can optimize the minimization of this slippage risk, with each rehedging period as one time step. Over the natural lifetime of this portfolio, we can reformulate this problem into a sequential decision process where we minimize the slippage cost. This is the core idea behind options pricing in practice, and there are many ways to do this, such as

- Stochastic volatility models
- Finite difference methods
- Optimal stopping methods

However, since the core idea behind all these methods remains the same when pricing options (minimizing slippage), we can see that reinforcement learning is a method that can also be applied to this. There are two main ways to go about this. First, if we choose to build an internal model of stock price dynamics, the QLBS algorithm can be used for approximate dynamic programming or model-based reinforcement learning. Second, in the absence of reliable stock price dynamics, we can choose to have the QLBS model learn the optimal hedge and price *directly from real or synthetic trading data.* This can be done by using a value iteration RL algorithm to approximate the Bellman optimality equation. Although this vanilla model only needs one variable (stock prices), it can easily be extended to cover other costs by penalizing the reward function appropriately. This paper will focus on the latter model, and numerical experiments comparing how the QLBS algorithm does to other popular options pricing formulas are shown later with various types of options. This provides several advantages:

- No need to construct an internal model of stock prices
- Works with high dimensionality of state spaces ($n > 4$)
- Only data needed is past trading data of portfolios (real or synthetic)
- Complicated options can be priced just by changing the payoff function

Converting the BSM model into a discrete setting through the Q-Learner has one more surprising result - the QLBS model contains both the option price and its delta in one mathematical object (the Q function) whereas the continuous time BSM model has two different formulas, both of which include transcendental functions like $\mathcal{N}(d_1)$ or $\mathcal{N}(d_2)$.

In Sect.2, I present a review of some current methods of pricing various financial derivatives, including the classical BSM model and popular stochastic volatility models. A review of reinforcement learning is covered in Sect.3, along with the basic conceptual framework that Q-Learning specifically falls under. Sect. 4 is dedicated to deriving a fair measure of risk and an optimal hedge strategy based on that definition of risk through Monte Carlo simulation. The next two sections explain both the DP (Dynamic Programming) approach - which only works if transition probabilities and stock dynamics are known - and the Q-Learner approach put forth by Watson. Finally, we conclude with a number of numerical experiements and suggestions for possible future works.

2 Literature Review

Ever since the celebrated Black-Scholes model was published, there has been a myriad of literature published on how to best improve upon the model, given some of its more unrealistic assumptions. For all of the methods listed below, we assume that stock prices (given by S_t) are lognormal and driven by geometric brownian motion.

2.1 Black-Scholes Model

The value of a call option for a non-dividend-paying underlying stock in terms of the Black-Scholes parameters is:

$$\begin{aligned} C\left(S_t, t\right) &= N\left(d_1\right) S_t - N\left(d_2\right) PV(K) \\ d_1 &= \frac{1}{\sigma\sqrt{T-t}}\left[\ln\left(\frac{S_t}{K}\right) + \left(r + \frac{\sigma^2}{2}\right)(T-t)\right] \\ d_2 &= d_1 - \sigma\sqrt{T-t} \\ PV(K) &= Ke^{-r(T-t)} \end{aligned}$$

The price of a corresponding put option based on put-call parity is:

$$\begin{aligned} P\left(S_t, t\right) &= Ke^{-r(T-t)} - S_t + C\left(S_t, t\right) \\ &= N\left(-d_2\right) Ke^{-r(T-t)} - N\left(-d_1\right) S_t \end{aligned}$$

For both, as above:
·$N(\cdot)$ is the cumulative distribution function of the standard normal distribution
·$T-t$ is the time to maturity (expressed in years)
·S_t is the spot price of the undertying asset
·K is the strike price
·r is the risk free rate (annual rate, expressed in terms of continuous compounding)
·σ is the volatility of returns of the underlying asset

This is the "vanilla" model that we will test our QLBS model against.

2.2 Finite Difference Model

This model (Forsythe, Wolfgang, 1960) [9] results by taking finite differences of a Taylor expansion of linear ordinary differential equations or non-linear partial differential equations to approximate its derivatives. This essentially creates a fine "mesh" in which we can approximate the function for each discretized interval. Out of simplicity's sake, take any linear ODE or nonlinear PDE function f.

The forward equation is

$$f(x+h) = f(x) + hf'(x) + \frac{1}{2}h^2 f''(x) + \dots$$

and the backwards equation is

$$f(x-h) = f(x) - hf'(x) + \frac{1}{2}h^2 f''(x) - \dots$$

The most commonly used differential is the central approximation difference operator, given by

$$f'(x) = \frac{f(x+h) - f(x-h)}{2h} + O\left(h^2\right)$$

The second partial derivative is given by:

$$f''(x) = \frac{f(x+h) - 2f(x) + f(x-h)}{h^2} + O\left(h^2\right)$$

As a result, this reduces hard-to-solve differential equations into a system of equations which can be easily solved with matrix algebra, which makes this technique very suited to modern computing power. Consequently, the finite difference model is one of the most dominant approaches to approximating numerical solutions of PDEs and ODEs. Using the simple heat equation as an example, we review some popular and widely used FDMs.

Consider, for the sake of simplicity, the normalized one dimensional heat equation with Dirchlet boundary conditions:

$$\begin{aligned} U_t &= U_{xx} \\ U(0,t) &= U(1,t) = 0 (\text{boundary condition}) \\ U(x,0) &= U_0(x) (\text{initial condition}) \end{aligned}$$

The FDM method uniformly partitions the domain in space using a mesh from $x_0, ..., x_J$ and in time using a mesh $t_0, ..., t_N$ such that

$$\begin{aligned} x_{j+1} - x_j &= h, j = 0, ..., J-1 \\ t_{n+1} - t_n &= k, n = 0, ..., N-1 \end{aligned}$$

and the individual points are given by

$$u(x_j, t_n) = u_j^n \simeq u(x_j, t_n)$$

2.2.1 Explicit method

Using a forward difference at time t_n and a second-order difference for the space derivative at position x_j(FTCS), we get the recurrence equation:

$$\frac{u_j^{n+1} - u_j^n}{k} = \frac{u_{j+1}^n - 2u_j^n + u_{j-1}^n}{h^2}$$

This is an explicit method for solving the one-dimensional heat equation. We can obtain u_j^{n+1} from the other values this way:

$$u_j^{n+1} = (1-2r)u_j^n + ru_{j-1}^n + ru_{j+1}^n$$

where $r = \frac{k}{h^2}$. Since we know the values at time $= n$, we can use this recurrence relation and the given boundary conditions to calculate the values at time $= n+1$. This is known to be numerically stable and convergent when $r \leq \frac{1}{2}$. The errors are also proportional to the time step and the square of the space step as follows:

$$\Delta u = O(k) + O(h^2)$$

The explicit scheme is the least accurate and can be unstable, but is also the easiest to implement and the least numerically intensive.

2.2.2 Implicit method

If we use the backward difference at time $t_n + 1$ and a second order central difference for the space derivative at position x_j (BTCS), we obtain an analagous recurrence equation:

$$\frac{u_j^{n+1} - u_j^n}{k} = \frac{u_{j+1}^{n+1} - 2u_j^{n+1} + u_{j-1}^{n+1}}{h^2}$$

This is an implicit method for solving the heat equation. Solving a system of linear equations, we can obtain u_j^{n+1}:

$$(1+2r)u_j^{n+1} - ru_{j-1}^{n+1} - ru_{j+1}^{n+1} = u_j^n$$

The scheme is always numerically stable and convergent but usually more numerically intensive than the explicit method as it requires solving a system of numerical equations on each time step. The errors are linear over the time step and quadratic over the space step:

$$\Delta u = O(k) + O\left(h^2\right)$$

The implicit scheme works the best for large time steps.

2.2.3 Crank-Nicolson method

For this method (Crank, Nicolson, 1947) [7], we use the central difference at time $t_{n+\frac{1}{2}}$ and a second-order central difference for the space derivative at position x_j (CTCS) to get the recurrence equation:

$$\frac{u_j^{n+1}-u_j^n}{k}=\frac{1}{2}\left(\frac{u_{j+1}^{n+1}-2u_j^{n+1}+u_{j-1}^{n+1}}{h^2}+\frac{u_{j+1}^{n}-2u_j^{n}+u_{j-1}^{n}}{h^2}\right)$$

Solving a system of linear equations, we can solve for u_j^{n+1}

$$(2+2r)u_j^{n+1}-ru_{j-1}^{n+1}-ru_{j+1}^{n+1}=(2-2r)u_j^n+ru_{j-1}^n+ru_{j+1}^n$$

Although this method is guaranteed to be numerically stable and convergent, it is also much more computationally intensive as it requires solving a system of equations at every time step. The errors are quadratic for both the time step and the space step.

$$\Delta u = O\left(k^2\right)+O\left(h^2\right)$$

The Crank-Nicolson method is usually the most accurate scheme for small time steps.

2.3 Stochastic Volatility Models

The core idea behind this model challenges the usual assumption that volatility is taken to be constant - numerous studies have proven this assumption to be false by looking at past market data. In order to combat this, we express the volatility itself as a stochastic process. The maximum likelihood of the volatility at different times t is given by:

$$\begin{aligned}\widehat{\sigma}^2 &= \left(\frac{1}{n}\sum_{i=1}^{n}\frac{\left(\ln S_{t_i}-\ln S_{t_{j-1}}\right)^2}{t_i-t_{i-1}}\right)-\frac{1}{n}\frac{\left(\ln S_{t_n}-\ln S_{t_0}\right)^2}{t_n-t_0}\\ &= \frac{1}{n}\sum_{i=1}^{n}(t_i-t_{i-1})\left(\frac{\ln\frac{S_{t_i}}{S_{t_{i-1}}}}{t_i-t_{i-1}}-\frac{\ln\frac{S_{t_n}}{S_{t_0}}}{t_n-t_0}\right)^2\end{aligned}$$

so the expected value is

$$\mathrm{E}\left[\widehat{\sigma}^2\right]=\frac{n}{n-1}\sigma^2$$

For a stochastic volatility model, we replace the constant volatility σ with a function ν_t, that models the variance of S_t. This variance function is also modeled as Brownian motion, and the form of ν_t depends on the particular SV model under study.

$$dS_t=\mu S_t dt+\nu_t S_t dW_t$$

$$d\nu_t=\alpha_{\nu,t}dt+\beta_{v,t}dB_t$$

where $\alpha_{v,t}$ and $\beta_{v,t}$ are some functions of ν, and dB_t is another standard gaussian that is correlated with dW_t with constant correlation factor ρ.

There are several popular models in use today in both academia and practice, given below.

2.3.1 Heston Model

The popular Heston model (Heston, 1993) [13] is a commonly used SV model, in which the randomness of the variance process varies as the square root of variance. In this case, the differential equation for variance takes the form:

$$d\nu_t = \theta\left(\omega - \nu_t\right)dt + \xi\sqrt{\nu_t}dB_t$$

where ω is the mean long-term variance, θ is the rate at which the variance revard its long-term mean, ξ is the volatility of the variance process, and dB_t is, like dW_t, a brownian motion with zero mean and $\sqrt{\nu_t}$ standard deviation. However, dW_t and dB_t are correlated with the correlation value ρ. In other words, the Heston SV model assumes that the variance is a random process that

- Exhibits a tendency to revert towards a long-term mean ω at a rate θ
- Exhibits a volatility proportional to the square root of its level
- Exhibits a source of randomness that is correlated (by ρ) with the randomness of the underlying price's process

2.3.2 CEV Model

The CEV model (Cox, 1975 [6] and Linetsky et al., 2010 [14]) describes the relationship between volatility and price:

$$dS_t = \mu S_t dt + \sigma S_t^{\gamma} dW_t$$

Intuitively, this explains the behaviour in some markets, like commodities, in which volatility rises with price. As a result, some literature on this model describes it as a "local volatility" model as it is not suitable for all market types.

2.3.3 GARCH Model

The Generalized Autoregressive Conditional Heteroskedasticity (GARCH) model (Brennan, 1979 [3] and Haukkson et al.,2001 [12] is another popular model(for estimating stochastic volatility. It assumes that the randomness of the variance process varies with the variance, as opposed to the square root of the variance as in the Heston model. The standard GARCH(1,1) model has the following form for the variance differential:

$$d\nu_t = \theta\left(\omega - \nu_t\right)dt + \xi\nu_t dB_t$$

This has led to several derivative models to be developed, such as NGARCH, TGARCH, IGARCH, LGARCH, EGARCH, GJR-GARCH.

3 Survey of Reinforcement Learning

In reinforcement learning, we deal with the problem of sequential decision modeling under a specified time horizon. In every RL problem, we have an agent that interacts with its environment in the form of actions. These actions lead to rewards and the goal is to usually maximize a future cumulative or terminal reward.

3.1 Framework

The main ingredients in RL are states, or mathematical objects that contain all the information about the state of the system up until time t. They are defined as such:

$$x_t = f(o_1, r_1, a_1, ..., a_{t-1}, o_t, r_t)$$

where

- o_t is the observation at time t
- r_t is the reward given after observing the environment at time t
- a_t is the action taken after processing the reward given at time t

A reinforcement learning agent has several major components, such as its

- Policy: the agent's behaviour function, or how the agent decides to what to do after receiving information about the rewards
- Value function: a relation between each state/action and its 'goodness' for the agent
- Model: the agent's representation(if any) of the environment

3.2 Policies

A **policy** is a function that maps from state to action.
A deterministic policy is given by:

$$a = \pi(x)$$

A stochastic policy is given by:

$$\pi(a|x) = \mathbb{P}[a|x]$$

A Q-value function gives the expected future total reward and it is usually given by

$$Q^\pi(x, a) = \mathbb{E}[r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + ...|x, a]$$

where

- (x, a) is the state and action

- π represents the policy
- $\gamma \in (0, 1)$ is the discount factor

A value function can also be defined using the Bellman equation:

$$Q^{\pi}(x, a) = \mathbb{E}_{x',a'}[r + \gamma Q^{\pi}(x', a')|x, a]$$

Now, for an optimal Q-value function $Q^{*}(x, a) = \max_{\pi} Q^{\pi}(x, a)$, the policy function in this case is deterministic, since the policy is unique. Therefore, the Bellman equation becomes

$$Q^{*}(x, a) = \mathbb{E}_{x'}[r + \gamma \max_{a'} Q^{*}(x', a')|x, a]$$

3.3 Dynamic Programming

We can use one of two types of algorithms to solve the above unknown expectation in general: value iteration and policy iteration.

3.3.1 Value Iteration

Value iteration involves initiating a random value function, which we then optimize iteratively. The main idea is to calculate the expected sum of rewards for a certain time horizon when taking the optimal action at each time.

Starting with $V_0^*(x) = 0, \forall x$

$$V_{t+1}^*(x) = \max_a \Sigma_{x'} T(x,a,x')[R(x,a,x') + V_t^*(x')]$$

where $T_t(x,a,x') = \mathbb{P}[X_{t+1} = x' | x_t = x, a_t = z]$, or the probability of the next state being x' given the current state and action taken. This process of optimizing sequential decisions requires the iterative calculation of the optimal value function - which is very computationally intensive - and then we extract the optimal policy from this via the optimality Bellman operator. This is the type of algorithm that Q-Learning falls under. A representation of how the algorithm reaches its optimal policy is given below:

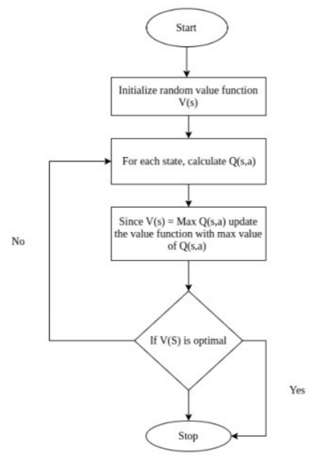

Start
Initialize random value function V(s)
For each state, calculate Q(s,a)
Since V(s) = Max Q(s,a) update the value function with max value of Q(s,a)
If V(S) is optimal
No
Yes
Stop

Figure 1: Value Iteration

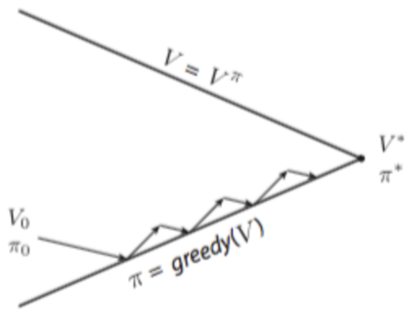


Figure 2: VI Solution Path

3.3.2 Policy Iteration

The second type of algorithm is of policy iteration, where we repeat two steps until our policy converges:

- *Policy evaluation:* Calculate (nonoptimal) values for some fixed policy until convergence.
- *Policy improvement:* Update the policy using a one-step look-ahead (expected sum of discounted future rewards) with the resulting converged utilities as future values.

