이학 석사 학위논문

On the rough volatility and optimal liquidation

(거친 변동성과 최적의 유동화에 관하여)

2021년 2월

서울대학교 대학원

수리과학부

한민규

On the rough volatility and optimal liquidation

(거친 변동성과 최적의 유동화에 관하여)

지도교수 박형빈

이 논문을 이학 석사 학위논문으로 제출함

2020년 10월

서울대학교 대학원

수리과학부

한민규

한민규의 이학 석사 학위논문을 인준함

2020년 12월

위 원 장 강 판 기 (인)

부 위 원 장 박 형 빈 (인)

위 원 서 안 서 (인)

On the rough volatility and optimal liquidation

A dissertation
submitted in partial fulfillment
of the requirements for the degree of
Master of Science
to the faculty of the Graduate School of
Seoul National University

by

Minkyu Han

Dissertation Director : Professor Hyungbin Park

Department of Mathematical Sciences
Seoul National University

February 2021

Abstract

In this thesis, we introduce rough volatility analysis using implied volatility and optimal liquidation strategy for local volatility model, and provide deep learning method to find the optimal liquidation strategy. In real financial markets, the dynamics of volatility is itself is a very important subject especially in derivatives markets. Thus, to provide a better understanding on derivatives, one needs to consider the roughness of volatility dynamics, which represents the memory effect of volatility. And, in real portfolio management, it requires to trade carefully with large amount of stocks. Therefore, it is essential to study how to evaluate and optimize the liquidation strategy.

To fulfill this, we first consider the fractional Brownian motion as the origin of roughness of volatility dynamics, and extract the roughness from the option data. And, to study the optimal liquidation, we first introduce the optimal liquidation problem under local volatility model. Furthermore, using dynamic programming, we obtain the associated PDE which is difficult to solve analytically. At last, we study deep learning methods to solve PDE and optimize the liquidation strategy

Key words: Volatility, implied volatility, rough volatility, fractional Brownian motion, volatility skew, geometric Brownian motion, local volatility model, optimal control, HJB equation, deep learning, neural network, gradient descent method
Student Number: 2018-26732

Contents

Chapter 1

Introduction

In financial markets, especially derivatives markets, the log-price process is often modeled as the following form:

$$d \log S_t = \mu_t dt + \sigma_t dW_t$$

where S_t is the stock price, μ_t is the drift term, σ_t is the volatility process, and W_t is a Brownian motion. Here, the volatility process σ_t is the most essential process to analyze. In the Black-Sholes framework, the volatility σ_t is assumed to be constant or a deterministic function of time [5]. In [4], the volatility is assumed to be a deterministic function of the time and underlying price. However, such models have limitations to represent the dynamics of volatility processes such as roughness. A recent series of researches has shown evidence that the stock volatility is rough, in the sense that the volatility process has no long-memory and thus has rougher path than the path of Brownian motion [8, 2, 3]. The rough models of stochastic volatility process mainly assume the volatility is driven by the fractional Brownian motion, not a Brownian motion. Empirical estimates of roughness in [8, 1] are rougher than a Brownian motion. In [8, 1], the roughness is estimated by the realized roughness, which is obtained by moments of variance of stock price process. In this thesis, as introduce by [10], we estimate the roughness by the implied roughness, which is obtained from

European option price data using the property of implied volatility given in [11]. Our result on the JPMorgan Chase stock are presented in Chapter 2.

In financial market, or economics, it is usually assumed that individuals cannot affect the large market price of an asset. However, in real world, an individual's trade could affect on the market price. It is a topic related to the microstructure of market, rather than the macrostructure. Therefore, in portfolio management, liquidating a huge amount of shares in a certain time interval, we cannot ignore the effect on the asset price by the trading strategy. [12, 15] provide an optimal liquidation strategy for Brownian motion stock price with linear effect of the trading speed. Also, [7] has proved that under geometric Brownian motion model, constant selling strategy is the optimal liquidation strategy optimizing the expected total cash flow. Furthermore, [14] provides the optimal liquidation for geometric Brownian motion price model with risk criterion VaR. In this thesis, we present this optimal liquidation problem as an optimal control problem as [13]. Using the dynamic programming, [13] suggests the optimal control problem as a partial differential equation, which is called the Hamilton-Jacobi-Bellman (HJB) equation. [14] provides an exact solution of the HJB equation under geometric Brownian motion model. However, as mentioned in the above paragraph, the geometric Brownian motion model is not enough to analyze the volatility dynamics. Thus, we aim to solve the HJB equation for more general volatility model. In Chapter 3, we present the HJB equation for local volatility model and study about the relation with optimal liquidation.

As mentioned in the above paragraph, the optimal liquidation problem leads us to solving a non-linear PDE, which is difficult to solve analytically. Thus, we need a numerical way to solve the HJB equation regarding to the optimal liquidation problem. [17, 18, 19] and [20] propose to use neural networks to solve PDEs and ODEs. Moreover, [23] proved that 1 hidden

layer neural network approximation converges to the solution of PDE in strong sense, with certain conditions on the PDE. In Chapter 4, we present deep learning methods to solve PDE, and show the numerical results on the optimal liquidation problem for geometric Brownian motion model and local volatility model.

Chapter 2

On the rough volatility

In financial markets, especially derivatives markets, the volatility of stock plays an important role for pricing. [2], [5] give different pricing formulas due to different volatility dynamics. Therefore, it is quite essential to understand the dynamics of volatility.

Assuming geometric Brownian motion, the Black-Scholes formula gives the following option pricing formula:

$$c(t,S) = SN(d_+(T-t,S)) - Ke^{-r(T-t)}N(d_-(T-t,S)); \text{ call} \quad (2.0.1)$$

$$p(t,S) = Ke^{-r(T-t)}N(-d_-(T-t,S)) - SN(-d_+(T-t,S)); \text{ put} \quad (2.0.2)$$

where $d_\pm(\tau,S) = \frac{1}{\sigma\sqrt{\tau}}\left[\log\frac{S}{K} + (r \pm \frac{\sigma^2}{2})\tau\right]$ and $N(y) = \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{y} e^{-\frac{z^2}{2}}dz$.

Here the volatility σ is assumed to be constant and thus the option pricing is rather easy. In practice, the implied volatility that gives the actual option prices using (2.0.1), (2.0.2) is a popular estimation of volatility. However, since the implied volatility of a stock varies for time, the implied volatility can be regarded as just another form of option prices, not the actual volatility. Furthermore, there are empirical results [8] that the actual volatility process is rougher than the Brownian motion which is the limit of random walk. The basic model of rough process is the fractional

Brownian motion $W^{(H)}$ with $H < \frac{1}{2}$. Under the fractional Brownian motion volatility model, [11] gives a specific form of implied volatility. Thus, in this chapter, we study about the implied volatility and how to detect the roughness of volatility from implied volatility.

The rest of this chapter of organized as follows. In Section 2.1, we study about Black-Scholes formula for European options, the implied volatility and fractional Brownian motion. In Section 2.2, we present how to extract roughness from option data of a stock. In Section 2.3, we present an empirical result on roughness of JPMorgan Chase stock in 2012. Finally, note that this chapter is based on [8, 9, 10, 11].

2.1 Preliminaries

In this section, we study European option pricing and calculating implied volatility from option data of a stock, and a stochastic process which has rougher or smoother path compared to the Brownian motion, the fractional Brownian motion.

2.1.1 European option price

Assume that a stock price S_t has the following dynamics GBM (Geometric Brownian Motion):

$$dS_t = \mu S_t dt + \sigma S_t dW_t \tag{2.1.1}$$

where $\mu \in \mathbb{R}, \sigma > 0$.

A European option for the stock is a Borel function $\Phi : \mathbb{R} \to \mathbb{R}$ so that the payoff $\Phi(S_T)$ is an $\mathcal{F}_T$-measurable random variable. The pricing of this European option before the maturity T is a main topic of mathematical finance. [5] propose a pricing via solving the following PDE:

$$\begin{aligned} V_t(t,S) + rSV_S(t,S) + \frac{1}{2}\sigma^2 V_{SS}(t,S) = rV(t,S), \quad \text{for } t < T, \\ V(T,S) = \Phi(S) \end{aligned} \tag{2.1.2}$$

where $r > 0$ is the risk-free interest rate.

Also, the risk-neutral pricing [6] suggests the following pricing formula:

$$V(t, S_t) = \mathbb{E}^{\mathbb{Q}}\left[e^{-r(T-t)}\Phi(S_T)|\mathcal{F}_t\right] \tag{2.1.3}$$

where $\mathbb{Q}$ is a risk-neutral probability measure.

In case of call and put options with strike price K, we have that $\Phi(S) = (S-K)^+$ or $\Phi(S) = (K-S)^+$. In that case, (2.1.2) and (2.1.3) both suggest the following pricing formula:

$$c(t, S) = SN(d_+(T-t, S)) - Ke^{-r(T-t)}N(d_-(T-t, S)); \text{ call} \tag{2.1.4}$$

$$p(t, S) = Ke^{-r(T-t)}N(-d_-(T-t, S)) - SN(-d_+(T-t, S)); \text{ put} \tag{2.1.5}$$

where $d_\pm(\tau, S) = \frac{1}{\sigma\sqrt{\tau}}\left[\log\frac{S}{K} + (r \pm \frac{\sigma^2}{2})\tau\right]$ and $N(y) = \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{y} e^{-\frac{z^2}{2}}dz$.

Note that $\frac{\partial c}{\partial \sigma} = \frac{\partial p}{\partial \sigma}$, which is defined as the Vega of option, is always positive, so c, p are one-to-one on σ. Therefore, we can find unique σ that makes c, p the given prices.

2.1.2 Implied Volatility

The prices (2.0.1), (2.0.2) are obtained based on the no-arbitrage principle under GBM model. This pricing formula requires stock price, time to maturity, risk-free interest rate and the volatility σ. However, in real world, since the stock price process is a stochastic process, it's quite difficult to obtain the accurate volatility σ based on the history of stock price. Therefore, there arises an alternative way to estimate the volatility via using the option prices.

The basic concept to estimate the volatility via the option prices is the assumption that the actual option prices determined by the law of supply and demand allows no-arbitrage in the financial market. This assumption gives us that the volatility makes the option prices (2.0.1), (2.0.2) same with the actual trading prices in financial market. Here, such volatility estimation is called the "implied volatility". Note that obtaining implied

volatility from (2.0.1) or (2.0.2) is to solve a non-linear one variable equation, when c, p, S, K, T, and t are given. Also, we can change the given values in the following form, to make (2.0.1), (2.0.2) simpler:

$$\begin{aligned} c/S &= N(d_+(\tau,k)) - e^{k-r\tau}N(d_-(\tau,k)) \\ p/S &= e^{k-r\tau}N(-d_-(\tau,k)) - N(-d_+(\tau,k)) \\ k &= \log K/S \\ \tau &= T-t \\ d_\pm(\tau,k) &= \frac{1}{\sigma\sqrt{\tau}}\left[k + (r \pm \frac{\sigma^2}{2})(T-\tau)\right] \end{aligned} \tag{2.1.6}$$

Therefore, we can obtain the implied volatility as following form:

$$\begin{aligned} \sigma_{\text{imp}}^{(\text{call})} &= \sigma_{\text{imp}}^{(\text{call})}(k,\tau,c/S) \\ \sigma_{\text{imp}}^{(\text{put})} &= \sigma_{\text{imp}}^{(\text{put})}(k,\tau,p/S) \end{aligned} \tag{2.1.7}$$

Here, note that $\sigma_{\text{imp}}^{(\text{call})}$ and $\sigma_{\text{imp}}^{(\text{put})}$ can be calculated differently in real financial markets.

Since (2.1.6) is non-linear equation, so we need a numerical way to solve. In this thesis, since the call and put options pricing formulas are increasing in σ, we suggest the following bisection-method to solve (2.1.6).

(Bisection method algorithm)
For increasing $f : [0,\infty) \to \mathbb{R}$, and fixed $c \in \mathbb{R}$,

1. Set $a_0 = 0$ and large $b_0 > 0$ so that $f(b_0) > c$
2. If $f(\frac{a_n+b_n}{2}) \geq c$, then let $a_{n+1} = a_n$, $b_{n+1} = \frac{a_n+b_n}{2}$
   If $f(\frac{a_n+b_n}{2}) < c$, then let $a_{n+1} = \frac{a_n+b_n}{2}$, $b_{n+1} = b_n$
3. Repeat (2) until convergence criterion satisfied

2.1.3 Fractional Brownian Motion

As calculating the implied volatility for each time, unlike the GBM, the implied volatility shows non-constant as time flows. The implied volatility

obtained from recent high frequency data, there has been observed rough path of volatility. [8] suggests that the dynamics of volatility is driven by the fractional Brownian Motion, first introduced by [9].

Definition 2.1.1. *For $H \in (0,1)$, the fractional Brownian Motion $W^{(H)}$ is defined as*

$$W_t^{(H)} = \frac{1}{\Gamma(H+\frac{1}{2})}\Big\{ \int_{-\infty}^{0} [(t-s)^{H-1/2} - (-s)^{H-1/2}] dW_s + \int_0^t (t-s)^{H-1/2} dW_s \Big\} \tag{2.1.8}$$

where W is a Brownian Motion.

The fractional Brownian Motion $W^{(H)}$ is a mean-zero Gaussian process with stationary increments and has the covariance function form of

$$\mathbb{E}\left[W_t^{(H)} W_s^{(H)}\right] = \frac{1}{2}\left(|t|^{2H} + |s|^{2H} - |t-s|^{2H}\right) \tag{2.1.9}$$

Using (2.1.9), we have the following property of moments of increments:

$$\mathbb{E}\left[\left|W_{t+\Delta}^{(H)} - W_t^{(H)}\right|^q\right] = \mathbb{E}\left[|Z|^q\right] \Delta^{qH} \tag{2.1.10}$$

where $\Delta \geq 0$, $q > 0$ and Z is a random variable with standard normal distribution.

From (2.1.10), we can see that smaller H, then the larger moments over short time interval Δ. In case of $H = \frac{1}{2}$, the fractional Brownian Motion $W^{(H)}$ becomes the Brownian motion W. Therefore, with $H < \frac{1}{2}$, then the path of $W^{(H)}$ is rougher than the path of Brownian motion.

Here, [9] points it out that if $H > \frac{1}{2}$, then $W^{(H)}$ has a long-range correlation and thus, it has long-memory effect so that the path is smooth. And, if $H < \frac{1}{2}$, then the quadratic variation of infinite, so that the path is rougher than the Brownian motion.

2.2 Roughness of volatility

In this section, we propose a way to detect the roughness of volatility by estimating H assuming the volatility is driven by the fractional Brownian Motion $W^{(H)}$. To extract H from a given data, [10] suggests to use the term structure of at-the-money(ATM) skew.

At a fixed time and a fixed stock price, we can get the implied volatility described in Section 2.1.2 from data consists of various call, put options with various strike prices K, time to maturity τ. Thus, at each time, we can obtain the implied volatility $\sigma_{\text{imp}}(k, \tau)$ where $k = \log K/S$ the log-moneyness and τ the time to maturity. This is called the volatility surface, and we can get it at each available time. The ATM skew at time to maturity τ is given by

$$\phi(\tau) = \left| \frac{\partial \sigma_{\text{imp}}(k, \tau)}{\partial k} \right|_{k=0} \tag{2.2.1}$$

The martingale expansion in [11] characterizes the rate of decay of the ATM skew for a large class of rough volatility models.

$$\phi(\tau) \approx \text{constant} \times \tau^{H-1/2} \qquad \text{as } \tau \to 0 \tag{2.2.2}$$

(2.2.2) is the approach we will use to compute H from the option data. This method is quite robust because (2.2.2) is a general property for a quite large class of rough volatility models rather than a specific model. At this point, we need two computation problems.

The first problem is that no data can provide continuous set of k to calculate (2.2.1). Thus, we need to estimate the implied volatility between the available data points. In this thesis, we propose a smoothing cubic spline K/S versus σ_{imp} for various time to maturity τ. Then, the ATM skew $\phi(\tau)$ is the absolute value of slope at $K/S = 1$.