We continue these two steps until the values are optimal. However, we note that in this type of algorithm, we calculate the value of taking actions as our policy dictates. It looks at each possible state and changes the policy's state values towards the values of the states that the policy's action will transition to until the values stop changing (at which point it is considered converged and optimal). The important difference is that this value function is *not optimal.* It is only used for evaluating the computationally intensive task of extracting a new iterative policy, which requires maximization over actions. This, however, happens relatively infrequently and policies often converge relatively quickly.

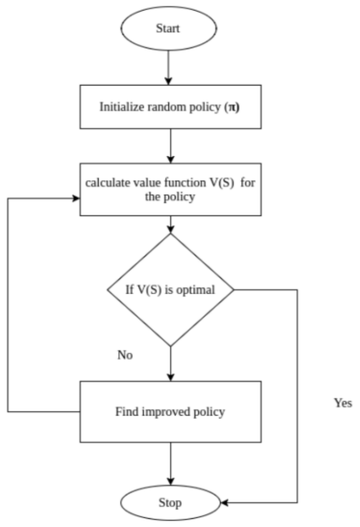

Start
Initialize random policy (π)
calculate value function V(S) for the policy
If V(S) is optimal
No
Yes
Find improved policy
Stop

Figure 3: Policy Iteration

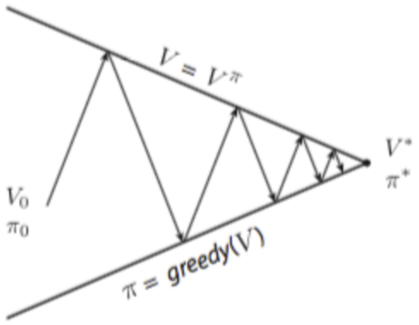


Figure 4: PI Solution Path

4 Discrete-time Black Scholes model

As mentioned above, the first step is to reformulate the Black-Scholes process into a sequential risk minimization process and transform it from a continuous state process to a discrete state process. The question of exactly *how* to do this is a question still being researched. The main problem is that the issue of exactly how to define "risk" in the market is still an open question. Approaches were pioneered by FÃűllmer and Schweizer [8], Schweizer [16], and Cerny and Kallsen [5]. We draw from a Ph.D thesis from Grau, 2007 [10], which in itself draws inspirations from physicists Potters and Bouchaud, 2003 [2]. This method essentially views pricing from the point of a market seller of a European put with maturity T and the terminal payoff of $H_T(S_T)$ at maturity. To hedge the option, the seller uses proceeds of the sale to set up a replicating hedged portfolio made of stock S_t and a risk-free bank deposit B_t.

4.1 Motivation behind a Pricing Model

Before we start with the derivation of our pricing model, it is important to consider the various factors that go into making a pricing model relatively realistic (obviously, we exclude market frictions, transactional costs, etc and focus only on the dynamics of options pricing).

We first note that a large OTC market exists where positions of options are traded. Some of these are exotic, and therefore rarely traded, making a market price virtually unobservable. Others are traded more regularly, but suffer from large ask-bid spreads (which may be subjective and on the whim of the risk preference of the options dealer). Even the ask price itself may be substantially different across multiple dealers due to different market participants using different models, since the classical BSM model is usually not sufficient. These traditional models instead try to determine a "fair market value" plus a risk premium, or a spread, that the bank wants to earn. However, as noted by Grau [10], it is uncertain whether this "fair market value" is *guaranteed* to cover the price of the hedging strategy itself (mostly cost of capital).

The reason why this is important is that traditional strategies utilizing techniques such as

- Maximal utility
- Minimal risk
- Market price of risk

cannot ensure that the price is expected to be greater than the cost of the hedging strategy due to a *mistreatment of risk itself.* A fair price is usually determined by a utility function of the bank, or a market price of risk. But, on the other hand, the bank must pay for the risk of keeping money in a margin account, and therefore, paying of the risk of her portfolio. This margin account will have to cover losses using the bank's funds, which in turn must earn the equity return to be profitable. We use the four fundamental properties listed in Grau and discuss them.

1. **A method which delivers the price of a derivative also delivers a corresponding hedging strategy, which an issuer can follow.** This is a feature of realistic markets - the dealer of the option must be able to replicate the option using a hedging strategy

2. **The price of a derivative, on average, at least covers all costs occurring to the dealer.** The average cost of a derivatives trade must be estimated, otherwise the company's earnings could not be accurately estimated.

3. **The hedging strategy reduces the risk of the dealer.** The important thing to note is that this hedging strategy should *not* be designed to maximize profits or utilities. Instead, it should be to minimize the required capital in the margin account since the bank has finite funds, and the cost of capital is relatively expensive in this case.

4. **Any realistic, physical market model is allowed in this pricing model.** This is required so that the dealer can put as much information as possible into the derivative price - the dealer should have specific knowledge of the underlying stock and its market and incorporate this into the pricing of the option itself.

Following these principles, we are now ready to construct a Monte Carlo framework in which to emulate the Black-Scholes model. Then, we show that the resulting recurrence equations and our choice of a symmetric measure of risk make it possible to construct a hedging strategy that amounts to a quadratic maximization problem. Transaction costs and ask-bid spreads will be discussed in a later section.

4.2 Preliminary assumptions

4.2.1 Classical Markowian Portfolio Theory

A portfolio at time t is represented by

$$\Pi_t = u_t S_t + B_t \tag{1}$$

where

- Π_t = Value of Portfolio
- u_t = The amount of stock held
- S_t = Price of the stock held
- B_t = Value of a risk-free bank deposit

Using this notation, we set the terminal value of the portfolio at time T. This is because we can convert all the stock to cash at the end of the time period, hence the function is measurable at time T:

$$\Pi_T = B_T = H_T\left(S_T\right) \tag{2}$$

4.2.2 Self Financing Constraint

An important assumption for the model is the self financing constraint. It says that any rehedges done must be done with no external flow of money. As such, it implies the following relation about the value of the portfolio given a rehedge at time $t+1$:

$$u_t S_{t+1} + e^{r\Delta t} B_t = u_{t+1} S_{t+1} + B_{t+1} \tag{3}$$

This condition ensures that any future changes in the hedge portfolio should be performed using the liquid assets in the bank.

As a result, we can find a recursive relationship for the amount of money to keep in the bank at any time $t < T$.

$$B_t = e^{-r\Delta t}\left[B_{t+1} + (u_{t+1} - u_t)\, S_{t+1}\right], \quad t = T-1, \ldots, 0 \tag{4}$$

Using this equation and inserting it into the previous formula gives us a recursive relationship for the portfolio value at time $t < T$.

$$\Pi_t = e^{-r\Delta t}\left[\Pi_{t+1} - u_t \Delta S_t\right], \quad \Delta S_t = S_{t+1} - e^{r\Delta t} S_t, \quad t = T-1, \ldots, 0 \tag{5}$$

The above formulas imply that B_t and Π_t are both not Lebesque-meseaurable, as they depend on future random unknown quantities. As a result, B_0 and Π_0 are both random variables with some unknown distribution. For any given hedging strategy $\{u_t\}_{t=0}^{T}$, we can use Monte Carlo simulation to simulate N paths $S_1 \rightarrow S_2 .... \rightarrow S_N$, and evaluate Π_t at each time step working backwards using the backwards recursive relation given above. Since our choice of hedging strategy *does not* affect the underlying process, we can simulate the N paths once and use the same paths for any hedging strategies we might want to use in the future.

In summary, we want to do one forward pass of simulations $S_1 \rightarrow S_2 \rightarrow \ldots \rightarrow S_N$ and then use backward recursion as formulated above to apply a certain hedging strategy $\{u_t\}_{t=0}^{T}$ by back propagating uncertainty in the future and recasting it in terms of uncertainty today. As the dealer of these options wants to price the options *today*, this is exactly what we are looking for. However, the question of what this optimal hedge strategy exactly **is** is still up in the air - the next section addresses this.

4.3 Optimal Hedging Strategy

Although it would be nice to have a "one size fits all" trading strategy, the reality of trading in real life means that we must treat liquidly traded derivatives and illiquidly traded derivatives differently. We examine the first case.

4.3.1 Hedging and Pricing of a Liquidly Traded Derivative

A liquidly traded derivative means that competition is firm, and therefore ask-bid spreads are relatively low. For frequently traded derivatives (such as a simple European put on an S&P 500 stock), the price is simply

$P_{\text{ask}} = P_{\text{ask,market}} + \epsilon$

where $P_{\text{ask,market}}$ is the market ask price of the option and ϵ is the transactional cost plus whatever ask-bid spread the dealer of the option wants to earn. Then, the hedging in this case is simply to buy the derivative for $P_{ask,market}$.

However, this model falls apart for semiliquid/illiquid derivatives.

4.3.2 Hedging and Pricing of an Illiquidly Traded Derivative

There are three parts to this process.

Deriving the Expected Cost of Hedging an Illiquid Portfolio

We consider an isolated example in which the dealer of the option sells this derivative and holds no other positions in the market. We can see from property 1 above, that we need a discrete-time model, since hedging can only happen a finite number of times. Note that this does not affect the computation of the option's payoff $H(S_t)$ since we can just use finite discrete samples of S_t at certain times. Considering a simple portfolio

$$\Pi_t = u_t S_t + B_t \tag{6}$$

using the same notation as before, the dealer has to pay the payoff of the derivative at maturity, $t = T$. That is, the trader sells the hedge

$$u_T = 0 \tag{7}$$

and the bank compensates the payoff to the holder of the option

$$B_T = H(S_T) \tag{8}$$

As we mentioned before, Π_T is measurable, so

$$\Pi_T = H(S_T) \tag{9}$$

Using the same recursion method as before, we see that the bank at time t should be able to compensate for the money required for the rehedging at time $t+1$ by the self financing constraint, such that

$$B_t = e^{-r(\Delta t)}(B_{t+1} + (u_{t+1} - u_t S_{t+1})) \tag{10}$$

where $(u_{t+1} - u_t S_{t+1})$ is the profit from the position in the underlying stock. Since we know that B_T is measurable at maturity as well, we can use this equation to derive the value of the hedge portfolio

$$\begin{aligned}\Pi_t &= B_t + uS_t \\ &= e^{-r(\Delta t)}(\Pi_{t+1} - u_t(S_{t+1} - e^{-r(\Delta t)}S_t))\end{aligned}$$

Finally, the expected cost of the hedging strategy is given by

$$C_0 = \mathbb{E}[\Pi_0] \tag{11}$$

Again, we remind ourselves that the hedge portfolio equation is not deterministic since the amount in the bank, B_0 denotes the money required to compensate for the hedging costs, *including the hedging error*. Since the exact path of the underlying stock is impossible to predict, the required amount in the bank is a stochastic process. To determine exactly what strategy $\{u_i\}_{t=0}^{T-1}$ we should take, we have to take into account the objectives of the derivatives department at an options dealer. The second part is actually deriving the option dealer's objective function as a function of their risk preferences.

Deriving the Objective Function of an Illiquidly Traded Derivative

A dealer of options has several objectives, which we must consider before deriving our hedging strategy. The dealer in practice is usually a bank, which has both a derivatives trading department and an investment department. Although the investment department is profitable by actively taking risks, the goal of the derivatives trading department should be instead to *minimize* possible risk. Therefore, traditional investment strategies are not very well suited to derivatives trading strategies. Hence, we state that *the objective of a derivatives hedge should be to minimize a symmetric risk measure.* Most other strategies demand that the drift

$$\mu = \frac{1}{(\Delta t)} \cdot \log \frac{\mathbb{E}\left[\Delta(S_t)\right]}{S_t} \tag{12}$$

of the underlying stock is correctly identified - which is usually extremely difficult. Investment departments usually have an optimal portfolio allocation (determined by the bank and their risk preferences), and if this quantity were known, then over or under-investment in an underlying stock would not be considered in order to make a profit. Usually such an over/under investment in the underlying derivatives does not result in a better portfolio allocation. As a result, we find the need for a hedging strategy that does *not* depend much, if at all, on the drift of the underlying asset. The classical BSM equations fulfill this property perfectly, since the drift μ does not appear at all in the pricing equations. In other models, it is desirable that the more often a hedging occurs, the less influence that the drift μ should have on the option's fair price. We then naturally turn to variance as a simple measure of risk. Variance corresponds to a quadratic utility function of the option dealer, which in turn gives us an "easy" problem to solve, as then the problem becomes one of quadratic maximization/minimization. There exist higher order minimization strategies, but there are several problems with using those metrics, including

- Increased difficulty of evaluation
- Loss of linear superposition of optimal hedges for individual options
- Difficulty of extending this to high-dimensional asset classes

For reference, a strategy minimizing the fourth moment of the error and the properties resulting from this is given by Selmi, Bouchad (2000) [2]. One more property to consider is the process of optimization. Consider for example, an options dealer that loses money due to hedging errors based on some hedging strategy u_i. This does not mean that the dealer should change her strategy to compensate for this - the derivatives department should not base a hedging strategy based on perceived gains/losses of the underlying asset, but to *minimize future risks.* Therefore, only future risk should be hedged, and as a result, a hedge *should be in a forward time horizon, rather than a global time horizon.*

Variance Minimization Strategies

There are three main minimization strategies explored in Grau:

- **Global hedging.** This strategy minimizes the total hedging error, but it has two main problems. It requires the knowledge of transition probabilities (which we often do not know), and it leads to some ramifications, such as a dealer trying to lose gains made in the past and even paying to offset those same gains - this is obviously unrealistic given the profit driven nature of the industry.

- **Local hedging.** This technique minimizes the variance from one time step to the next, and is intended for hedging of options for which we can observe a market price. However, since the transitions from one time step to the next in this method are obviously different from the variances observed in global hedging, we cannot compare these variances directly. The difference in global vs local hedging is analogous to the reinforcement learning concept of *offline learning* vs *online learning.*

- **Forward global hedging.** This is the strategy that we focus on in this paper, and it is a combination of the above two methods.

We choose to focus on the third method, as the first two methods are inherently unstable and do not make sense for a risk metric that is supposed to "look ahead". In a forward global variance minimization, we solve

$$\Pi_t = e^{-r\Delta t}(\Pi_{t+1} - u_t^*(S_{t+1} - e^{-r(\Delta t)}S_t), \Pi_T = H(S_T) \tag{13}$$

with the optimal policy u_t^* given as

$$\begin{aligned} u_t^\star(S_t) &= \arg\min_u \operatorname{Var}\left[\Pi_t | S_t, t\right] \\ &= \arg\min_u \operatorname{Var}\left[\Pi_{t+1} - u_t \Delta S_t | S_t, t\right], \quad t = T-1, \ldots, 0 \end{aligned}$$

Unlike path-wise simulation as with calculating portfolio values, we use *cross-sectional* simulation that simulates every point in time for all Monte Carlo paths. By way of example, if we choose to simulate a million paths, at time t, we know all information about every one of those million paths, but only up to time t (i.e, $T, T-1, ..., t+1, t$), since we cannot know the future when computing a hedge. We let

$$\mathcal{F}_t^{(n)} = \left\{ \left(X_t^{(n)}, a_t^{(n)}, R_t^{(n)}, X_{t+1}^{(n)} \right) \right\}_{t=0}^{T-1}, n = 1, \ldots, N_{MC} \tag{14}$$

which represents the cross-sectional information available up to time t. Then the above optimal hedge becomes

$$\begin{aligned} u_t^*(S_t) &= \arg\min_u \operatorname{Var}\left[\Pi_t | \mathcal{F}_t\right] \\ &= \arg\min_u Var\left[\Pi_{t+1} - u_t \Delta S_t | \mathcal{F}_t\right], \quad t = T-1, \ldots, 0 \end{aligned}$$

This equation implies that all uncertainty in the portfolio value is due to the uncertainty concerning the amount B_t required in the bank to cover any future hedges that may occur. As a result, the optimal hedge should minimize the cost of hedging (capital) for any such position at time t.