The second problem is to extract H using (2.2.2), (2.2.2) can be re-written as following:

$$\log \phi(\tau) \approx \text{constant} + (H - 1/2) \log \tau, \qquad \text{as } \tau \to 0 \tag{2.2.3}$$

Using (2.2.3), our plan to extract H is to conduct the linear regression of $\log \tau$ versus $\log \phi(\tau)$. Then, the slope is $H - 1/2$.

In the next Section, we would conduct these computations for the JP-Morgan Chase option data

2.3 Empirical Results

2.3.1 Implied volatility from option prices

In calculating implied volatility, the basic concept is describe in section 2.1.2. However, a given data set of options of a stock isn't enough to calculate the implied volatility. We need the risk-free interest rate r for each time to maturity τ. Since there is no data set containing continuous-time risk-free interest rate, we need to estimate the risk-free interest rate. In this thesis, we estimate the risk-free interest rate by smoothing cubic spline of time to maturity versus zero-coupon bond price.

Figure 2.1 shows the zero-coupon bond price estimation in 2012 in US. After estimating the risk-free interest rate, there is another problem with calculating the implied volatility. It is that the implied volatilities obtained from call option data and put option data are different as mentioned in section 2.1.2. In this thesis, we would calculate the implied volatility as following:

$$\sigma_{\text{imp}} = \begin{cases} \sigma_{\text{imp}}^{\text{call}} & \text{if } k \geq 0, \\ \sigma_{\text{imp}}^{\text{put}} & \text{if } k < 0 \end{cases} \tag{2.3.1}$$

(2.3.1) is due to the aims of call options and put options. Basically, call option is a derivative for preparing the stock price rising, and put option is a derivative for preparing the stock price declining. Therefore, we would assume that the implied volatility from call option data is valid for high strike price and the implied volatility from put option data is valid for low strike price.

As a result, the implied volatility curve for moneyness K/S gets flattened at longer maturities. Figure 2.2 shows examples of volatility surfaces

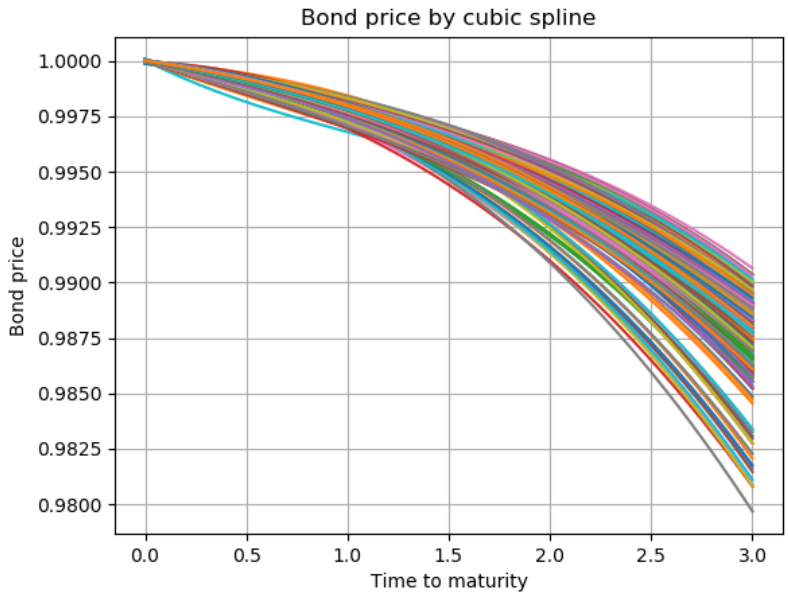

Figure 2.1: Each line represents the term structure of bond price at each day in 2012

of JPMorgan Chase.

2.3.2 ATM skew and Roughness

As shown in section 2.3.1, the ATM skew decreases as the maturity increases. Figure 2.3 shows the ATM skew and its fitting curve.

As a result, Figure 2.4 shows the distribution of roughness of JPMorgan Chase in 2012 and suggests rougher path than the volatility driven by Brownian motion.

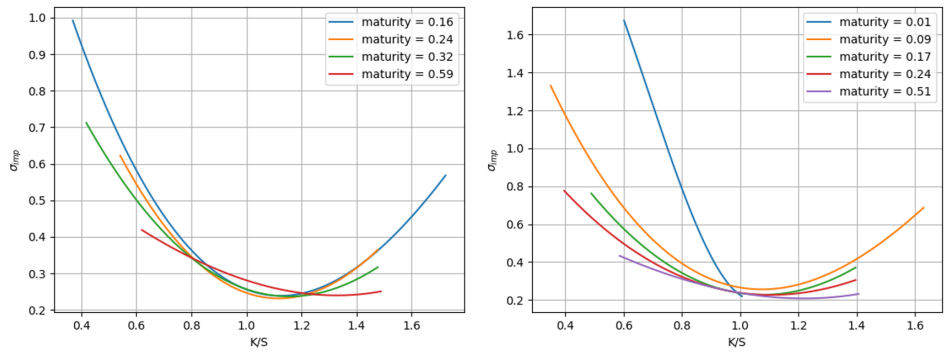

Figure 2.2: Volatility surfaces at left: Nov, 20, 2012, right: Dec, 17, 2012

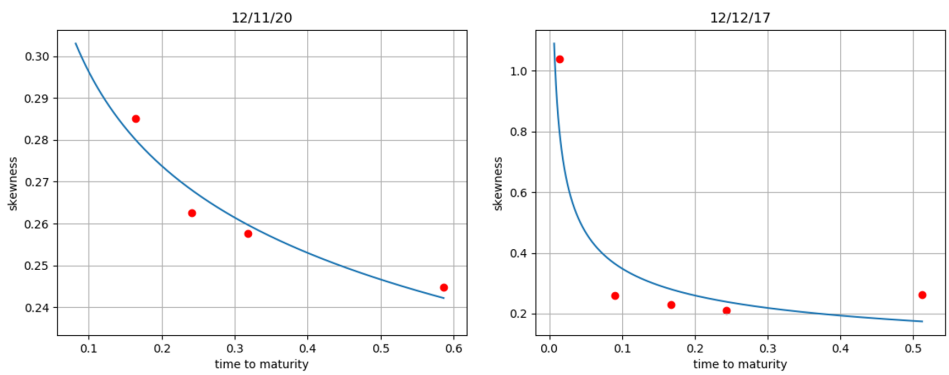

Figure 2.3: Time to maturity versus ATM skew at left: Nov, 20, 2012, right: Dec, 17, 2012, red point: actual data, blue line: fitting curve

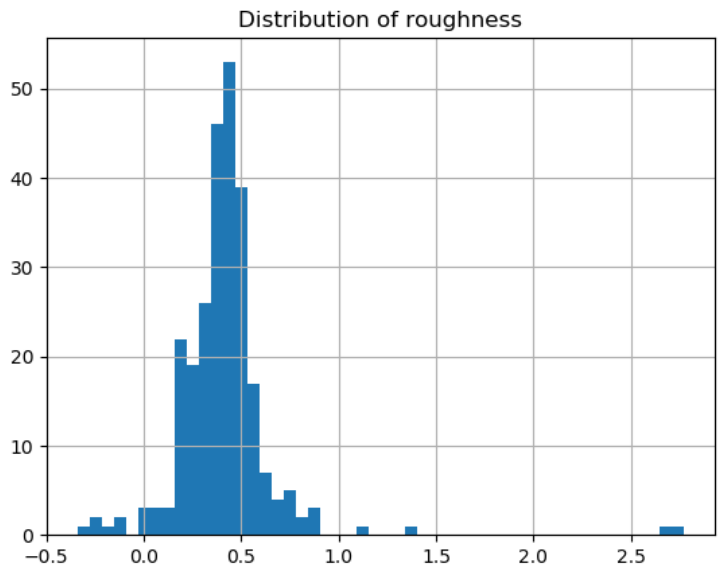

Figure 2.4: Distribution of roughness of JPMorgan Chase in 2012, average $H = 0.415$

Chapter 3

On the optimal liquidation

In most portfolio theory, the asset price is assumed to be determined by the large market, but not be affected by small individual market participant. However, if an agent, such as big bank or financial institution, deals with a huge amount of shares of an asset, then it is natural to consider the agent affects the asset price. In this chapter, we present a stochastic optimal control problem associated with the re-balance of portfolio.