We take the derivative of the RHS of the above equation and set it to zero. This gives

$$u_t^\star(S_t) = \frac{\operatorname{Cov}\left(\Pi_{t+1}, \Delta S_t | \mathcal{F}_t\right)}{\operatorname{Var}\left(\Delta S_t | \mathcal{F}_t\right)}, \quad t = T-1, \ldots, 0 \tag{15}$$

Now we have an equation describing the optimal hedge in terms of the variance of the portfolio.

4.4 Fair option pricing

The theoretical fair price for an option, $\hat{C}_t$, is given by

$$\hat{C}_t = \mathbb{E}_t \left[\Pi_t | \mathcal{F}_t \right] \tag{16}$$

However, this is *not* the price that a seller of the option should charge as discussed before. We need to add a "risk premium" (discussed later in this section).

We can use the tower law of conditional expectations to express this fair option price recursively:

$$\begin{aligned}\hat{C}_t &= \mathbb{E}_t \left[e^{-r\Delta t} \Pi_{t+1} | \mathcal{F}_t \right] - u_t \left(S_t \right) \mathbb{E}_t \left[\Delta S_t | \mathcal{F}_t \right] \\ &= \mathbb{E}_t \left[e^{-r\Delta t} \mathbb{E}_{t+1} \left[\Pi_{t+1} | \mathcal{F}_{t+1} \right] | \mathcal{F}_t \right] - u_t \left(S_t \right) \mathbb{E}_t \left[\Delta S_t | \mathcal{F}_t \right] \\ &= \mathbb{E}_t \left[e^{-r\Delta t} \hat{C}_{t+1} | \mathcal{F}_t \right] - u_t \left(S_t \right) \mathbb{E}_t \left[\Delta S_t | \mathcal{F}_t \right], \quad t = T-1, \ldots, 0\end{aligned}$$

Similarly, we can express the aforementioned optimal hedge in terms of $\hat{C}_t$ rather than Π_t.

$$u_t^{\star} \left(S_t \right) = \frac{\operatorname{Cov} \left(\Pi_{t+1}, \Delta S_t | \mathcal{F}_t \right)}{\operatorname{Var} \left(\Delta S_t | \mathcal{F}_t \right)} = \frac{\operatorname{Cov} \left(\hat{C}_{t+1}, \Delta S_t | \mathcal{F}_t \right)}{\operatorname{Var} \left(\Delta S_t | \mathcal{F}_t \right)} \tag{17}$$

As a result, we now get a recursive relation for $\hat{C}_t$:

$$\hat{C}_t = e^{-r\Delta t} \mathbb{E}^{\hat{\mathbb{Q}}} \left[\hat{C}_{t+1} | \mathcal{F}_t \right], \quad t = T-1, \ldots, 0 \tag{18}$$

where $\mathbb{Q}$ is a signed measure with transition probabilities

$$\tilde{q} \left(S_{t+1} | S_t \right) = p \left(S_{t+1} | S_t \right) \left[1 - \frac{\left(\Delta S_t - \mathbb{E}_t \left[\Delta S_t \right] \right) \mathbb{E}_t \left[\Delta S_t \right]}{\operatorname{Var} \left(\Delta S_t | \mathcal{F}_t \right)} \right] \tag{19}$$

A well known problem of quadratic risk minimization schemes is the possibility of negative fair option prices. However, since the dealer of the option will always charge a *risk premium* for assuming the costs of holding a margin account, this is not a significant problem. A possible specification of a risk premium to add on top of the fair option price would be to add the cumulative expected discounted variance of the hedge portfolio along all time steps $t = 0, 1, ..., T$ with a risk-aversion λ, given as such:

$$C_0^{(ask)}(S, u) = \mathbb{E}_0 \left[\Pi_0 + \lambda \sum_{t=0}^{T} e^{-rt} \operatorname{Var} \left[\Pi_t | \mathcal{F}_t \right] | S_0 = S, u_0 = u \right] \tag{20}$$

The term $\sum_{t=0}^{T} e^{-rt} \operatorname{Var} \left[\Pi_t | \mathcal{F}_t \right]$ is the sum of time-discounted variances of portfolio values, called the *risk premium*, and λ is a risk parameter chosen by the option seller's risk preferences, as one has to take into account the risk of overdrawing from the bank when rehedging. While the notion of adding a risk premium that is proportional to the variance of the hedge portfolio was first suggested by Potters and Bouchaud [2] on an intuitive basis, it is important to note that a utility-based approach actually derives this as a quadratic approximation to a utility-based option price - a method that many firms and financial institutions around the world use as aforementioned. This also establishes a relation between a risk aversion parameter λ and the parameter γ in the exponential utility $U(X) = -\exp(-\gamma X)$. As we take the limit $\gamma \to 0$ and make corrections by expanding powers of γ, the utility function approaches the above risk premium (proof not reproduced here).

4.5 Taking the limit to a continuous state

To perform a sanity check, we make sure that the framework above does indeed correspond to the BSM equations when $\Delta t \to 0$. In this limit, the BSM dynamics under the physical measure $\mathbb{P}$ are given by a continuous time GBM with a drift μ and volatility σ:

$$\frac{dS_t}{S_t} = \mu dt + \sigma dW_t \tag{21}$$

where W_t is a standard Brownian motion.

Using the first-order Taylor expansion

$$\hat{C}_{t+1} = C_t + \frac{\partial C_t}{\partial S_t} \Delta S_t + O(\Delta t) \tag{22}$$

to take the optimal hedge strategy to $\Delta t \to 0$, we obtain

$$u_t^{BS}(S_t) = \lim_{\Delta t \to 0} u_t^{\star}(S_t) = \frac{\partial C_t}{\partial S_t} \tag{23}$$

To find the continuous-time limit of the option price

$$\hat{C}_t = \mathbb{E}_t\left[e^{-r\Delta t}\hat{C}_{t+1}|\mathcal{F}_t\right] - u_t(S_t)\,\mathbb{E}_t\left[\Delta S_t|\mathcal{F}_t\right], \quad t = T-1, \ldots, 0 \tag{24}$$

$$\lim_{\Delta t \to 0} u_t(S_t)\,\mathbb{E}_t\left[\Delta S_t|\mathcal{F}_t\right] = \lim_{dt \to 0} u_t^{BS} S_t(\mu - r)dt = \lim_{dt \to 0} (\mu - r) S_t \frac{\partial C_t}{\partial S_t} dt \tag{25}$$

To evaluate the first term, we use the second-order Taylor expansion

$$\begin{aligned}\hat{C}_{t+1} &= C_t + \frac{\partial C_t}{\partial t} dt + \frac{\partial C_t}{\partial S_t} dS_t + \frac{1}{2}\frac{\partial^2 C_t}{\partial S_t^2}(dS_t)^2 + \ldots \\ &= C_t + \frac{\partial C_t}{\partial t} dt + \frac{\partial C_t}{\partial S_t} S_t(\mu dt + \sigma dW_t) + \frac{1}{2}\frac{\partial^2 C_t}{\partial S_t^2} S_t^2\left(\sigma^2 dW_t^2 + 2\mu\sigma dW_t dt\right) + O\left(dt^2\right)\end{aligned}$$

Substituting these terms into the option price equation listed above and using the facts that $\mathbb{E}[dW_t] = 0$ and $\mathbb{E}\left[dW_t^2\right] = dt$, we find that the drift μ disappears from the problem, and we are left with the well-known BSM equation in the limit $\Delta t \to 0$

$$\frac{\partial C_t}{\partial t} + rS_t \frac{\partial C_t}{\partial S_t} + \frac{1}{2}\sigma^2 S_t^2 \frac{\partial^2 C_t}{\partial S_t^2} - rC_t = 0 \tag{26}$$

As a result, if the environment we test on is lognormal, our framework in the previous section becomes exactly the original BSM model in continuous time.

5 Reformulating the Black-Scholes process into an MDP

The next step is to reformulate the results of the previous section in terms of Markov Decision Processes. In order to do this, we work in the physical measure $\mathbb{P}$ rather than the probabilistic measure $\mathbb{Q}$. When the transition probabilities and the reward function are given, we can use a recursive backward Value Iteration method to analytically derive a Bellman optimality equation. Optimizing this turns out to be a maximization of a quadratic function.

However, in the much more likely scenario that we do not know the true stock price dynamics, we can solve a backwards recursion for the Bellman optimality equation with using only samples of data - otherwise known as **Reinforcement Learning**. This can also be solved semianalytically (due to a quadratic reward function) using the **Q-Learning** method put forth by Watkins, which is guaranteed to converge to the optimal hedge for any time step size given enough time and training data. If the stock price dynamics are assumed to be lognormal, then the fair price is given by our optimal value function. It is important to note that although we assume the stock prices are lognormal, nowhere in the Q-Learning algorithm does it *actually use* the fact that the prices are lognormal. That is to say, our policy does not try to "guess" the parameters or build a lognormal model based on the data.

5.1 State variables

The basic assumptions are:

- A frictionless market
- No transaction costs
- Risk-less assets earn the risk-free rate r
- Short selling is allowed without restrictions

Another more conceptual assumption is that the price of the underlying S_t behaves like a geometric Brownian motion. This implies no way of forecasting with certainty, and the asset PDE evolves as such:

$$\begin{aligned} X_t &:= -\left(\mu - \tfrac{\sigma^2}{2}\right) t + \log S_t \\ dX_t &= -\left(\mu - \tfrac{\sigma^2}{2}\right) dt + d\log S_t = \sigma dW_t \end{aligned} \tag{27}$$

where the process S_t has been "demeaned" to result in X_t. From this, we can see that X_t is a standard Brownian motion scaled by σ, and therefore a martingale. The state variable X_t is time uniform and the relation above can be used to map non-stationary dynamics of S_t into stationary dynamics of X_t. These relations can still be applied when the stock price dynamics are not lognormal - however, X_t will not be guaranteed to be a martingale in this situation. It is nevertheless very useful for separating non-stationarity of optimization from the non-stationarity of state variables.

5.2 Value Function

We can now see that it is relatively simple to transform time-dependent state variables S_t in terms of time-homogenous variables X_t. In order to fully embrace the world of Markov Decision Processes, it is necessary to introduce new notation. Denote actions $a_t = a_t(X_t)$ and $u_t = u_t(S_t)$ to take (hedges in this situation).

$$u_t\left(S_t\right) = a_t\left(X_t\left(S_t\right)\right) = a_t\left(\log S_t - \left(\mu - \frac{\sigma^2}{2}\right)t\right) \tag{28}$$

We can generalize this to find a general *hedging strategy* for any state X_t, rather than worrying about individual hedging decisions $a_t(x_t)$ (which should be interpreted as a hedging decision made given x_t, which is an instance of the random state X_t at time t). Therefore, since $\{X_t\}_{t=0}^{T}$ is a time-homogeneous variable, we define a time-dependent policy

$$\pi : \{0, ..., T-1\} \times \mathcal{X} \to \mathcal{A}$$

This is a deterministic policy that maps the time t and the current state $X_t = x_t$ into the action (hedge) $a_t \in \mathcal{A}$:

$$a_t = \pi(t, x_t) \tag{29}$$

In order to optimize an MDP problem, we need to find the *objective*, or *Q-value* function to maximize. In this case, we are trying to minimize the slippage risk between the true fair price of the option and what the price is calculated as. It is obvious that the objective function in this case should be of the *fair option price* given earlier. More specifically, since we are dealing with a value maximization problem, we need to *maximize the negative* of $\hat{C}_t$ (which is essentially the same thing as minimizing the actual price). Thus we define our value function:

$$V_t\left(S_t\right) = -C_t^{(ask)}(S_t, u) = \mathbb{E}_t\left[-\Pi_t - \lambda \sum_{t'=t}^{T} e^{-r\left(t'-t\right)}\operatorname{Var}\left[\Pi_{t'}|\mathcal{F}_{t'}\right]|\mathcal{F}_t\right] \tag{30}$$

We now rewrite this equation in terms of our state variable X_t and our state policy π.

$$\begin{aligned} V_t^{\pi}\left(X_t\right) &= \mathbb{E}_t\left[-\Pi_t\left(X_t\right) - \lambda \sum_{t'=t}^{T} e^{-r\left(t'-t\right)}\operatorname{Var}\left[\Pi_{t'}\left(X_{t'}\right)|\mathcal{F}_{t'}|\,\mathcal{F}_t\right]\right. \\ &= \mathbb{E}_t\left[-\Pi_t\left(X_t\right) - \lambda\operatorname{Var}\left[\Pi_t\right] - \lambda \sum_{t'=t+1}^{T} e^{-r\left(t'-t\right)}\operatorname{Var}\left[\Pi_{t'}\left(X_{t'}\right)|\mathcal{F}_{t'}\right]|\mathcal{F}_t\right] \end{aligned}$$

where

$$-\lambda\mathbb{E}_{t+1}\left[\sum_{t'=t+1}^{T} e^{-r\left(t'-t\right)}\operatorname{Var}\left[\Pi_{t'}|\mathcal{F}_{t'}\right]\right] = \gamma\left(V_{t+1} + \mathbb{E}_{t+1}\left[\Pi_{t+1}\right]\right), \quad \gamma \equiv e^{-r\Delta t} \tag{31}$$

using the definition of the value function with a temporal shift. Rearranging the value equation and using the portfolio process, we obtain the Bellman function for our MDP Black Scholes Model

$$V_t^{\pi}\left(X_t\right) = \mathbb{E}_t^{\pi}\left[R\left(X_t, a_t, X_{t+1}\right) + \gamma V_{t+1}^{\pi}\left(X_{t+1}\right)\right] \tag{32}$$

where the one step time-dependent random reward is as follows:

$$\begin{aligned}R_t\left(X_t, a_t, X_{t+1}\right) &= \gamma a_t \Delta S_t\left(X_t, X_{t+1}\right) - \lambda \operatorname{Var}\left[\Pi_t | \mathcal{F}_t\right], t = 0, \ldots, T-1 \\ &= \gamma a_t \Delta S_t\left(X_t, X_{t+1}\right) - \lambda \gamma^2 \mathbb{E}_t\left[\hat{\Pi}_{t+1}^2 - 2 a_t \Delta \hat{S}_t \hat{\Pi}_{t+1} + a_t^2\left(\Delta \hat{S}_t\right)^2\right]\end{aligned}$$

where we use the following facts:

- $\Pi_t = e^{-r\triangle t}[\Pi_{t+1} - u_t \triangle S_t], \triangle S_t = S_{t+1} - e^{r\triangle t} S_t, t = T-1, ..., 0$ (given above)
- $\hat{\Pi}_{t+1} \equiv \Pi_{t+1} - \bar{\Pi}_{t+1}$, where $\bar{\Pi}_{t+1}$ is the sample mean of all values of Π_{t+1}, and similarly for $\Delta \hat{S}_t$.
- $R_T = -\lambda \operatorname{Var}\left[\Pi_T\right]$ where Π_T is determined by the terminal condition.

(Although this model varies the reward function, there are other approaches to constructing a risk-sensitive MDP by modifying one-step variance penalties, which may be discussed later.)

We see that the expected reward is quadratic with respect to the action a_t:

$$\mathbb{E}_t\left[R_t\left(X_t, a_t, X_{t+1}\right)\right] = \gamma a_t \mathbb{E}_t\left[\Delta S_t\right] - \lambda \gamma^2 \mathbb{E}_t\left[\hat{\Pi}_{t+1}^2 - 2 a_t \Delta \hat{S}_t \hat{\Pi}_{t+1} + a_t^2\left(\Delta \hat{S}_t\right)^2\right] \tag{33}$$

Since our reward function includes variance of the portfolio as a risk penalty, we can modify the one-step reward function in tradition risk-neutral RL frameworks to incorporate this risk penalty. The action value function, or the **Q Function** as it is better known, is similar to the value function, but is conditioned on the current state X_t *and* the initial action or hedge a_t. It then follows a policy π:

$$Q_t^\pi(x, a) = \mathbb{E}_t\left[-\Pi_t\left(X_t\right) | X_t = x, a_t = a\right] - \lambda \mathbb{E}_t^\pi\left[\sum_{t'=t}^{T} e^{-r\left(t'-t\right)} \operatorname{Var}\left[\Pi_{t'}\left(X_{t'}\right) | \mathcal{F}_{t'}\right] | X_t = x, a_t = a\right] \tag{34}$$

The optimal policy $\pi_t^\star\left(\cdot | X_t\right)$ is defined as the policy which maximizes the value function $V_t^\pi\left(X_t\right)$, or alternatively and equivalently, maximizes the action-value function $Q_t^\pi\left(X_t, a_t\right)$

$$\pi_t^\star\left(X_t\right) = \operatorname{argmax}_\pi V_t^\pi\left(X_t\right) = \operatorname{argmax}_{a_t \in \mathcal{A}} Q_t^\star\left(X_t, a_t\right) \tag{35}$$

The optimal value function satisfies the Bellman optimality equation

$$V_t^\star\left(X_t\right) = \mathbb{E}_t^{\pi^*}\left[R_t\left(X_t, u_t = \pi_t^*\left(X_t\right), X_{t+1}\right) + \gamma V_{t+1}^\star\left(X_{t+1}\right)\right] \tag{36}$$

The Bellman optimality equation for the action-value function reads

$$Q_t^*(x, a) = \mathbb{E}_t\left[R_t\left(X_t, a_t, X_{t+1}\right) + \gamma \max_{a_{t+1} \in \mathcal{A}} Q_{t+1}^*\left(X_{t+1}, a_{t+1}\right) | X_t = x, a_t = a\right], t = 0 \tag{37}$$

where the terminal condition at $t = T$ is given by

$$Q_T^\star\left(X_T, a_T = 0\right) = -\Pi_T\left(X_T\right) - \lambda \operatorname{Var}\left[\Pi_T\left(X_T\right)\right] \tag{38}$$

We already know that $\Pi_T = B_T$, since it has been given.

To summarize, we framed the classic BSM model as an MDP such that the Q-value function to be maximized is the negative of the risk-adjusted fair option price, giving explicit formulas for functions like the reward function. Once again, we remind ourselves that the variance term here means the variance of all Monte Carlo paths that terminate in a given state.

5.2.1 Optimal Policy

Now the optimal policy is the one that maximizes the action-value function $Q_t^\pi(x,a)$. It is given by

$$\pi_t^\star\left(X_t\right)=\operatorname{argmax}_\pi V_t^\pi\left(X_t\right)=\operatorname{argmax}_{a_t\in\mathcal{A}} Q_t^*\left(X_t,a_t\right) \tag{39}$$

Then, we use the expected reward (calculated above) and substitute it into the Bellman optimality equation:

$$\begin{aligned}Q_t^*\left(X_t,a_t\right)&=\gamma\mathbb{E}_t\left[Q_{t+1}^*\left(X_{t+1},a_{t+1}^*\right)+a_t\Delta S_t\right]\\&-\lambda\gamma^2\mathbb{E}_t\left[\hat{\Pi}_{t+1}^2-2a_t\hat{\Pi}_{t+1}\Delta\hat{S}_t+a_t^2\left(\Delta\hat{S}_t\right)^2\right],t=0,\ldots,T-1\end{aligned}$$

We follow the standard assumption of BSM that no individual seller/buyer of options produces market impact. Then, we can see that the term $\mathbb{E}_t\left[Q_{t+1}^*\left(X_{t+1},a_{t+1}^*\right)\right]$ depends only on the current action a_t through decomposing the expectation and looking at the conditional probability $p(X_{T+1}|X_ta_t)$. The next state probability can only depend on the current action when there is an impact of the action on the market, which we have just assumed there is not. Therefore, the next state probability does not depend on the current action. As follows, the above Q function is a quadratic function of a_t and we can calculate the hedge $a_t^*(S_t)$ that maximizes this Q function analytically:

$$a_t^*\left(X_t\right)=\frac{\mathbb{E}_t\left[\Delta\hat{S}_t\hat{\Pi}_{t+1}+\frac{1}{2\gamma\lambda}\Delta S_t\right]}{\mathbb{E}_t\left[\left(\Delta\hat{S}_t\right)^2\right]} \tag{40}$$

Taking the limit of this as $\triangle t\to 0$ by using Taylor expansions results in

$$\lim_{\Delta t\to 0}a_t^\star=\frac{\partial\hat{C}_t}{\partial S_t}+\frac{\mu-r}{2\lambda\sigma^2}\frac{1}{S_t} \tag{41}$$

If we set $\mu=r$ (or equivalently $\lambda\to\infty$), two things happen. The limit above becomes identical to the BS delta. Secondly, the analytical solution to the optimal hedge includes the finite $\triangle t$ delta which turns into a local-risk minimization delta given by

$$u_t^*\left(S_t\right)=\frac{\operatorname{Cov}\left(\Pi_{t+1},\Delta S_t|\mathcal{F}_t\right)}{\operatorname{Var}\left(\Delta S_t|\mathcal{F}_t\right)},\quad t=T-1,\ldots,0 \tag{42}$$

What this means is that the quadratic hedging that approximates option delta looks only at how risky a portfolio is when determining the optimal hedge. However, the hedge maximizing the Q-value function above adds a **drift term** $\mathbb{E}[\Pi_t]$ to the objective function in a similar way to Markowitz risk-adjusted portfolio return analysis. This leads to a linear first term in the quadratic expected reward. **Hedges maximizing**

the Q-value function are therefore different from hedges obtained by only minimizing risk.

6 Q-Learning and Iteration

The important issue now is to approximate this unknown expectation seen in Q-function. This is where Reinforcement Learning flexes its muscles - it does *not* assume knowledge of the transition probabilities nor the reward function, and instead relies on samples of data to find an optimal policy. Our setting assumes a batch learning, where we only have access to a block of historically collected data. There is no access to a real-time environment nor a simulator of such an environment. Then, we use Value Iteration RL methods that worked with the same objects (the value function and action-value function) as DP did.

A general Q-function is updated in Q-learning through a simple value iteration update. Assume a general Q-function expressed as an expectation:

$$Q(s_t, a_t) = \mathbb{E}_{s_t,a_t}\left[r_t(s_t,a_t) + \gamma \max_{\hat{a}} Q(s_{t+1}, a_t)\right] \tag{43}$$

We use the Robbins-Monro approximation to obtain

$$Q^{new}(s_t,a_t) \leftarrow \underbrace{Q\left(s_t,a_t\right)}_{\text{old value}} + \underbrace{\alpha}_{\text{learning rate}} \cdot \underbrace{\left(\underbrace{r_t(s_t,a_t)}_{\text{reward}} + \underbrace{\gamma}_{\text{discount factor}} \cdot \underbrace{\max_a Q\left(s_{t+1},a\right)}_{\text{estimate of optimal future value}} - \underbrace{Q\left(s_t,a_t\right)}_{\text{old value}}\right)}_{\text{new value (temporal difference target)}} \tag{44}$$

In our case, however our unknown expectation is

$$Q_t^{\star}(x,a) = \mathbb{E}_t\left[R_t\left(X_t,a_t,X_{t+1}\right) + \gamma \max_{a_{t+1}\in\mathcal{A}} Q_{t+1}^{\star}\left(X_{t+1},a_{t+1}\right) | X_t = x, a_t = a\right], t = 0,\ldots,T-1 \tag{45}$$

In much the same way, we use the Robbins-Monro approximation on the unknown expectation to write out an explicit update rule given the datapoint $\left(X_t^{(n)}, a_t^{(n)}, R_t^{(n)}, X_{t+1}^{(n)}\right)$

$$Q_t^{\star,k+1}\left(X_t,a_t\right) = \left(1-\alpha^k\right) Q_t^{\star,k}\left(X_t,a_t\right) + \alpha^k \left[R_t\left(X_t,a_t,X_{t+1}\right) + \gamma \max_{a_{t+1}\in\mathcal{A}} Q_{t+1}^{\star,k}\left(X_{t+1},a_{t+1}\right)\right] \tag{46}$$

which is guaranteed to asymptotically converge to a true optimal action-value function, without making any assumptions about the data-generating distribution that produced the datapoint (which is a property of the Q-learning algorithm). It takes the datapoint and updates the Q-action-value function for the state-action tuple $\left(X_t^{(n)}, a_t^{(n)}\right)$. Ideally, the algorithm should take in as input

1. Data in the form of historical(or simulated) stock prices and times
2. The risk parameter λ (dependent on how risk averse the option seller/buyer is)
3. The time period
4. Number of MC paths to simulate

5. The type of financial derivative to be priced

and should give as output the optimal Q-function, and therefore the optimal risk-adjusted option price as well.

However, even though the Q-learning algorithm is computationally efficient with respect to other similar RL algorithms, it stil may be too slow in practice to use the pure vanilla model, since we would be taking cross-sectional information from a very large number of simulations at the same time t. This cross-sectional information would not be available for any online learning setting of updating Q-values and optimal hedge rations $a_t^*(X_t)$ on a space grid. The solution to this is to update these values *for all points on the grid simultaneously* by looking at a snapshot of all Monte Carlo paths at time t. This is possible since we are in a **batch-mode** setting of RL, and the data is readily available. We cover this in the next section.

6.1 Batch or Fitted Q-Learning

A much faster way of computing the Q-values and optimal hedge ratios is given by the Fitted Q Iteration algorithm, which uses basis functions to approximate the updated function at each time step. This model is now generalized to a continuous-state space, since the only difference between a continuous-state space and a discrete-state specification is the choice of basis function.

The main idea behind Fitted Q Iteration is that we can model the updated Q-value-action Function as a series of linear regressions, where the regression models themselves are polynomial splines. This is possible since the Q-function $Q_t^\star(X_t, a_t)$is quadratic with respect to a_t, and therefore can be decomposed as an expansion of basis functions $\{\Phi_n(x)\}$, with time-dependent coefficients which are parameterized by some matrix $\mathbf{W}_t$. Optimizing the choice of basis functions is a topic for further research, but cubic or quartic splines will likely provide enough complexity to model the data.

$$
\begin{aligned}
Q_t^*(X_t, a_t) &= \left(\begin{array}{c} 1, a_t, \frac{1}{2}a_t^2 \end{array} \right) \begin{pmatrix} W_{11}(t) & W_{12}(t) & \cdots & W_{1M}(t) \\ W_{21}(t) & W_{22}(t) & \cdots & W_{2M}(t) \\ W_{31}(t) & W_{32}(t) & \cdots & W_{3M}(t) \end{pmatrix} \begin{pmatrix} \Phi_1(X_t) \\ \vdots \\ \Phi_M(X_t) \end{pmatrix} \\
&\equiv \mathbf{A}_t^T \mathbf{W}_t \Phi(X_t) \equiv \mathbf{A}_t^T \mathbf{U}_W(t, X_t)
\end{aligned}
$$

which can be rearranged into an product of a parameter vector and a state-action dependent vector:

$$
\begin{aligned}
Q_t^\star(X_t, a_t) = \mathbf{A}_t^T \mathbf{W}_t \mathbf{\Phi}(X) &= \sum_{i=1}^{3} \sum_{j=1}^{M} \left(\mathbf{W}_t \odot \left(\mathbf{A}_t \otimes \mathbf{\Phi}^T(X) \right) \right)_{ij} \\
&= \overrightarrow{\mathbf{W}}_t \cdot vec\left(\mathbf{A}_t \otimes \mathbf{\Phi}^T(X) \right) \equiv \overrightarrow{\mathbf{W}}_t \vec{\Psi}(X_t, a_t)
\end{aligned}
$$

Note: $\odot$ stands for an element-wise (Hadamard) product of two matrices.

The vector of time- dependent parameters $\overline{\mathbf{W}}_t$ is obtained by concatenating columns of matrix $\mathbf{W}_t$, and similarly, $\vec{\Psi}(X_t, a_t) = vec\left(\mathbf{A}_t \otimes \mathbf{\Phi}^T(X)\right)$ stands for a vector obtained by concatenating columns of the outer product of vectors $\mathbf{A}_t$ and $\mathbf{\Phi}(X)$.

Then, we recursively calculate $\mathbf{W_t}$ backwards in time for $t = T-1, ..., 0$. As a result, we can express the

Bellman Optimality Equation (reproduced below for convenience) as a regression.

$$Q_t^{\star}(x,a) = \mathbb{E}_t\left[R_t\left(X_t,a_t,X_{t+1}\right) + \gamma \max_{a_{t+1}\in\mathcal{A}} Q_{t+1}^{\star}\left(X_{t+1},a_{t+1}\right)|X_t = x, a_t = a\right], t = 0,\ldots,T-1 \tag{47}$$

The term inside the expectation is regressed as follows:

$$R_t\left(X_t,a_t,X_{t+1}\right) + \gamma \max_{a_{t+1}\in\mathcal{A}} Q_{t+1}^{\star}\left(X_{t+1},a_{t+1}\right) = \overrightarrow{\mathbf{W}}_t\vec{\Psi}\left(X_t,a_t\right) + \varepsilon_t \tag{48}$$

with random noise ϵ_t with mean 0 at time t. Therefore,

$$Q_t^{\star}(x,a) = \mathbb{E}[\overrightarrow{\mathbf{W}}_t\vec{\Psi}\left(X_t,a_t\right) + \varepsilon_t] \tag{49}$$

Actually calculating the coefficients $\mathbf{W_t}$ are found by solving a simple **least squares** optimization problem:

$$\mathcal{L}_t\left(\mathbf{W}_t\right) = \sum_{k=1}^{N_{MC}}\left(R_t\left(X_t,a_t,X_{t+1}\right) + \gamma \max_{a_{t+1}\in\mathcal{A}} Q_{t+1}^{\star}\left(X_{t+1},a_{t+1}\right) - \overrightarrow{\mathbf{W}}_t\vec{\mathbf{\Psi}}\left(X_t,a_t\right)\right)^2 \tag{50}$$

Solving this equation gives

$$\overrightarrow{\mathbf{W}}_t^{\star} = \mathbf{S}_t^{-1}\mathbf{M}_t \tag{51}$$

where

$$\begin{aligned}
S_{nm}^{(t)} &= \sum_{k=1}^{N_{MC}} \Psi_n\left(X_t^k,a_t^k\right)\Psi_m\left(X_t^k,a_t^k\right)\\
M_n^{(t)} &= \sum_{k=1}^{N_{MC}} \Psi_n\left(X_t^k,a_t^k\right)\left(R_t\left(X_t^k,a_t^k,X_{t+1}^k\right) + \gamma \max_{a_{t+1}\in\mathcal{A}} Q_{t+1}^k\left(X_{t+1}^k,a_{t+1}\right)\right)
\end{aligned}$$

Finding $\max_{a_{t+1}\in\mathcal{A}} Q_{t+1}^{\star}\left(X_{t+1}^k,a_{t+1}\right)$ is now easy since we know $\mathbf{W}_{t+1}$ and therefore we know

$$\mathbf{U}_W(t+1,X_{t+1}) = \mathbf{W}_{t+1}\mathbf{\Phi}(X_{t_+1})$$

Therefore, we can express the updated Q-function as

$$Q_{t+1}^{*}(X_{t+1},a_{t+1}^{*}) = \mathbf{U}_W^{(0)}(t+1,X_{t+1}) + a_{t+1}^{*}\mathbf{U}_W^{(1)}(t_1,X_{t+1}) + \frac{(a_{t+1}^{*})^2}{2}\mathbf{U}_W^{(2)}(t+1,X_{t+1}) \tag{52}$$

Important note: this is a quadratic equation with respect with to a_{t+1}^{*}, but it is incorrect to use the maximum as a function of a_{t+1}^{*} to solve for the optimal value. This would mean that the algorithm uses the same dataset to estimate both the optimal action and the optimal Q-function. This leads to overestimation of

$$\max_{a_{t+1}\in\mathcal{A}} Q_{t+1}^k\left(X_{t+1}^k,a_{t+1}\right)$$

due to Jensen's inequality and the convexity of the $\max(\cdot)$ function. The correct calculation of the optimal action a_{t+1}^{*} would be to expand the optimal action in terms of basis functions, much like we did with the Q-function.