Throughout this chapter, we assume an asset price process follows local volatility model:

$$dS_t = \sigma(t, S_t)S_t dW_t$$

where σ is a known local volatility function.

Even though we assume this price process has its own dynamics, if an agent participate the asset trade, then the "actual" trade price can be affected by the action of the agent. The agent could increase the actual trade price than S_t when the agent buys, or could decrease the actual trade price than S_t when the agent sells the asset. Our basic model of this effect is as following:

$$\tilde{S}_t = S_t - \eta\alpha_t - \gamma \int_0^t \alpha_s ds$$

where α is the agent's selling speed. Note that even if S_t itself rises, the

actual trade price can be very low by the poor decision of α. Throughout this chapter, we're dealing with how to find which α is poor or good for our specific reward from the trade. To be precise, finding α which maximizes the reward is the main goal of this chapter. This problem can be regarded as a continuous time Markov Decision Problem with specific model and reward function. This problem is highly related to a PDE problem which is named as Hamilton-Jacobi-Bellman equation. We introduce formulating the problem and the HJB equation associated with this problem.

The rest of this chapter of organized as follows. In Section 3.1, we study several basic concepts on optimal control problem, and then study the relation with HJB equation. In Section 3.2, we present an optimal liquidation problem as an optimal control problem. Specifically, we induce the associated HJB equation with Geometric Brownian Motion model and Local Volatility model. In Section 3.3, we consider a risk criterion issue on optimal liquidation problem, and adjust the problem by adding a suitable risk criterion. Finally, note that this chapter is based on [13, 15, 16].

3.1 Preliminaries

In this section, we study a quite general class of optimal control problems, and the associated partial differential equations.

3.1.1 Optimal control problem

In this subsection, we now go on to set a general class of optimal control problems. Let $(\Omega, \mathcal{F}, \mathbb{P})$ be a probability space, $(W_t)_{t\geq 0}$ be a d-dimensional Brownian motion, and $(\mathcal{F}_t)_{t\geq 0}$ be a filtration generated by $(W_t)_{t\geq 0}$. Let $\mu(t,x,u)$ and $\sigma(t,x,u)$ be given functions:

$$\mu : [0,\infty) \times \mathbb{R}^n \times \mathbb{R}^k \to \mathbb{R}^n, \quad \sigma : [0,\infty) \times \mathbb{R}^n \times \mathbb{R}^k \to \mathbb{R}^{n\times d}$$

For a given $x_0 \in \mathbb{R}^n$, let us consider a controlled process satisfying the following controlled dynamics:

$$\begin{aligned} &dX_t = \mu(t, X_t, u_t)dt + \sigma(t, X_t, u_t)dW_t, \quad t > 0, \\ &X_0 = x_0. \end{aligned} \tag{3.1.1}$$

where the n-dimensional process X is the state process, and u is a k-dimensional adapted process, which we would call the control process.
The first modelling problem is to determine the admissible class of control processes. This concerns how to construct an adapted control process. A natural way to construct an adapted control process is choosing a deterministic function $g : [0, \infty) \times \mathbb{R}^n \to \mathbb{R}^k$ and then define a control process u by

$$u_t = g(t, X_t)$$

Such function g is called a feedback control law, or a policy, and we will restrict the controls to be defined by policies. After we have chosen a policy g, then we can put $u_t = g(t, X_t)$ into (3.1.1) to get the following SDE:

$$\begin{aligned} &dX_t = \mu(t, X_t, g(t, X_t))dt + \sigma(t, X_t, g(t, X_t))dW_t, \quad t > 0, \\ &X_0 = x_0. \end{aligned} \tag{3.1.2}$$

Now we can define the general class of admissible policies.

Definition 3.1.1. *A policy g is called admissible if for any $(t, x) \in [0, \infty) \times \mathbb{R}^n$, the SDE*

$$\begin{aligned} &dX_s = \mu(s, X_s, g(s, X_s))dt + \sigma(s, X_s, g(s, X_s))dW_s, \quad s > t, \\ &X_t = x. \end{aligned} \tag{3.1.3}$$

has a unique solution.

The class of admissible policies is denoted by $\mathcal{U}_0$.

In most cases, we also have some control constraints. Thus, when some constraints are given, we use a smaller class $\mathcal{U}$ which is the set of admissible policy $g \in \mathcal{U}_0$ satisfying the given constraints.

We now go on to set the control problem. To set a control problem, we first need to decide the objective function of the control problem. Let us consider a given pair of functions

$$f : [0, \infty) \times \mathbb{R}^n \times \mathbb{R}^k \to \mathbb{R}, \quad \phi : \mathbb{R}^n \to \mathbb{R}$$

Now we define the value function $\mathcal{J}_0 : \mathcal{U} \to \mathbb{R}$ by

$$\mathcal{J}_0(g) = \mathbb{E}\left[\int_0^T f(t, X_t, g(t, X_t))dt + \phi(X_T)\right]$$

where X is the solution to (3.1.2).

Finally, our formal problem can be set as the problem of maximizing $\mathcal{J}_0(g)$ over all $g \in \mathcal{U}$, and we define the optimal value $\tilde{\mathcal{J}}_0$ by

$$\tilde{\mathcal{J}}_0 = \sup_{g \in \mathcal{U}} \mathcal{J}_0(g)$$

If there exists $\tilde{g} \in \mathcal{U}$ such that $\tilde{\mathcal{J}}_0 = \mathcal{J}_0(\tilde{g})$, then we call $\tilde{g}$ an optimal policy for the given problem. Our main objective is to find this optimal policy (of course if it exists) of a given control problem.

To find the optimal policy, we first find the optimal value. The methodology used to solve this problem will be that of dynamic programming. The key idea is to enlarge our problem into a larger class of problems. Now we define the following control problem

Definition 3.1.2. *For $(t, x) \in [0, \infty) \times \mathbb{R}^n$, the control problem $\mathcal{C}(t, x)$ is defined as to maximize*

$$\mathbb{E}\left[\int_t^T f(s, X_s^g, g(s, X_s^g))ds + \phi(X_T^g) \mid X_t = x\right]$$

over $g \in \mathcal{U}$ where X^g is the solution of (3.1.3)

Note that our original problem is $\mathcal{C}(0, X_0)$. Now let us define the value function and the optimal value function.

Definition 3.1.3. *For a given control problem $\mathcal{C}(t,x)$,*

1. *The value function $\mathcal{J} : [0,\infty) \times \mathbb{R}^n \times \mathcal{U} \to \mathbb{R}$ is defined by*

$$\mathcal{J}(t,x,g) = \mathbb{E}\left[\int_t^T f(s, X_s^g, g(s, X_s^g))ds + \phi(X_T^g) \mid X_t = x\right]$$

2. *The optimal value function $V : [0,\infty) \times \mathbb{R}^n \to \mathbb{R}$ is defined by*

$$V(t,x) = \sup_{g \in \mathcal{U}} \mathcal{J}(t,x,g)$$

Our main object is the optimal value function V, and we will see the connection between this control problem and a PDE, which is called the Hamilton-Jacobi-Bellman equation in the next subsection.

3.1.2 The Hamilton-Jacobi-Bellman Equation

In this subsection, we review the link between Hamilton-Jacobi-Bellman equation (often referred as HJB equation) and a control problem defined as Definition 3.1.2.