6.1.1 Estimating the optimal hedge

We express the optimal hedge as

$$a_t^*(X_t) = \sum_n^M \phi_{nt} \mathbf{\Phi_n}(X_t) \tag{53}$$

where we use the same basis functions as before. For convenience, the Q-function is reproduced below:

$$\begin{aligned} Q_t^*\left(X_t, a_t\right) &= \gamma \mathbb{E}_t\left[Q_{t+1}^*\left(X_{t+1}, a_{t+1}^*\right) + a_t \Delta S_t\right] \\ &\quad - \lambda\gamma^2 \mathbb{E}_t\left[\hat{\Pi}_{t+1}^2 - 2a_t\hat{\Pi}_{t+1}\Delta\hat{S}_t + a_t^2\left(\Delta\hat{S}_t\right)^2\right], t = 0, \ldots, T-1 \end{aligned}$$

We find ϕ_{nt} as follows:

1. Replace the expectation in the equation above by an MC estimate
2. Drop all a_t-independent terms
3. Substitute the expansion $a^*t(X_t) = \sum_n^M \phi_{nt}\mathbf{\Phi_n}(X_t)$ for a_t
4. Change the overall sign to change from maximization to minimization

This equivalently leads to minimization of the following equation:

$$G_t(\phi) = \sum_{k=1}^{N_{MC}} \left(-\sum_n \phi_{nt}\mathbf{\Phi}_n(X_t^k)\triangle S_t^k + \gamma\lambda(\hat{\Pi}_{t+1}^k - \sum_n \phi_{nt}\mathbf{\Phi}_n(X_t^k)\triangle\hat{S}_t^k)^2 \right) \tag{54}$$

Minimizing this with respect to the coefficients ϕ_{nt} leads to the following set of linear equations:

$$\sum_m^M A_{nm}^{(t)}\phi_{mt} = B_n^{(t)}, n = 1, ..., M \tag{55}$$

where

$$\mathbf{A_{nm}^{(t)}} = \sum_{k=1}^{N_{MC}} \mathbf{\Phi}_n(X_t^k)\mathbf{\Phi}_m(X_t^k)(\triangle\hat{S}_t^k)^2 \tag{56}$$

$$\mathbf{B_{nm}^{(t)}} = \sum_{k=1}^{N_{MC}} \mathbf{\Phi}_n(X_t^k)\left[\hat{\Pi}_{t+1}^k\triangle\hat{S}_t^k + \frac{1}{2\gamma\lambda}\triangle S_t^k\right] \tag{57}$$

which leads to the solution for the coefficients ϕ_t^* of the optimal action $a_t^*(X_t)$ as

$$\phi_t^* = \mathbf{A_t^{-1}B_t} \tag{58}$$

where **A** and **B** are given above. Now we are done.

Note: since an analytical optimal action is available, some problems that arise in classical Q-learning are avoided, such as the potential overestimation of the optimal action. Usually, this is addressed by using a variant of the Q-learning method, like Double Q-Learning. However, this issue is avoided in this QLBS model.

7 Dicussion

Essentially, this model allows us to price and hedge a stock option by learning **directly from past trading data**. It implements the principle of hedging first and pricing second in a consistent way for a discrete-time version of the Black Scholes model.

If we have an options pricing model where the option price is the negative of a Q-function, and the option hedge is its second argument, the optimal action value Q-function hedges and prices the option **by definition.** The only modelling assumption is that local rewards (or negative losses) are quadratic in terms of hedging actions - this leads to a model/distribution free solution. The only thing that the Q-learning algorithm needs is a dataset in the form of cross-sectional tuples

$$\mathcal{F}_t^{(n)} = \left\{ \left(X_t^{(n)}, a_t^{(n)}, R_t^{(n)}, X_{t+1}^{(n)} \right) \right\}_{t=0}^{T-1}, n = 1, \ldots, N_{MC} \tag{59}$$

across all simulations. It eliminates the need for any volatility surface models or jump-diffusion models. As long as rewards are quadratic with respect to the action taken, the algorithm produces model-independent results that define the optimal price and the optimal hedge.

Since the Q-Learner is an *off policy* algorithm, it will still learn the optimal price/hedge from training data given enough data *even if the actions taken are completely random.* As a result, it is possible to create synthetic trading histories by combining real-market data (stock prices) with a pre-defined trading strategy in stocks.

From a financial point of view, the Black Scholes model washes out any risk out of rehedging a portfolio in lognormal dynamics when the rehedging time step $\triangle t$ is taken to the limit $\triangle t \to 0$, and theoretically eliminates the need for an options trading market, a conceptual result that makes no intuitive sense but complete mathematical sense. For each of these options, there will be a different Bellman Optimality equation due to the mechanics of the option itself which will have to be solved analytically in order to guarantee quadratic rewards. The American option, for instance, can choose to either rehedge or exit the position entirely ahead of the expiration date, an action that makes it difficult to price. The interesting aspect about this QLBS model is that it could theoretically rediscover the Black-Scholes formula for the option price by using the QLBS by analyzing data collected with random actions (since it is an off policy method) and using a Neural Network to approximate the value function. However, the question remains as to whether this method is numerically stable in real world applications.

7.1 Optimizing risk preferences

The parameter λ is part of the *risk premium*, the term that determines how risk-averse the seller of the option is. Usually, this is a small positive number. We propose historical stock analysis from a large number of stocks across multiple industries for the past 5 years in order to determine what historical average λ would be in the market today. This could, in a sense, be called the "fair" or "risk-neutral" (not mathematically speaking) value to be compared against when pricing options using the QLBS model.

8 Implementation and Trial Run

We perform an initial sanity check of our data and take a look at all the different components of this algorithm to check if they are 1) stable, and 2) accurate. In order to do this, we calculate a basic European Put using the classical BS model, and the QLBS model.
First, we set all our parameters, such as

1. Initial stock price, $S_0 = 100$
2. Drift, $\mu = 0.05$
3. Volatility, $\sigma = 0.15$
4. Risk-free rate, $r = 0.03$
5. Maturity, $M = 1$
6. Number of MC paths, $N_{MC} = 10000$
7. Number of time steps, $T = 10$
8. Time interval, $\Delta t = \frac{M}{T}$
9. Discount factor, $\gamma = e^{-r \cdot \Delta t}$
10. Risk aversion, $\lambda = 0.001$

We also define our payoff function that computes the terminal payoff of a European put option.

$$H_T\left(S_T\right) = \max\left(K - S_T, 0\right)$$

8.1 Black-Scholes Simulation

In this step, we simulate N_{MC} stock price sample paths with T steps by the classical Black-Scholes formula and Monte Carlo simulation.

$$dS_t = \mu S_t dt + \sigma S_t dW_t \qquad S_{t+1} = S_t e^{\left(\mu - \frac{1}{2}\sigma^2\right)\Delta t + \sigma\sqrt{\Delta t} Z}$$

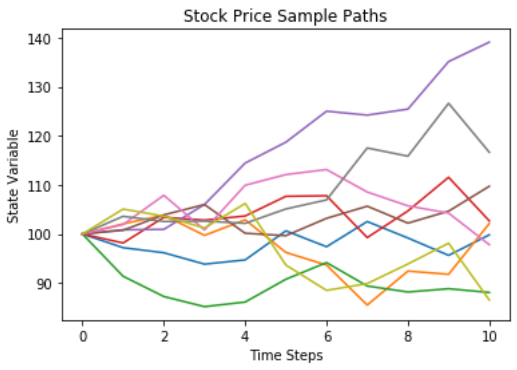
where Z is a standard normal random variable.
We generate a synthetic dataset with a seed value of 42. Based on the simulated stock price S_t paths, we compute state variable X_t by the following relation.

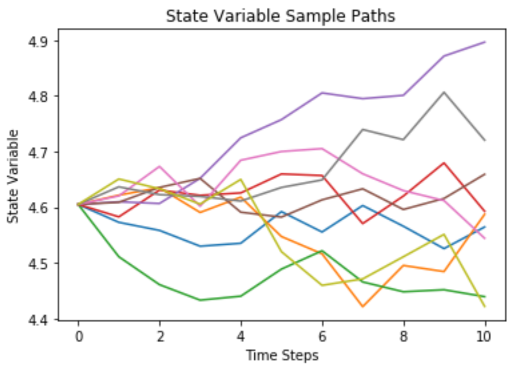
$$X_t = -\left(\mu - \frac{1}{2}\sigma^2\right) t\Delta t + \log S_t$$

We also compute

$$\Delta S_t = S_{t+1} - e^{r\Delta t} S_t \qquad \Delta\hat{S}_t = \Delta S_t - \Delta\bar{S}_t \qquad t = 0, ..., T-1$$

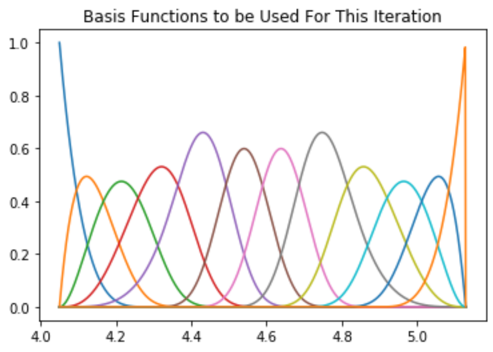
where $\Delta\bar{S}_t$ is the sample mean of all values of ΔS_t.
Plots of 10 stock price S_t and state variable X_t paths are shown below.





8.2 Define Spline Basis Functions

We use cubic splines and select the appropriate knots to use. The basis functions to be used are shown below.



Finally, we create data matrices with feature values. The "features" here are the values of basis functions at data points. The outputs are 3D arrays of dimensions (Number of time steps) x (Number of MC Simulations) x (Number of Basis Functions).

8.3 DP Solution For QLBS

The MDP problem in this case is to solve the following Bellman optimality equation for the action-value function.

$$Q_t^\star(x,a) = \mathbb{E}_t\left[R_t(X_t,a_t,X_{t+1}) + \gamma \max_{a_{t+1}\in\mathcal{A}} Q_{t+1}^\star(X_{t+1},a_{t+1})\,|X_t = x, a_t = a\right], t=0,...,T-1, \quad \gamma = e^{-r\Delta t}$$

where $R_t(X_t,a_t,X_{t+1})$ is the one-step time-dependent random reward and $a_t(X_t)$ is the action (hedge).

With this set of basis functions (to approximate this equation) $\left\{\Phi_n\left(X_t^k\right)\right\}_{n=1}^N$, expand the optimal action (hedge) $a_t^\star(X_t)$ and optimal Q-function $Q_t^\star(X_t,a_t^\star)$ in basis functions with time-dependent coefficients.

$$a_t^\star(X_t) = \sum_n^N \phi_{nt}\Phi_n(X_t) \qquad Q_t^\star(X_t,a_t^\star) = \sum_n^N \omega_{nt}\Phi_n(X_t)$$

Coefficients ϕ_{nt} and ω_{nt} are computed recursively backward in time for $t = T1, ..., 0$. Coefficients for expansions of the optimal action $a_t^\star(X_t)$ are solved by

$$\phi_t = \mathbf{A}_t^{-1}\mathbf{B}_t$$

where $\mathbf{A}_t$ and $\mathbf{B}_t$ are matrix and vector respectively with elements given by

$$A_{nm}^{(t)} = \sum_{k=1}^{N_{MC}} \Phi_n\left(X_t^k\right)\Phi_m\left(X_t^k\right)\left(\Delta\hat{S}_t^k\right)^2 \qquad B_n^{(t)} = \sum_{k=1}^{N_{MC}} \Phi_n\left(X_t^k\right)\left[\hat{\Pi}_{t+1}^k\Delta\hat{S}_t^k + \frac{1}{2\gamma\lambda}\Delta S_t^k\right]$$

8.4 Compute optimal hedge and portfolio value

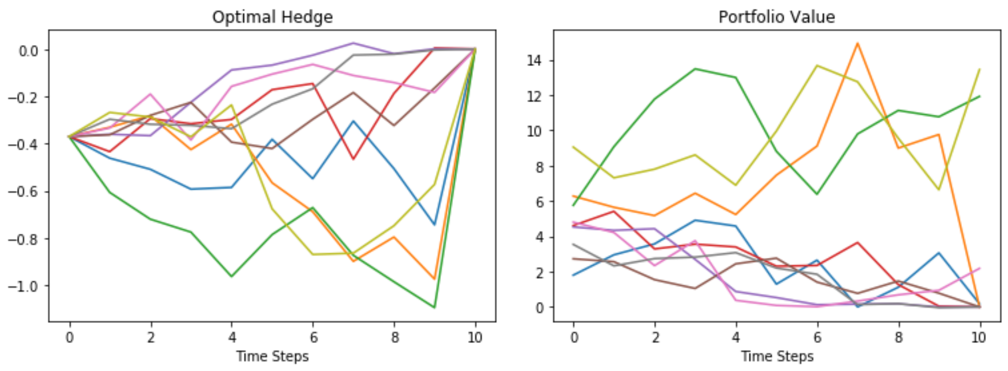
Call function A and function B for $t = T-1, ..., 0$ together with basis function $\Phi_n(X_t)$ to compute optimal action $a_t^\star(X_t) = \sum_n^N \phi_{nt}\Phi_n(X_t)$ backward recursively with terminal condition $a_T^\star(X_T) = 0$.

Once the optimal hedge $a_t^\star(X_t)$ is computed, the portfolio value Π_t could also be computed backward recursively by

$$\Pi_t = \gamma\left[\Pi_{t+1} - a_t^\star\Delta S_t\right] \quad t = T-1, ..., 0$$

together with the terminal condition $\Pi_T = H_T(S_T) = \max(K - S_T, 0)$ for a European put option.

Also compute $\hat{\Pi}_t = \Pi_t - \bar{\Pi}_t$, where $\bar{\Pi}_t$ is the sample mean of all values of Π_t. We write function A and function B to compute the value of matrix $\mathbf{A}_t$ and vector $\mathbf{B}_t$. Plots of 10 optimal hedge $a_t^\star$ and portfolio value Π_t paths are shown below.



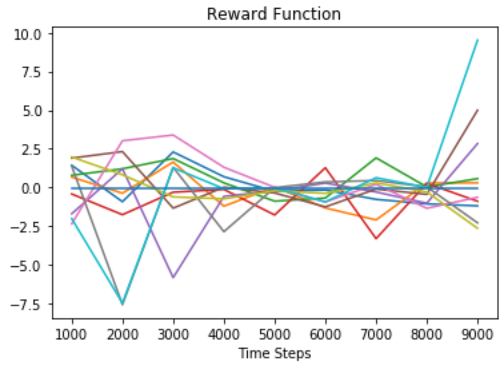
We perform a quick sanity check to make sure nothing in this model is contradictory to what we already know. Looking at the portfolio values (which are then the result of inputting the optimal hedge), we see that some of the portfolio values trend to 0. Looking at the corresponding original stock simulation of the portfolio - indeed, those stocks all have a terminal value of 100 or less at the end of the time period.

Once the optimal hedge $a_t^\star$ and portfolio value Π_t are all computed, the reward function $R_t(X_t, a_t, X_{t+1})$ could then be computed by

$$R_t(X_t, a_t, X_{t+1}) = \gamma a_t\Delta S_t - \lambda Var\left[\Pi_t|\mathcal{F}_t\right] \quad t = 0, ..., T-1$$

with terminal condition $R_T = -\lambda Var\left[\Pi_T\right]$.

A plot of 10 reward function R_t paths is shown below.



8.5 Compute the optimal Q-function with the DP approach

Coefficients for expansions of the optimal Q-function $Q_t^\star(X_t, a_t^\star)$ are solved by

$$\omega_t = \mathbf{C}_t^{-1}\mathbf{D}_t$$

where $\mathbf{C}_t$ and $\mathbf{D}_t$ are matrix and vector respectively with elements given by

$$C_{nm}^{(t)} = \sum_{k=1}^{N_{MC}} \Phi_n\left(X_t^k\right)\Phi_m\left(X_t^k\right) \qquad D_n^{(t)} = \sum_{k=1}^{N_{MC}} \Phi_n\left(X_t^k\right)\left(R_t\left(X_t, a_t^\star, X_{t+1}\right) + \gamma \max_{a_{t+1}\in\mathcal{A}} Q_{t+1}^\star\left(X_{t+1}, a_{t+1}\right)\right)$$

Call function C and function D for $t = T-1, ..., 0$ together with basis function $\Phi_n(X_t)$ to compute optimal action Q-function $Q_t^\star(X_t, a_t^\star) = \sum_n^N \omega_{nt}\Phi_n(X_t)$ backward recursively with terminal condition $Q_T^\star(X_T, a_T = 0) = -\Pi_T(X_T) - \lambda Var\left[\Pi_T(X_T)\right]$. This is computed in the next step.