For a control problem $\mathcal{C}$, to find a PDE satisfied by the optimal value function V, let's assume the following:

$$\text{There exists an optimal policy } \tilde{g} \text{ for } \mathcal{C} \tag{3.1.4}$$

$$\text{The optimal value function } V \text{ is in } C^{1,2} \tag{3.1.5}$$

$$(t,\omega) \mapsto V_x(t, X_t^g)' \sigma(t, X_t^g, g(t, X_t^g)) \text{ is in } \mathcal{L}^2 \text{ for each policy } g \tag{3.1.6}$$

For $g \in \mathcal{U}$, $(t,x) \in (0,T) \times \mathbb{R}^n$, and small enough $h > 0$, let us consider $g^* \in \mathcal{U}$ defined as

$$g^*(s,y) = \begin{cases} g(s,y) & \text{if} \quad (s,y) \in [t, t+h] \times \mathbb{R}^n, \\ \tilde{g}(s,y) & \text{if} \quad (s,y) \in (t+h, T] \times \mathbb{R}^n. \end{cases}$$

Then, by Definition 3.1.3, we obtain the following:

$$\mathcal{J}(t,x,g^*) = \mathbb{E}\left[\int_t^{t+h} f(s,X_s^g,g(s,X_s^g))ds + V(t+h,X_{t+h}^g) \mid X_t = x\right],$$
$$V(t,x) \geq \mathcal{J}(t,x,g^*). \tag{3.1.7}$$

And also, by the Ito formula, we have

$$\begin{aligned} V(t+h,X_{t+h}^g) = V(t,x) &+ \int_t^{t+h} \left[V_t(s,X_s^g) + A^g V(s,X_s^g)\right] ds \\ &+ \int_t^{t+h} V_x(s,X_s^g)'\sigma(s,X_s^g,g(s,X_s^g))dW_s \end{aligned} \tag{3.1.8}$$

where $\mathcal{A}^g = \sum_{i=1}^n \mu(t,x,g(t,x))\frac{\partial}{\partial x_i} + \frac{1}{2}\sum_{i,j=1}^n (\sigma(t,x,g(t,x))\sigma(t,x,g(t,x))')_{i,j}\frac{\partial^2}{\partial x_i \partial x_j}$. By (3.1.7), (3.1.8), (3.1.6), and the fundamental theorem of calculus, taking limit $h \to 0$ gives the following:

$$f(t,x,g(t,x)) + V_t(t,x) + A^g V(t,x) \leq 0, \quad \text{for } g \in \mathcal{U}$$

Since the equality holds when $g = \tilde{g}$, we have the following:

$$V_t(t,x) + \sup_{g\in\mathcal{U}}\{f(t,x,g(t,x)) + A^g V(t,x)\} = 0 \tag{3.1.9}$$

Since (t,x) is an arbitrary point, we obtain the following HJB equation:

Theorem 3.1.1. *Under the same assumptions above, the following hold:*

1. *V satisfies*

$$\begin{cases} V_t(t,x) + \sup_{g\in\mathcal{U}}\{f(t,x,g(t,x)) + A^g V(t,x)\} = 0, & (t,x) \in (0,T)\times\mathbb{R}^n, \\ V(T,x) = \phi(x), & x \in \mathbb{R}^n. \end{cases} \tag{3.1.10}$$

2. *For each $(t,x) \in [0,T]\times\mathbb{R}^n$, the supremum in (3.1.10) is attained by $g = \tilde{g}$*

Note that Theorem 3.1.1 has a form of necessary condition for the corresponding control problem. Which means, if the control problem is solved, then the corresponding HJB equation holds. Since our main problem is to check if the control problem can be solved, it is not enough. Fortunately, [13] gives the verification theorem for dynamic programming which tells us that HJB equation is a sufficient condition for the control problem.

Theorem 3.1.2. *Assume that* $H, g : [0,T] \times \mathbb{R}^n \to \mathbb{R}$ *satisfy*

- H *satisfies* (3.1.6) *and* (3.1.10)
- $g \in \mathcal{U}$
- *For each* (t,x)*, the supremum in* (3.1.10) *is attained by* g

Then, H *is the optimal value function of the given control problem and* g *is an optimal policy*

As a result, it remains only to solve the corresponding HJB equation (3.1.10) for solving a given control problem.

3.2 Optimal liquidation of assets

In this section, we present an optimal asset liquidation problem as a control problem under asset price model GBM and local volatility model. We first set the optimal asset liquidation problem for a general asset price process $(S_t)_{t\geq 0}$. Suppose that we have N_0 number of asset with price process $(S_t)_{t\geq 0}$ to liquidate in time T. Then, the natural objective function we want to maximize is the expected total cash-flow during the liquidation time. We choose the selling rate $(\alpha_t)_{t\geq 0}$ to be the control and the pair $(S_t, N_t)_{t\geq 0}$ of price process $(S_t)_{t\geq 0}$ and remaining number of asset process $(N_t)_{t\geq 0}$ to be the state process. Then $dN_t/dt = -\alpha_t$ and α affects the dynamics of $(S_t)_t$. Here, the control effect on asset price dynamics can make the asset price dynamics extremely hard to analyze. To avoid this problem, we choose $(S_t)_t$ by "semi-price" such as bid-ask mid price for stock price. Then model $(S_t)_t$

not be affected by the control, but the "actual" transaction price $(\tilde{S}_t)_t$ be affected. In this thesis, the control effect on the actual transaction price is assumed to be as following:

$$\tilde{S}_t^{\alpha} = S_t - \eta\alpha_t + \gamma(N_t - N_0) \tag{3.2.1}$$

where η is the temporary impact constant and γ is the permanent impact constant. Then, the expected total cash-flow during time T can be written as following:

$$\mathcal{J}_0(\alpha) = \mathbb{E}\left[\int_0^T (S_t - \eta\alpha_t + \gamma(N_t - N_0))\alpha_t dt\right] \tag{3.2.2}$$

So the optimal liquidation problem can be viewed as the following control problem:

(Optimal liquidation problem)

$$\begin{aligned} \text{Maximize} \quad & \mathbb{E}\left[\int_0^T (S_t - \eta\alpha_t + \gamma(N_t - N_0))\alpha_t dt\right] \\ \text{over} \quad & \{\alpha \mid \int_0^T \alpha_t dt = N_0\} \end{aligned} \tag{3.2.3}$$

3.2.1 Optimal liquidation under Geometric Brownian Motion model

In this subsection, we introduce the previous result on optimal liquidation for asset under Geometric Brownian Motion model. The dynamics of state process is assumed to be as following:

$$\begin{aligned} dX_t &= d\begin{pmatrix} S_t \\ N_t \end{pmatrix} = \begin{pmatrix} 0 \\ -\alpha_t \end{pmatrix} dt + \begin{pmatrix} \sigma S_t \\ 0 \end{pmatrix} dW_t, \quad t > 0 \\ X_0 &= \begin{pmatrix} S_0 \\ N_0 \end{pmatrix} \end{aligned} \tag{3.2.4}$$

where $\sigma > 0$ is a constant and $S_t = S_0 e^{-\frac{1}{2}\sigma^2 t + \sigma W_t}$.

The corresponding (3.1.9) is:

$$V_t + \sup_{u\in\mathbb{R}}\left[u(S-\eta u+\gamma(N-N_0)) - uV_N + \frac{1}{2}\sigma^2 S^2 V_{SS}\right] = 0$$

The supremum is attained when $u = \tilde{g}(t,S,N) = \frac{S+\gamma(N-N_0)-V_N(t,S,N)}{2\eta}$. Therefore, the corresponding HJB equation is as following:

$$\begin{cases} V_t + \frac{1}{2}\sigma^2 S^2 V_{SS} + \frac{(S+\gamma(N-N_0)-V_N)^2}{4\eta} = 0, & \text{on } (0,T)\times\mathbb{R}\times\mathbb{R}, \\ V(T,S,N) = \phi(S,N), \quad (S,N)\in\mathbb{R}\times\mathbb{R}. \end{cases} \tag{3.2.5}$$

where ϕ is defined as:

$$\phi(S,N) = \begin{cases} 0 & \text{if} \quad N = 0, \\ -\infty & \text{if} \quad N \neq 0. \end{cases} \tag{3.2.6}$$

which is defined to restrict the policy to liquidate the whole shares so that $\int_0^T g(t,X_t^g)dt = N_0$ for all admissible policy g. [14] suggests the exact solution of the following form:

$$V(t,S,N) = SN + \frac{1}{2}\gamma(N^2 - 2N_0N) - \frac{\eta N^2}{T-t}, \quad (t,S,N)\in[0,T)\times\mathbb{R}\times\mathbb{R} \tag{3.2.7}$$

And the corresponding optimal policy is:

$$\tilde{g}(t,S,N) = \frac{N}{T-t}$$

Thus, we obtain the following deterministic optimal liquidation strategy with constant liquidation rate:

$$\begin{aligned} &\frac{dN_t}{dt} = -\frac{N_t}{T-t} \\ &\Rightarrow N_t = \frac{N_0(T-t)}{T} \end{aligned} \tag{3.2.8}$$

3.2.2 Optimal liquidation under local volatility model

In this subsection, we introduce the optimal liquidation for asset under local volatility model. The dynamics of state process is assumed to be as following:

$$\begin{aligned} dX_t &= d\begin{pmatrix} S_t \\ N_t \end{pmatrix} = \begin{pmatrix} 0 \\ -\alpha_t \end{pmatrix} dt + \begin{pmatrix} \sigma(t, S_t)S_t \\ 0 \end{pmatrix} dW_t, \quad t > 0 \\ X_0 &= \begin{pmatrix} S_0 \\ N_0 \end{pmatrix} \end{aligned} \tag{3.2.9}$$

where σ is a strictly positive local volatility function.

The corresponding (3.1.9) is:

$$V_t + \sup_{u \in \mathbb{R}} \left[u(S - \eta u + \gamma(N - N_0)) - uV_N + \frac{1}{2}\sigma^2(t, S)S^2 V_{SS} \right] = 0$$

The supremum is attained when $u = \tilde{g}(t, S, N) = \frac{S+\gamma(N-N_0)-V_N(t,S,N)}{2\eta}$. Therefore, the corresponding HJB equation is as following:

$$\begin{cases} V_t + \frac{1}{2}\sigma^2(t, S)S^2 V_{SS} + \frac{(S + \gamma(N - N_0) - V_N)^2}{4\eta} = 0, & \text{on } (0, T) \times \mathbb{R} \times \mathbb{R}, \\ V(T, S, N) = \phi(S, N), \quad (S, N) \in \mathbb{R} \times \mathbb{R}. \end{cases} \tag{3.2.10}$$

which is similar with (3.2.5) except that $\sigma = \sigma(t, S)$ is not constant in this case. Surprisingly, the solution (3.2.7) for Geometric Brownian Motion case also solves (3.2.10).

$$V(t, S, N) = SN + \frac{1}{2}\gamma(N^2 - 2N_0 N) - \frac{\eta N^2}{T - t}, \quad (t, S, N) \in [0, T) \times \mathbb{R} \times \mathbb{R}$$

And the corresponding optimal policy is again:

$$\tilde{g}(t, S, N) = \frac{N}{T - t}$$

Thus, we obtain the same deterministic optimal liquidation strategy with constant liquidation rate again:

$$\begin{aligned} &\frac{dN_t}{dt} = -\frac{N_t}{T-t} \\ &\Rightarrow N_t = \frac{N_0(T-t)}{T} \end{aligned} \tag{3.2.11}$$

This result is quite surprising because the volatility of asset price dynamics doesn't affect the optimal liquidation strategy. However, it is quite natural that the strategy should be different if the volatility has different dynamics because the volatility obviously affects the distribution of asset price. Therefore, we need some adjustments for the optimal liquidation problem.

3.3 Adjusted optimal liquidation problem

In this section, we observe the origin of issue that the optimal liquidation strategy doesn't depend on the volatility. And make some adjustments to settle this problem.

The main origin of this issue is that our objective function (3.2.2) is just the expectation of total cash-flow. Throughout Section 3.2, we have maximized the expectation, but there is a huge risk that the total cash-flow might be poor depending on the volatility σ. Thus, we need to adjust the objective function with risk.

In this section, we introduce a risk criterion for an optimal liquidation, and proceed solving an adjusted optimal liquidation problem under Geometric Brownian Motion model and local volatility model.

3.3.1 Risk Criterion - VaR

In this subsection, we introduce a risk criterion defined by the time-average of instantaneous risk. First, let us define the instantaneous risk. We define the instantaneous risk at time t holding N shares of stock with price S_t as

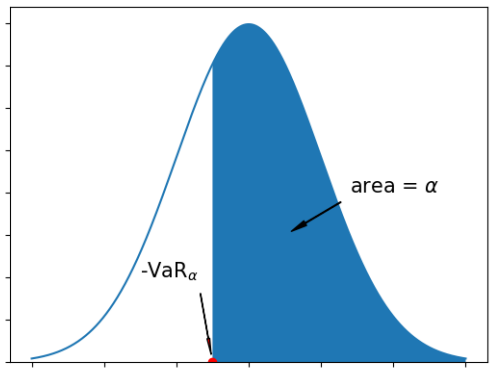

Figure 3.1: pdf of Y, the red point represents $-\mathrm{VaR}_\alpha$

the maximum loss for a certain time period Δ within probability α, which the value-at-risk (VaR) at confidence level $\alpha \in (0,1)$,

$$\mathrm{VaR}_\alpha(Y) = -\sup\{m \in \mathbb{R} \mid \mathbb{P}(Y \geq m) \geq \alpha\}$$

where $Y = N(S_{t+\Delta} - S_t)$ is the profit of holding the stocks for time period Δ. Figure 3.1 shows the definition of VaR.

Under GBM or local volatility model, when we consider small $\Delta << 1$, it is clear that the instantaneous risk is proportional to NS_t. Therefore, it is natural to choose the risk criterion as $\lambda\mathbb{E}\left[\int_0^T S_t N_t dt\right]$ with risk-aversion constant $\lambda > 0$. Thus, we obtain the following adjusted optimal liquidation problem:

(Adjusted optimal liquidation problem)

$$\begin{aligned} \text{Maximize} \quad & \mathbb{E}\left[\int_0^T ((S_t - \eta\alpha_t + \gamma(N_t - N_0))\alpha_t - \lambda N_t S_t)dt\right] \\ \text{over} \quad & \{\alpha \mid \int_0^T \alpha_t dt = N_0\} \end{aligned} \tag{3.3.1}$$

In the next two subsections, we derive the corresponding HJB equation for GBM and local volatility model, and discuss about the solution.

3.3.2 Adjusted optimal liquidation problem under Geometric Brownian Motion model

In this subsection, we introduce the previous result on the risk-adjusted optimal liquidation for asset under Geometric Brownian Motion model. The dynamics of state process is assumed to be as same as (3.2.4). The corresponding (3.1.9) is:

$$V_t + \sup_{u\in\mathbb{R}} \left[u(S - \eta u + \gamma(N - N_0)) - \lambda NS - uV_N + \frac{1}{2}\sigma^2 S^2 V_{SS} \right] = 0$$

The supremum is attained when $u = \tilde{g}(t, S, N) = \frac{S+\gamma(N-N_0)-V_N(t,S,N)}{2\eta}$. Therefore, the corresponding HJB equation is as following:

$$\begin{cases} V_t + \frac{1}{2}\sigma^2 S^2 V_{SS} - \lambda NS + \frac{(S + \gamma(N - N_0) - V_N)^2}{4\eta} = 0, & \text{on } (0, T) \times \mathbb{R}^2, \\ V(T, S, N) = \phi(S, N), \quad (S, N) \in \mathbb{R}^2. \end{cases} \tag{3.3.2}$$

where ϕ is defined as (3.2.6) which is defined to restrict the policy to liquidate the whole shares so that $\int_0^T g(t, X_t^g)dt = N_0$ for all admissible policy g. [14] suggests the exact solution of the following form:

$$\begin{aligned} V(t, S, N) = SN + \frac{1}{2}\gamma(N^2 - 2N_0 N) - \frac{\eta N^2}{T - t} - \frac{1}{2}\lambda(T - t)NS \\ + \frac{\lambda^2}{8\eta\sigma^6} S^2 \left(e^{\sigma^2(T-t)} - 1 - \sigma^2(T - t) - \frac{1}{2}\sigma^4(T - t)^2 \right) \\ \text{for } (t, S, N) \in [0, T) \times \mathbb{R}^2 \end{aligned} \tag{3.3.3}$$

And the corresponding optimal policy is:

$$\tilde{g}(t, S, N) = \frac{N}{T - t} + \frac{\lambda}{4\eta}(T - t)S$$

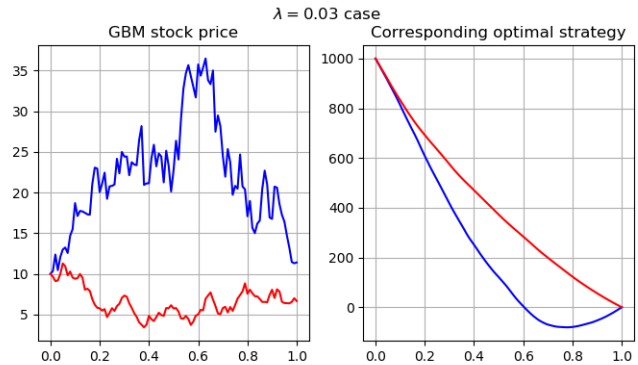

Figure 3.2: Two realizations of GBM with $\lambda = 0.03$

Thus, we obtain the following stochastic optimal liquidation strategy:

$$\begin{aligned} &\frac{dN_t}{dt} = -\frac{N_t}{T-t} - \frac{\lambda}{4\eta}(T-t)S_t \\ &\Rightarrow N_t = \frac{N_0(T-t)}{T} - \frac{\lambda(T-t)}{4\eta}\int_0^t S_u du \end{aligned} \tag{3.3.4}$$

Figure 3.2 shows two realizations of GBM optimal liquidation problem. Note that the selling rate is higher than constant selling rate at first, and lower than the constant selling rate at final. Also, when the stock price is high, the selling rate becomes higher, and even short position can also happen when the stock price increases.