8.6 Hedging and Pricing with Reinforcement Learning

The following is a batch-mode off-policy model-free Q-Learning by Fitted Q-Iteration. The only data available is given by a set of N_{MC} paths for the underlying state variable X_t, hedge position a_t, instantaneous reward R_t and the next-time value X_{t+1}.

$$\mathcal{F}_t^k = \left\{\left(X_t^k, a_t^k, R_t^k, X_{t+1}^k\right)\right\}_{t=0}^{T-1} \qquad k = 1, ..., N_{MC}$$

Expand the Q-function through basis functions with time-dependent coefficients parametrized by a matrix $\mathbf{W}_t$.

$$Q_t^\star(X_t, a_t) = \mathbf{A}_t^T\mathbf{W}_t\Phi(X_t) = \mathbf{A}_t^T\mathbf{U}_W(t, X_t) = \vec{W}_t^T\vec{\Psi}(X_t, a_t)$$

$$\mathbf{A}_t = \begin{pmatrix} 1 \\ a_t \\ \frac{1}{2}a_t^2 \end{pmatrix} \qquad \mathbf{U}_W(t, X_t) = \mathbf{W}_t\Phi(X_t)$$

where $\vec{W}_t$ is obtained by concatenating columns of matrix $\mathbf{W}_t$ while $(\mathbf{\Psi}(X_t, a_t)) = \left(\mathbf{A}_t \otimes \mathbf{\Phi}^T(X)\right)$ stands for a vector obtained by concatenating columns of the outer product of vectors $\mathbf{A}_t$ and $\mathbf{\Phi}(X)$.

We compute vector $\mathbf{A}_t$ then compute $\vec{\Psi}(X_t, a_t)$ for each X_t^k and store in a dictionary with key path and time $[k, t]$.

8.7 Generate off-policy data

- **on-policy** data - contains an optimal action and the corresponding reward
- **off-policy** data - contains a random action and the corresponding reward

Given a large enough sample, i.e. N_{MC} tending to infinity, Q-Learner will learn an optimal policy from the data in a model-free setting. In our case a random action is an optimal action + noise generated by sampling from a uniform distribution

$$a_t(X_t) = a_t^\star(X_t)\,\Delta U[1-\eta, 1+\eta]$$

where η is a disturbance level In other words, each noisy action is calculated by taking optimal action computed previously and multiplying it by a uniform r.v. in the interval $[1-\eta, 1+\eta]$

Therefore, we will use synthetic trading data to better estimate the accuracy of this algorithm.

In the loop below, we do the following:

- Compute the optimal policy, and write the result to a_t^*
- Now disturb these values by a random noise

$$a_t(X_t) = a_t^\star(X_t)\,\Delta U[1-\eta, 1+\eta]$$

- Compute portfolio values corresponding to observed actions

$$\Pi_t = \gamma\left[\Pi_{t+1} - a_t^\star \Delta S_t\right] \quad t = T-1, ..., 0$$

- Compute rewards corresponding to observed actions

$$R_t(X_t, a_t, X_{t+1}) = \gamma a_t \Delta S_t - \lambda Var\left[\Pi_t | \mathcal{F}_t\right] \quad t = T-1, ..., 0$$

with terminal condition

$$R_T = -\lambda Var\left[\Pi_T\right]$$

8.8 Calculate $\mathbf{S}_t$ and $\mathbf{M}_t$

Vector $\vec{W}_t$ can be solved by

$$\vec{W}_t = \mathbf{S}_t^{-1}\mathbf{M}_t$$

where $\mathbf{S}_t$ and $\mathbf{M}_t$ are a matrix and a vector respectively, with elements given by

$$S_{nm}^{(t)} = \sum_{k=1}^{N_{MC}} \Psi_n\left(X_t^k, a_t^k\right) \Psi_m\left(X_t^k, a_t^k\right) \qquad M_n^{(t)} = \sum_{k=1}^{N_{MC}} \Psi_n\left(X_t^k, a_t^k\right) \left(R_t\left(X_t, a_t, X_{t+1}\right) + \gamma \max_{a_{t+1} \in \mathcal{A}} Q_{t+1}^{\star}\left(X_{t+1}, a_{t+1}\right)\right)$$

Define function S and function M to compute the value of matrix $\mathbf{S_t}$ and vector $\mathbf{M_t}$.

8.9 Computing the optimal action and the optimal action-value pair for the Q-function

Call function S and function M for $t = T-1, ..., 0$ together with vector $\vec{\Psi}\left(X_t, a_t\right)$ to compute $\vec{W}_t$ and learn the Q-function

$$Q_t^{\star}\left(X_t, a_t\right) = \mathbf{A}_t^T \mathbf{U}_W\left(t, X_t\right)$$

implied by the input data backward recursively with terminal condition

$$Q_T^{\star}\left(X_T, a_T = 0\right) = -\Pi_T\left(X_T\right) - \lambda Var\left[\Pi_T\left(X_T\right)\right]$$

.

When the vector $\vec{W}_t$ is computed as per the above at time t, we can convert it back to a matrix $\mathbf{W_t}$ obtained from the vector $\vec{W}_t$ by reshaping to the shape $3 \times M$.

We can now calculate the matrix $\mathbf{U}_t$ at time t for the whole set of MC paths as follows:

$$\mathbf{U}_W\left(t, X_t\right) = \begin{bmatrix} \mathbf{U}_W^{0,k}\left(t, X_t\right) \\ \mathbf{U}_W^{1,k}\left(t, X_t\right) \\ \mathbf{U}_W^{2,k}\left(t, X_t\right) \end{bmatrix} = \mathbf{W_t} \mathbf{\Phi_t}\left(\mathbf{t}, \mathbf{X_t}\right)$$

Here the matrix $\mathbf{\Phi}_t$ has the shape shape $M \times N_{MC}$. Therefore, their dot product has dimension $3 \times N_{MC}$, as it should be.

Once this matrix $\mathbf{U}_t$ is computed, individual vectors $\mathbf{U}_W^1, \mathbf{U}_W^2, \mathbf{U}_W^3$ for all MC paths are read off as rows of this matrix.

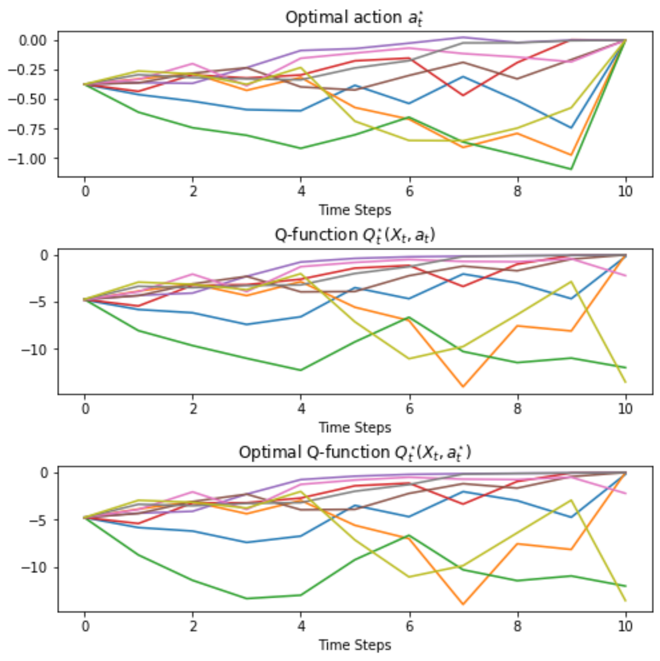
From here, we can compute the optimal action and optimal Q-function $Q^{\star}(X_t, a_t^{\star})$ at the optimal action for a given step t. This will be used to evaluate the $\max_{a_{t+1} \in \mathcal{A}} Q^{\star}\left(X_{t+1}, a_{t+1}\right)$.

The optimal action and optimal Q-function with the optimal action are computed as such:

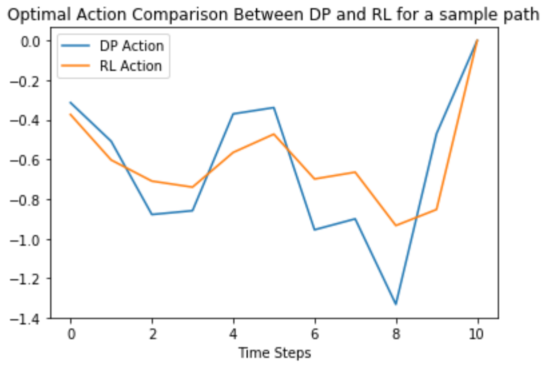
$$a_t^{\star}\left(X_t\right) = \frac{\mathbb{E}_t\left[\Delta \hat{S}_t \hat{\Pi}_{t+1} + \frac{1}{2\gamma\lambda} \Delta S_t\right]}{\mathbb{E}_t\left[\left(\Delta \hat{S}_t\right)^2\right]}, \qquad Q_t^{\star}\left(X_t, a_t^{\star}\right) = \mathbf{U}_W^{(0)}\left(t, X_t\right) + a_t^{\star} \mathbf{U}_W^{(2)}\left(t, X_t\right) + \frac{1}{2}\left(a_t^{\star}\right)^2 \mathbf{U}_W^{(2)}\left(t, X_t\right)$$

with terminal condition $a_T^{\star} = 0$ and $Q_T^{\star}\left(X_T, a_T^{\star} = 0\right) = -\Pi_T\left(X_T\right) - \lambda Var\left[\Pi_T\left(X_T\right)\right]$.

Plots of 10 optimal action $a_t^{\star}\left(X_t\right)$, optimal Q-function with optimal action $Q_t^{\star}\left(X_t, a_t^{\star}\right)$ and implied Q-function $Q_t^{\star}\left(X_t, a_t\right)$ paths are shown below.



Comparing the two sets of optimal actions for DP and QLBS, we see that the noise-disturbed optimal actions are more robust and less prone to variability and sudden price shocks. This could be an area of further research as to determine exactly why the Q-Function is more "smooth" than the DP equivalent.



8.10 Summary

The summary for this trial simulation is as follows:

```
---------------------------------
       QLBS RL Option Pricing
---------------------------------


Initial Stock Price:        100
Drift of Stock:             0.05
Volatility of Stock:        0.15
```

```
Risk-free Rate:            0.03
Risk aversion parameter : 0.001
Strike:                    100
Maturity:                  1
Random seed:               42
The QLBS Put Price 1 :     4.7655
Black-Scholes Put Price:  4.5296
Computational time for Q-Learning: 1.146627902984619l4 seconds
```

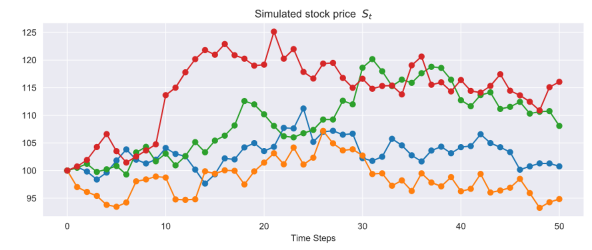
9 Numerical Experimentation

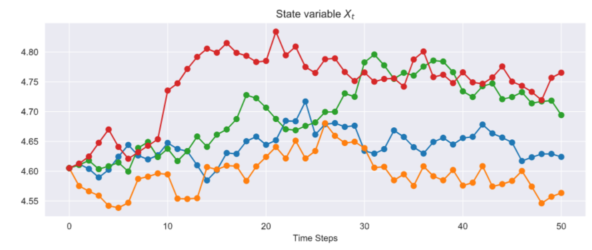
In this section, we will explore different numerical experiments done on the various parameters of these simulations, along with error graphs. Experiments were done twice, with a put option and a call option priced on the same parameters. All graphs shown represent the (signed) percentage error.

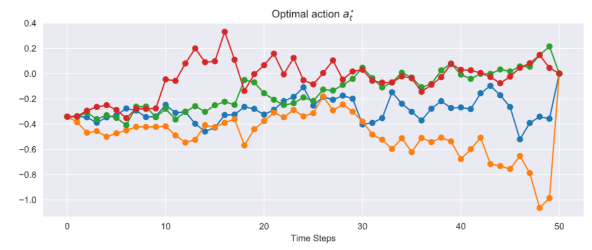
9.1 Vanilla Simulation

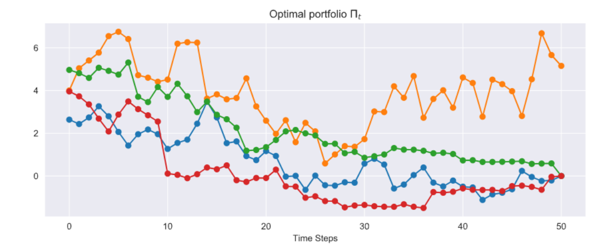
A vanilla put simulation was done for the purposes of comparison. The parameters used are as follows:

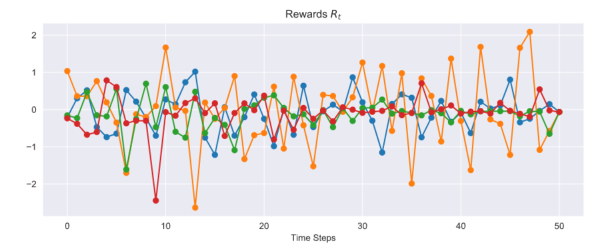
Initial Stock Price	100
Drift of Stock	0
Volatility of Stock	0.15
Risk-free Interest Rate	0.03
Risk Aversion Parameter	0.001
Strike Price	100
Time to Maturity (Years)	1
Random Seed	42
Number of Monte Carlo Paths	10000
Number of time steps	10

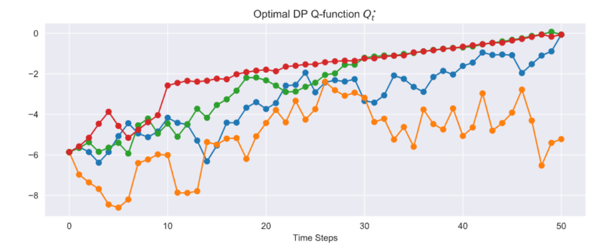








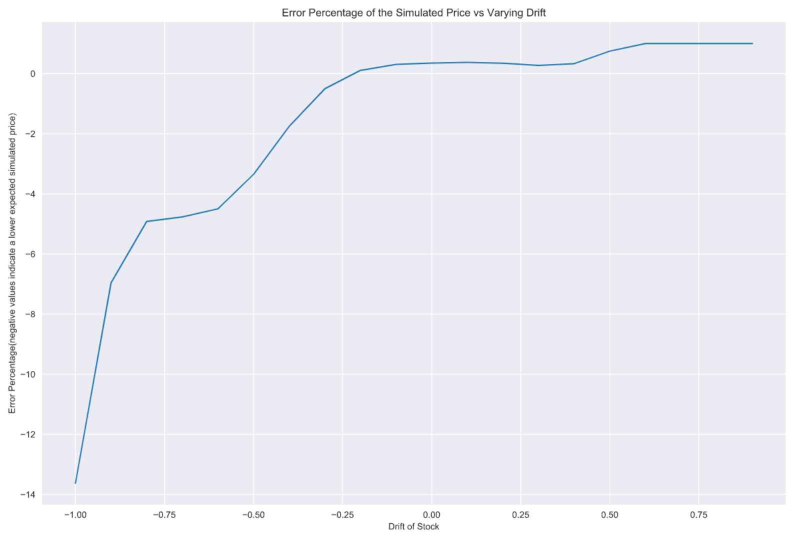




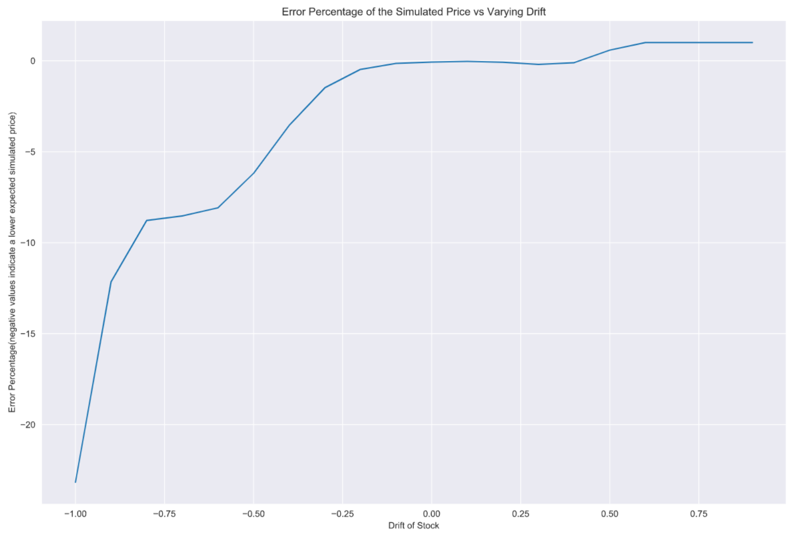
Here, we can see the pronounced effects of having more time steps and no drift. We can also see that the rewards for the different simulations have generally followed white noise patterns.