Figure 3.3 shows the distribution of total cash-flow for various risk-aversion constant λ by 10^8 simulations. Note that as λ increases, the probability gets concentrated more. However, the probability for high end decreases and the probability for low end increases, which implies that the choice of risk-aversion constant affects the result of optimal liquidation strategy.

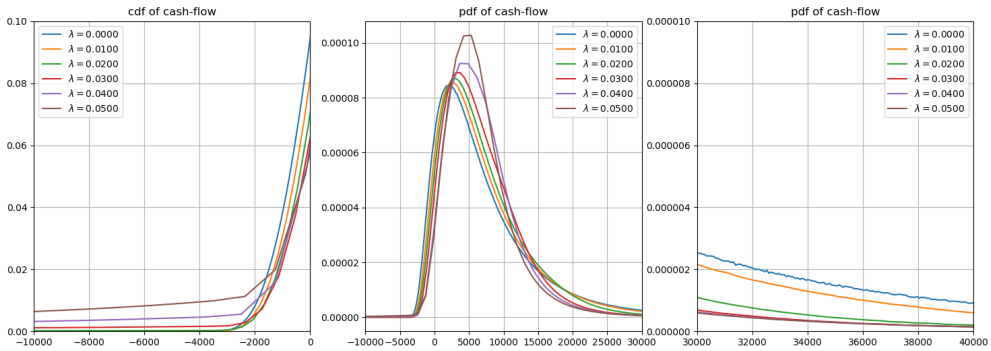

Figure 3.3: Distribution of total cash-flow with various λ. Left: Cdf for negative cash-flow. Middle: Pdf of cash-flow. Right: Pdf of high cash-flow

3.3.3 Adjusted optimal liquidation problem under local volatility model

In this subsection, we introduce the risk-adjusted optimal liquidation for asset under local volatility model. The dynamics of state process is assumed to be as same as (3.2.9). The corresponding (3.1.9) is:

$$V_t + \sup_{u\in\mathbb{R}}\left[u(S-\eta u+\gamma(N-N_0)) - \lambda NS - uV_N + \frac{1}{2}\sigma^2(t,S)S^2V_{SS}\right] = 0$$

The supremum is attained when $u = \tilde{g}(t,S,N) = \frac{S+\gamma(N-N_0)-V_N(t,S,N)}{2\eta}$. Therefore, the corresponding HJB equation is as following:

$$\begin{cases} V_t + \frac{1}{2}\sigma^2(t,S)S^2V_{SS} - \lambda NS + \frac{(S+\gamma(N-N_0)-V_N)^2}{4\eta} = 0, \\ \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{on } (0,T)\times\mathbb{R}^2, \\ V(T,S,N) = \phi(S,N), \qquad (S,N)\in\mathbb{R}^2. \end{cases} \tag{3.3.5}$$

which is similar with (3.3.2) except that $\sigma = \sigma(t,S)$ is not constant in this case. In this case, it is not easy to solve since σ itself is not constant. Recall

that the solution for GBM case (3.2.7) is also a solution of (3.2.10) for local volatility model case in section 3.2.2. In section 3.3.2, we have observed that the solution (3.3.3) for GBM case contains the solution (3.2.7) for not adjusted optimal liquidation problem. Therefore, let us denote (3.2.7) by V_0, and let $V = V_0 + U$. Then, (3.3.5) can be re-written as following:

$$\begin{cases} U_t + \frac{1}{2}\sigma^2(t,S)S^2U_{SS} - \lambda NS + \frac{1}{4\eta}U_N^2 - \frac{NU_N}{T-t} = 0, & \text{on } (0,T)\times\mathbb{R}^2, \\ U(T,S,N) = 0, \qquad (S,N) \in \mathbb{R}^2. \end{cases} \tag{3.3.6}$$

Here, the terminal condition can be any other function with zero value at $(S,N) = (0,0)$. As a result, only solving (3.3.6) remains. However, in general, it is quite hard to solve a non-linear partial differential equation such as (3.3.6). In this thesis, we propose a way to solve this non-linear pde by using deep learning method. In the next chapter, we will introduce the required deep learning method for solving (3.3.6) after forming differently.

Chapter 4

Deep learning method for optimal liquidation

In most area of mathematics, there are several problems that we cannot solve analytically. In that case, we usually find an "approximate" solution, which is not accurate solution but obtainable in numerical way. Here, the "approximate" solution means a sequence converging to the accurate solution in some sense. One of the most requiring approximate solution area is the area of partial differential equations. A non-linear pde such as (3.3.6) is a typical example of a problem requires "approximate" solution.

The basic method to obtain "approximate" solution of a pde is the finite difference method. The finite difference method changes the pde to a system of equations. However, the finite difference method costs us a huge amount of computations. This problem comes from the forming meshes. Therefore, there arises need for an alternative method which requires less computations.

The alternative method we propose in this thesis is the deep learning method. [17, 20, 23] propose to use neural networks to solve PDEs. In this chapter, we introduce Neural network approximation method and Gradient descent method for optimization, and form a PDE into a Neural network approximation problem.

The rest of this chapter is organized as follows. In Section 4.1, we provide basic concepts and methods of Neural Network and Gradient Descent methods. In Section 4.2, we study some previous results on solving PDE using neural network, and form the PDE 3.3.6 differently. In Section 4.3, we propose an algorithm for solving a PDE, and numerical results on (3.3.6) for GBM and CEV model. Finally, note that this chapter is based on [23].

4.1 Preliminaries

In this section, we study some deep learning methods which is the Neural network approximation, and the Gradient Descent method for the relevant optimization problem in Neural network approximation.

4.1.1 Neural Network

In this subsection, we introduce basic concepts of Neural network and its properties. The Neural network aims to approximate complex non-linear functions defined on a finite-dimensional space. In contrast with usual additive approximation theory, it uses compositions of rather simple non-linear functions.

Figure 4.1 shows a Neural network system with m hidden layers from d-dimensional space to k-dimensional space. The input layer represents the domain of the Neural network, and the output layer represents the range of the Neural network. And each hidden layer accepts signal from the previous layer and emits signal to the next layer. In signal emission, each neuron of each layer emits signal as the neuron activation in our brains using non-linear activation functions such as sigmoid, tanh, ReLU, ELU.

Figure 4.2 shows several non-linear activation functions which are similar with neuron activation in our brains.