9.2 Varying Drift

Initial Stock Price	100
Drift of Stock	$-1 \to 1$
Volatility of Stock	0.15
Risk-free Interest Rate	0.03
Risk Aversion Parameter	0.001
Strike Price	100
Time to Maturity (Years)	1
Random Seed	42
Number of Monte Carlo Paths	10000
Number of time steps	10



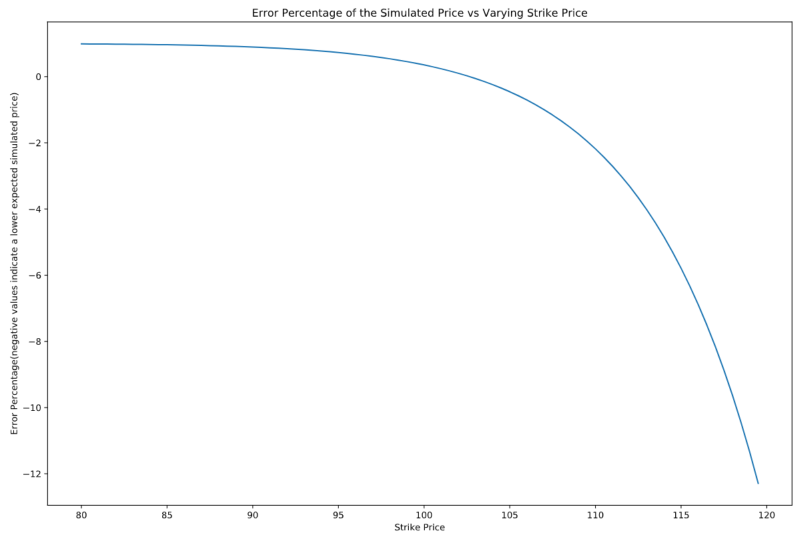
Call



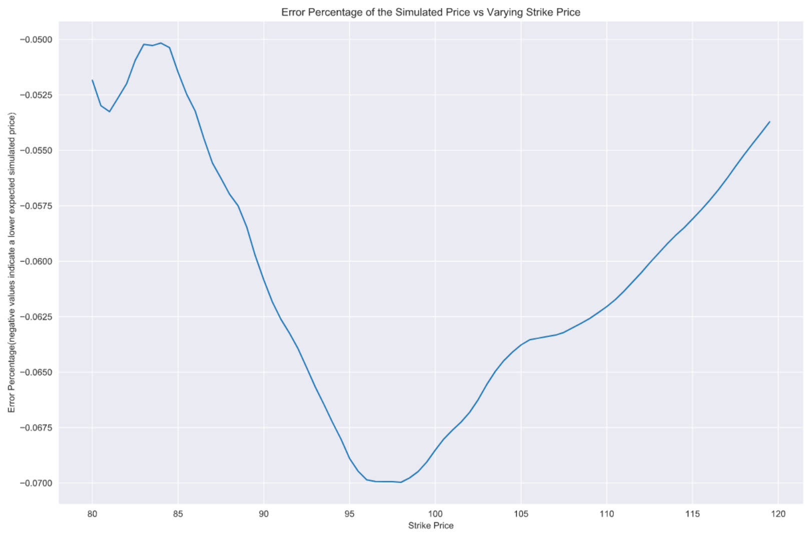
Put

9.3 Varying Strike Price

Initial Stock Price	$80 \rightarrow 120$
Drift of Stock	0
Volatility of Stock	0.15
Risk-free Interest Rate	0.03
Risk Aversion Parameter	0.001
Strike Price	100
Time to Maturity (Years)	1
Random Seed	42
Number of Monte Carlo Paths	10000
Number of time steps	10



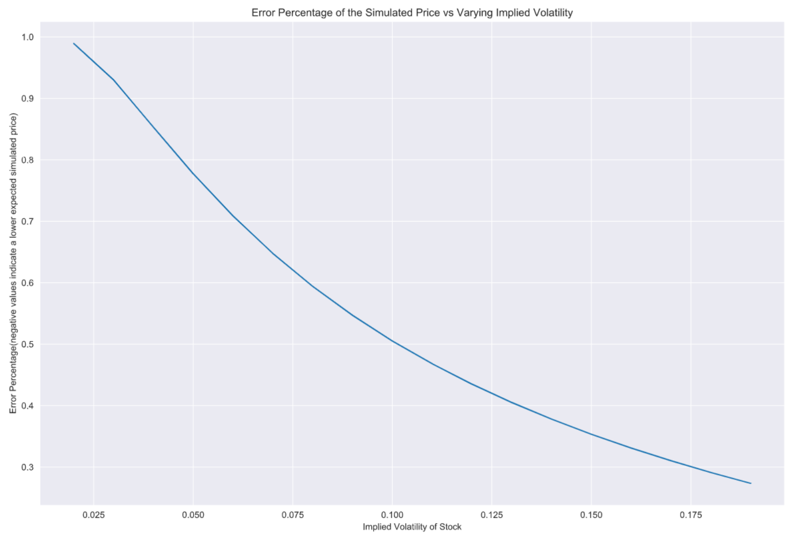
Call



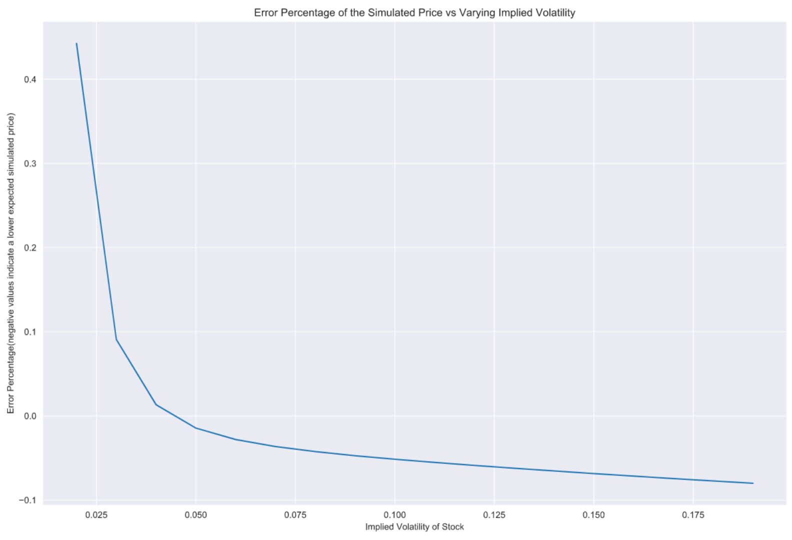
Put

9.4 Varying Volatility

Initial Stock Price	100
Drift of Stock	0
Volatility of Stock	$0.02 \to 0.2$
Risk-free Interest Rate	0.03
Risk Aversion Parameter	0.001
Strike Price	100
Time to Maturity (Years)	1
Random Seed	42
Number of Monte Carlo Paths	10000
Number of time steps	10



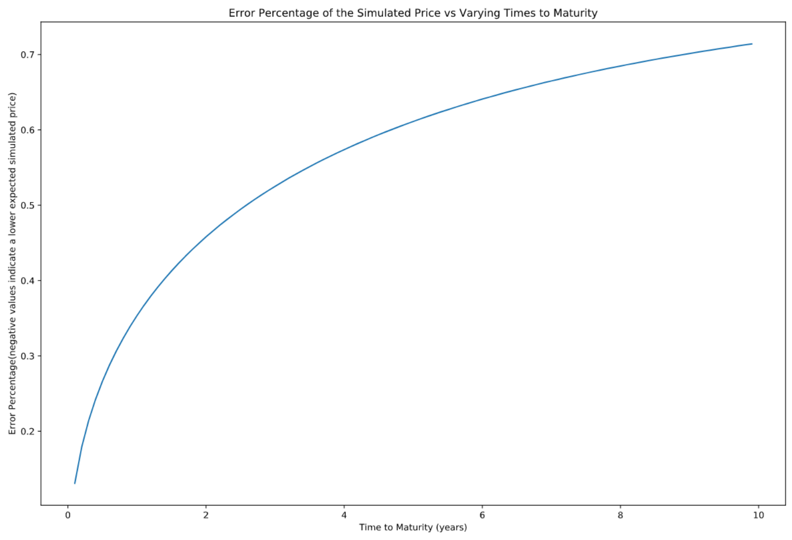
Call



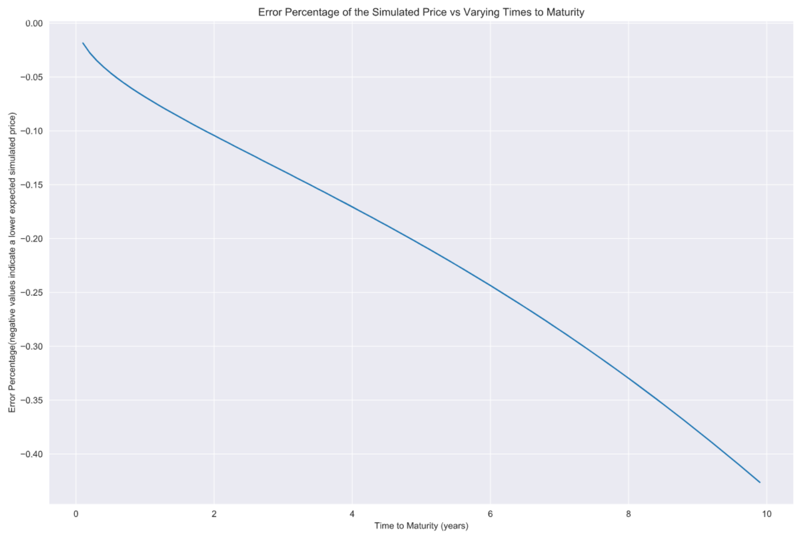
Put

9.5 Varying Maturity

Initial Stock Price	100
Drift of Stock	0
Volatility of Stock	0.15
Risk-free Interest Rate	0.03
Risk Aversion Parameter	0.001
Strike Price	100
Time to Maturity (Years)	$0 \to 10$
Random Seed	42
Number of Monte Carlo Paths	10000
Number of time steps	10



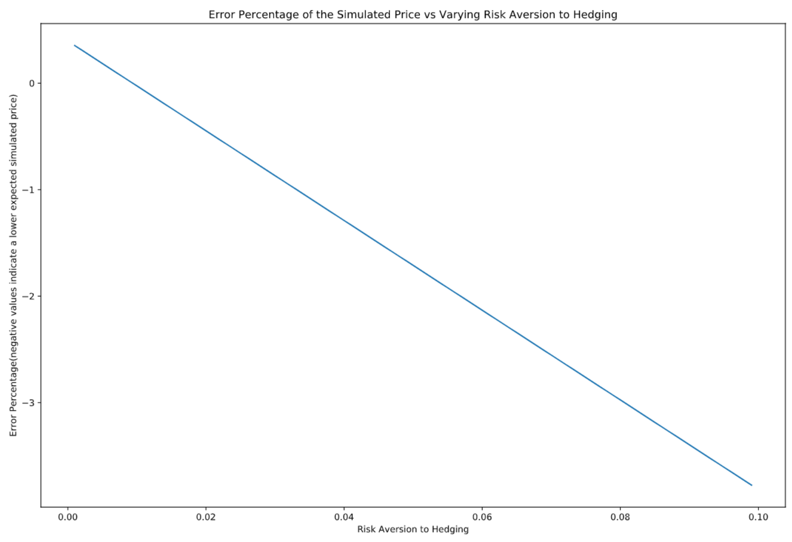
Call



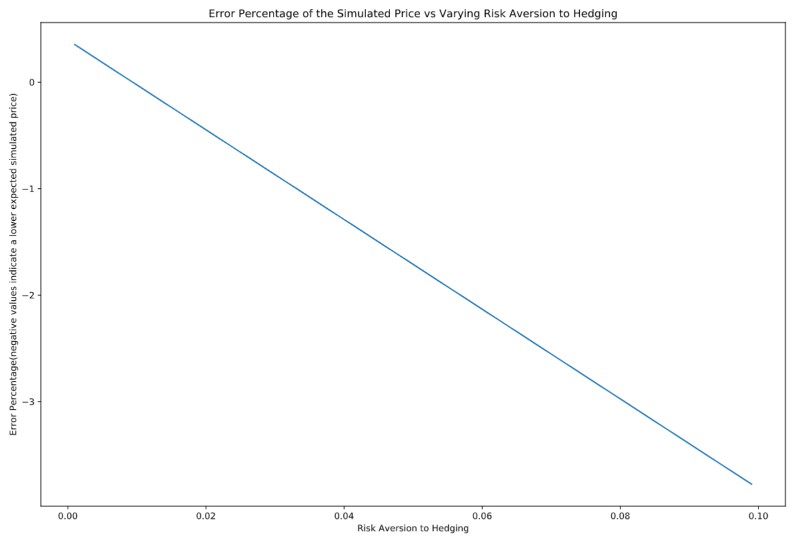
Put

9.6 Varying Risk Aversion

Initial Stock Price	100
Drift of Stock	0
Volatility of Stock	0.15
Risk-free Interest Rate	0.03
Risk Aversion Parameter	$0 \to 0.1$
Strike Price	100
Time to Maturity (Years)	1
Random Seed	42
Number of Monte Carlo Paths	10000
Number of time steps	10



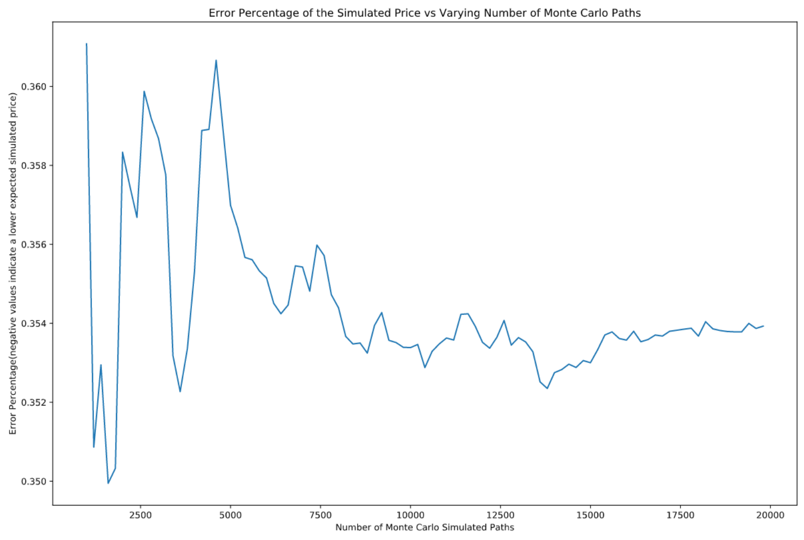
Call



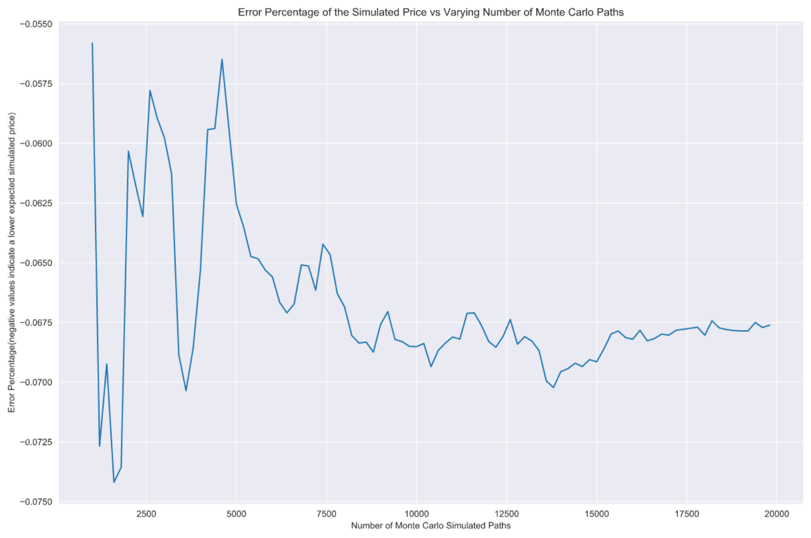
Put

9.7 Varying Monte Carlo Paths

Initial Stock Price	100
Drift of Stock	0
Volatility of Stock	0.15
Risk-free Interest Rate	0.03
Risk Aversion Parameter	0.001
Strike Price	100
Time to Maturity (Years)	1
Random Seed	42
Number of Monte Carlo Paths	$2000 \to 20000$
Number of time steps	10



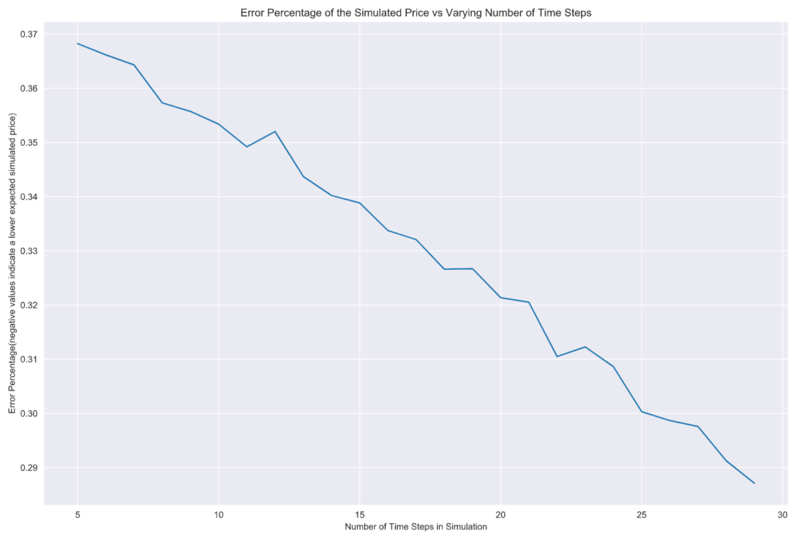
Call



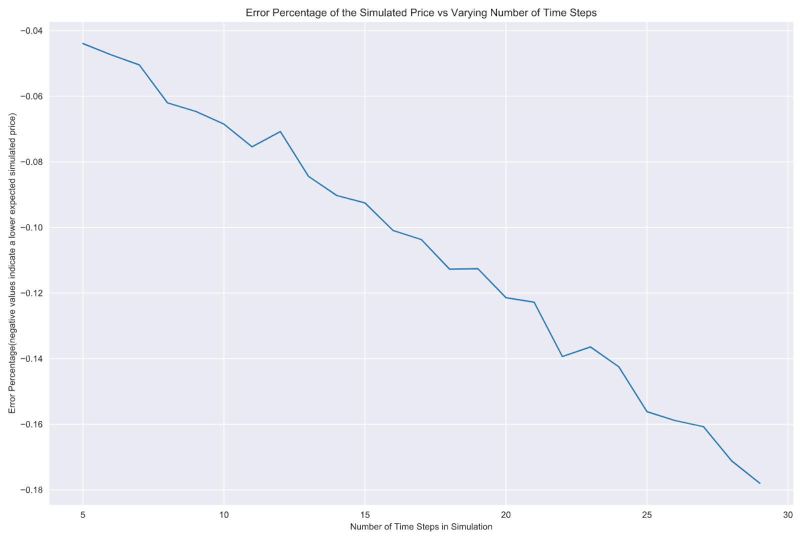
Put

9.8 Varying Time Steps

Initial Stock Price	100
Drift of Stock	0
Volatility of Stock	0.15
Risk-free Interest Rate	0.03
Risk Aversion Parameter	0.001
Strike Price	100
Time to Maturity (Years)	1
Random Seed	42
Number of Monte Carlo Paths	10000
Number of time steps	$10 \to 30$



Call

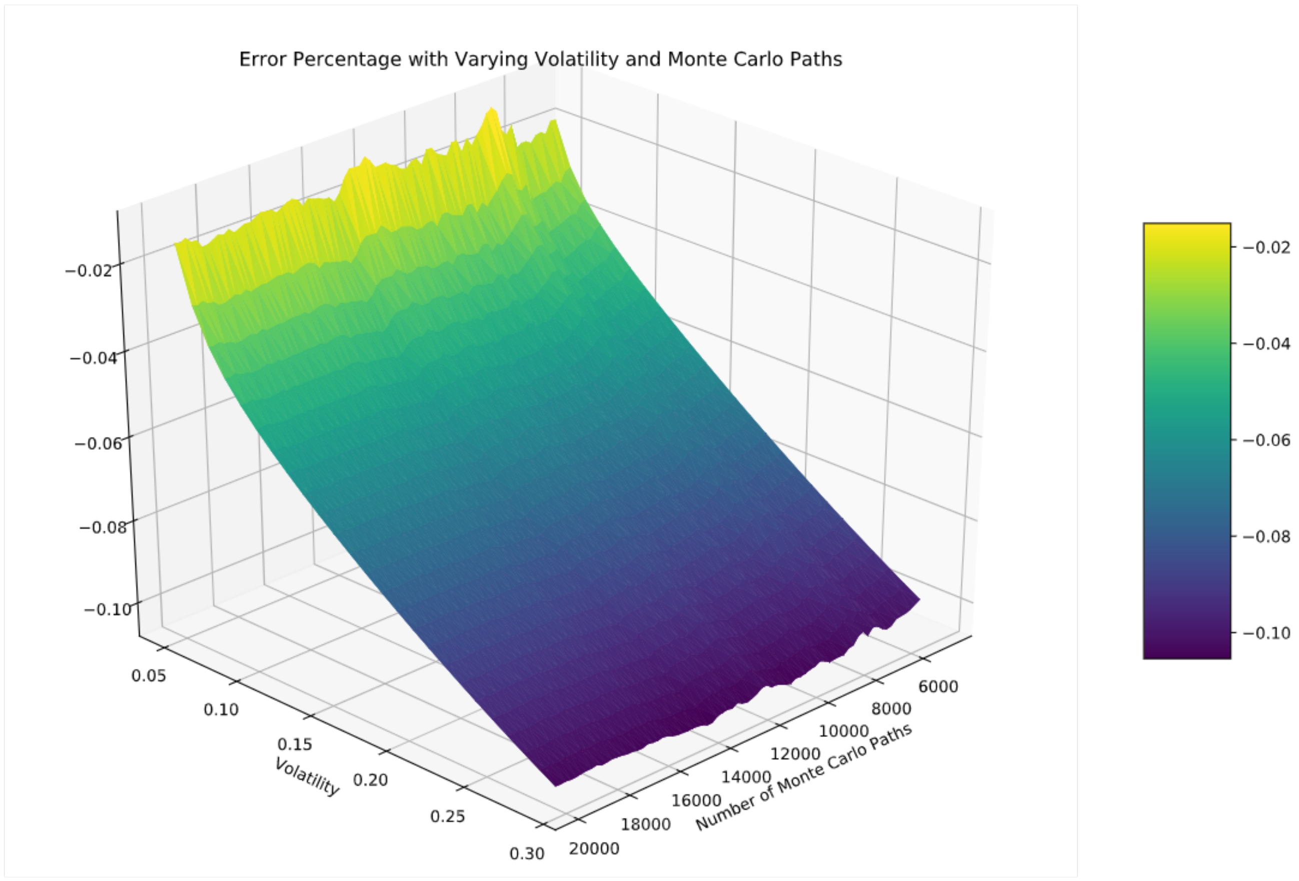


Put

9.9 Varying Volatility and Number of Monte Carlo Paths

Initial Stock Price	100
Drift of Stock	0
Volatility of Stock	$0.02 \rightarrow 0.2$
Risk-free Interest Rate	0.03
Risk Aversion Parameter	0.001
Strike Price	100
Time to Maturity (Years)	1
Random Seed	42
Number of Monte Carlo Paths	$2000 \rightarrow 20000$
Number of time steps	10

Put



10 Discussion

It seems that there is a small but consistent error when using the QLBS model to price stocks. The biggest source of error would be from the mishedging error, so increasing the number of Monte Carlo paths and the number of time steps should theoretically decrease the error. However, the numerical results given do not perfectly correspond with such a hypothesis.

The number of Monte Carlo paths instead seems to stabilize the error rather than decreasing it mostly monotonically, as would be expected. Both puts and calls seem to result in remarkably similar error graphs as the number of MC paths increases. The number of time steps decreases the error for call options, but increases the absolute error for put options. the interesting thing of note here is of the remarkably similar error graphs, even though the options priced have differently signed payoff functions. Quite possibly the most interesting result comes with the variance of time to maturity. The error graphs for puts and calls are also different, even though theoretically the graphs should looks similar. Variance of variance seems to result in asymptotically decreasing functions, although those asymptotes are not plateauing around 0. Further analysis as to these anomalous results and seeming difference in pricing strategy for puts vs calls should be done as to determine the reason behind different error graphs, even with a symmetric risk measure.

10.1 Further Possible Research Pointers

The following could serve as possible future research questions as an extension to this paper.

- Deriving the Bellman optimality equation for American, Barrier, and Asian options
- Determining if the discrepancy between the prices is significant
- The relationship between the optimal actions taken by the DP algorithm and the QLBS algorithm and how they affect the Q-Function
- Volatility of the QLBS learner when exposed to price shocks
- Real life market calibration with trading data
- Comparison of different VI reinforcement learning methods
- Usage and mathematical rigor of using QLBS as a financial pricing model
- Comparison of RL and DP methods in minimizing symmetric risk measures
- Exploration of other types of symmetric risk measures

11 Conclusions

This paper presents and implements the QLBS model. Born from the need to find an unbiased method of pricing financial derivatives that would hedge first and price second, we find that an optimal action-value Q-function does this process of *hedging and pricing by learning.* This can be solved by setting the option price as the negative of this Q-function, and the option hedge is its second argument. This MDP then can be solved by Fitted Q-Learning, which learns optimal hedges and prices directly from trading data for a replicating portfolio of a stock and a bank cash account. The fact that hedge frequencies are finite in this case ($\triangle t \to 0$ mean that our model gains independence from trying to "guess" what distribution the data fits. Overall, the QLBS model does a remarkably good job at accurately pricing options from stock data.

From the work presented in this paper, we have demonstrated the feasibility of a toy "BSM" simulated financial environment in a discrete time and space. Q Learning has been implemented as the RL algorithm of study in this environment, but the QLBS model can also admit any kind of RL model, as the generation of the data and the analyzing of the data are two disjoint and distinct processes. Due to the high scalability of this implementation - which includes nothing more than Monte Carlo Simulation and linear algebra - it is possible to benchmark different types of reinforcement learning algorithms, such as policy gradient methods, actor-critic algorithms, DQRL, and so on. However, the only requirement is that the optimal policies and action-value functions be easily computable by means of DP.

The more interesting question to raise, however, is if this QLBS model can actually constitute a legitimate financial model, like the Black-Scholes model does. There are several facts in its favor, such as the realization that Q-Learning gives both the price and hedge in one formula, if the option price is defined properly. The QLBS model is in fact simpler than the original Black-Scholes model, which depends on complicated functions like the cumulative normal distributions of composite arguments, which expand to an infinite series of elementary functions. Although we do not know whether stocks themselves inherently correspond to Brownian motion or any type of partial differential equation, what we do know is that our Q-Learning model is able to learn the trends of the data fed into it, *regardless of the nature of the distribution of the data.*

The main problem with the classical BSM model is that the requirement of continuous rehedging, i.e. $\Delta t \to 0$. Although this continuous rehedging makes perfect sense in a mathematical sense, it does not make any financial sense. The reason why markets exist to trade options is the result of not being able to run continuous hedging. This exact mishedging risk that is a result of a faulty BSM model is in fact a reality in financial markets, and therefore should be accounted for in any financial model.

As a result, the QLBS model provides an easily extendable financial pricing model to virtually any type of financial derivative. Due to reliance on data and not a predetermined distribution, specfic problems like the volatility smile problem do not exist for the QLBS - the pattern in the data that indicates the smile problem is in fact part of the process, rather than an anomaly in otherwise perfectly distributed data. Since we know that this QLBS learner is *model-free*, it will be able to handle any type of trading data that we throw at it - in other words, we are leaving the risk-free $\Delta t \to 0$ BSM world for the *risky, rational* discrete time/space world.

References

[1] Fischer Black and Myron Scholes. "The Pricing of Options and Corporate Liabilities". In: *Journal of Political Economy* 81.3 (1973), pp. 637–654. ISSN: 00223808, 1537534X. URL: http://www.jstor.org/stable/1831029.

[2] Jean-Philippe Bouchard and Marc Potters. *Theory of financial risk and derivative pricing: From statistical physics to risk management*. Cambridge Univ. Press, 2003.

[3] Michael J Brennan. "The pricing of contingent claims in discrete time models". In: *The journal of finance* 34.1 (1979), pp. 53–68.

[4] Greg Brockman et al. "Openai gym". In: *arXiv preprint arXiv:1606.01540* (2016).

[5] Aleš Černý, Jan Kallsen, et al. "On the structure of general mean-variance hedging strategies". In: *The Annals of probability* 35.4 (2007), pp. 1479–1531.

[6] John Cox. "Notes on option pricing I: Constant elasticity of variance diffusions". In: *Unpublished note, Stanford University, Graduate School of Business* (1975).

[7] John Crank and Phyllis Nicolson. "A practical method for numerical evaluation of solutions of partial differential equations of the heat-conduction type". In: *Mathematical Proceedings of the Cambridge Philosophical Society*. Vol. 43. 1. Cambridge University Press. 1947, pp. 50–67.

[8] Hans Follmer. "Hedging of contingent claims". In: *Applied stochastic analysis* 5 (1991), p. 389.

[9] George E Forsythe and Wolfgang R Wasow. "Finite Difference Methods". In: *Partial Differential* (1960).

[10] Andreas J Grau. "Applications of least-squares regressions to pricing and hedging of financial derivatives". PhD thesis. Technische Universität München, 2008.

[11] Igor Halperin. "QLBS: Q-Learner in the Black-Scholes(-Merton) Worlds". In: *arXiv e-prints*, arXiv:1712.04609 (Dec. 2017), arXiv:1712.04609. arXiv: 1712.04609 [q-fin.CP].

[12] HA Hauksson and ST Rachev. "The GARCH-stable option pricing model". In: *Mathematical and computer modelling* 34.9-11 (2001), pp. 1199–1212.

[13] Steven L Heston. "A closed-form solution for options with stochastic volatility with applications to bond and currency options". In: *The review of financial studies* 6.2 (1993), pp. 327–343.

[14] Vadim Linetsky and Rafael Mendoza. "Constant elasticity of variance (CEV) diffusion model". In: *Encyclopedia of Quantitative Finance* (2010).

[15] Jing Peng and Ronald J Williams. "Incremental multi-step Q-learning". In: *Machine Learning Proceedings 1994*. Elsevier, 1994, pp. 226–232.

[16] Martin Schweizer. *A guided tour through quadratic hedging approaches*. Tech. rep. SFB 373 Discussion Paper, 1999.