In accepting signal, each neuron is assumed to accept the signals from previous neurons linearly. Thus, we need to set parameters of layer's links which are called the weight and bias. Let W_j and b_j be the weight and

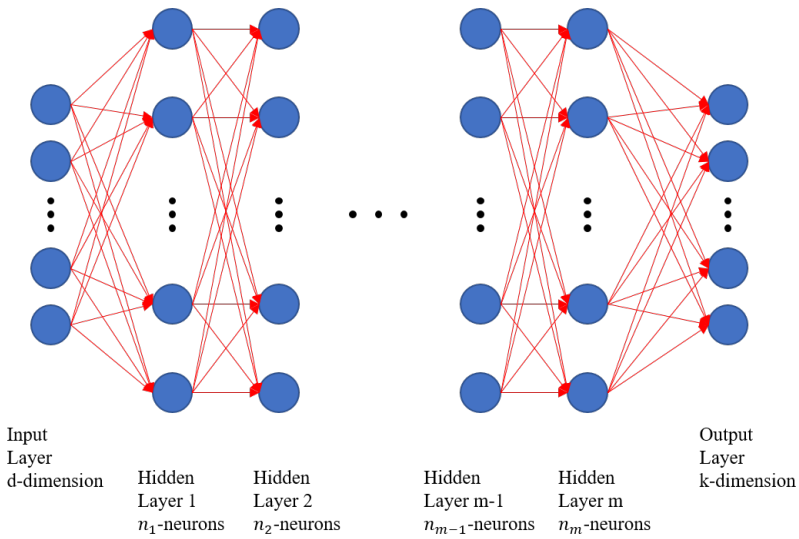

Figure 4.1: Neural Network with m hidden layers

bias of j'th hidden layer's acceptance and let W be the final weight at the output layer. Then $W_1 \in \mathbb{R}^{n_1 \times d}$, $b_1 \in \mathbb{R}^{n_1}$, $W_2 \in \mathbb{R}^{n_2 \times n_1}$, $b_2 \in \mathbb{R}^{n_2}$,..., $W_m \in \mathbb{R}^{n_m \times n_{m-1}}$, $b_m \in \mathbb{R}^{n_m}$, $W \in \mathbb{R}^{k \times n_m}$. Now we can write the Neural network as following:

$$\Phi(x) = W\sigma(W_m\sigma(W_{m-1}\sigma(\cdots\sigma(W_1x + b_1)\cdots) + b_{m-1}) + b_m) \quad (4.1.1)$$

where $x \in \mathbb{R}^d$ and activation function σ is defined component-wisely. (4.1.1) says that the Neural network varies by the parameters $W, W_1, ...W_m, b_1, ...b_m$. The Neural network suggests an approximation by choosing suitable parameters. Therefore, Neural network approximation leaves us finding the "optimal" parameters. In most cases, the "optimal" means to minimize(maximize) the cost(value) using a given set of training data. However, in general, an optimization requires a huge computation. Thus, in the next subsection, we introduce an optimization method which is called the gradient descent method in concept, and some alternative variation with smaller computations.

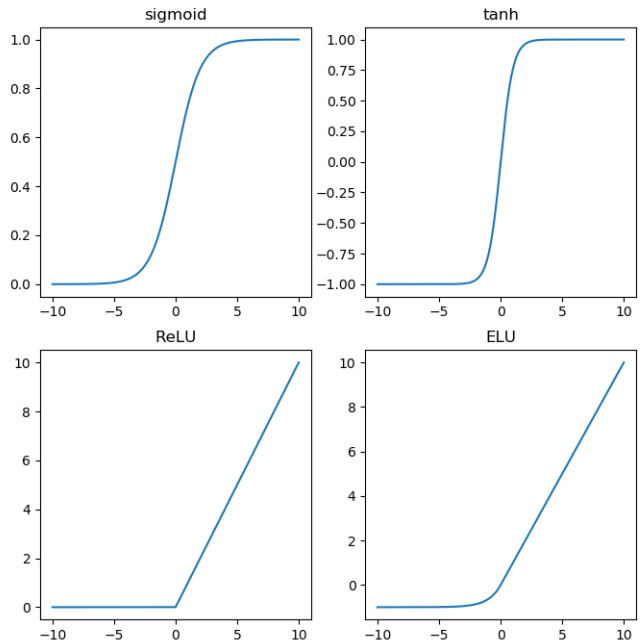

Figure 4.2: Various activation functions

4.1.2 Gradient Descent Method

Approximation by Neural network requires an optimization with respect to a set of parameters, which we can write as the following form:

$$\theta^* \in \arg\min_{\theta} J(X;\theta)$$

where J is a given cost function with a set of parameters θ, and X is a training set.

In general, it is quite difficult to achieve the optimal θ^* directly. Thus, it requires a numerical way to find the optimal θ^*. Here, we introduce a classical method in optimization which is often referred to as Gradient Descent method.

The idea of the Gradient Descent method is to move θ toward the "fastest" direction to reduce the cost function value starting from an arbitrary θ_0. Here, the "fastest" direction comes from the following:

$$J(X;\theta + d\theta) - J(X;\theta) \approx \nabla_\theta J(X;\theta) \cdot d\theta \tag{4.1.2}$$

Since our aim is to reduce the value of cost function J, it is the "fastest" direction to move θ toward the opposite direction of $\nabla_\theta J(X;\theta)$, i.e., toward the direction of $-\nabla_\theta J(X;\theta)$. Since (4.1.2) is useful for only small $d\theta$, the Gradient Descent method leads to update

$$\theta \to \theta - \gamma \nabla_\theta J(X;\theta)$$

where γ is small stepsize which is called the learning rate.

This method is also called the Batch Gradient Descent method, and its algorithm is as following:

1. Initialize $\theta = \theta_0$
2. Take a descent step with learning rate γ_n ; $\theta_{n+1} = \theta_n - \gamma_n \nabla_\theta J(X;\theta_n)$
3. Repeat (2) until convergence criterion is satisfied

This algorithm is stable, but, in case of the size of training set X is huge, the Batch Gradient Descent algorithm shows very slow convergence speed and also the computation can be very time consuming. Thus, we suggest an alternative way.

Let us consider the case of the training set $X = \{x_j\}_{j=1}^M$ has batch-size M. And assume that the cost function J has the following form:

$$J(X;\theta) = \sum_{j=1}^{M} C_j(x_j;\theta) \tag{4.1.3}$$

which means that the total cost is sum of individual cost of each training data. The alternative idea is to reduce some part of this cost at each step, not the whole cost function. The simplest method, which is called the Stochastic Gradient Descent method, gives the following algorithm:

1. Initialize $\theta = \theta_0$
2. Choose $x_j \in X$ randomly
3. Take a descent step with learning rate γ_n ; $\theta_{n+1} = \theta_n - \gamma_n \nabla_\theta C_j(x_j; \theta_n)$
4. Repeat (2)-(3) until convergence criterion is satisfied

The Stochastic Gradient Descent algorithm requires computation only for single random point in the training set X. Therefore, it is very fast, but is an unstable algorithm.

In the above two Gradient Descent methods, we can see there is a trade-off between stability and computation speed. As another alternative way to balance between stability and speed, the Mini-Batch Gradient Descent method gives the following algorithm:

1. Initialize $\theta = \theta_0^{(0)}$ and choose a positive integer M_b that divides M
2. Divide X into $\frac{M}{M_b}$ subsets $\{X_k\}_{k=1}^{\frac{M}{M_b}}$ of M_b elements
3. For $k = 1, 2, ..., \frac{M}{M_b}$, take a descent step with learning rate $\gamma_n^{(k)}$;
$$\theta_n^{(k)} = \theta_n^{(k-1)} - \frac{\gamma_n^{(k)}}{M_b} \nabla_\theta \sum_{x_j \in X_k} C_j(x_j; \theta_n^{(k-1)})$$
4. Set $\theta_{n+1}^{(0)} = \theta_n^{(\frac{M}{M_b})}$
5. Repeat (2)-(4) until convergence criterion is satisfied

The Gradient Descent methods we've just introduced above are the rather simplest algorithms that have been constructed to find optimal parameters. In this thesis, for the sake of simplicity and computation speed, we exploit the Stochastic Gradient Descent method even though its unstability. Fortunately, [22] shows that under some technical conditions, the Stochastic Gradient Descent method gives the convergence of θ_n to a critical point of $J(X; \cdot)$ as $n \to \infty$ in the sense of:
$$\lim_{n \to \infty} \|\nabla_\theta J(X; \theta_n)\| = 0$$

4.2 Neural Network Approximation for solving PDE

In this section, we see some previous results on solving PDE using Neural network approximation. At last, we form the Neural network approximation problem for the optimal liquidation problem in Section 3.3.3.

4.2.1 Neural Network Approximation problem

In this subsection, we study some previous results on solving parabolic PDE by Neural network approximation based on [23]. Consider a bounded set $\Omega \subset \mathbb{R}^d$ with smooth boundary $\partial\Omega$ and denote $\Omega_T = (0,T] \times \Omega$ and $\partial\Omega_T = (0,T] \times \partial\Omega$. Let us consider the following PDE (possibly non-linear):

$$\begin{aligned} \frac{\partial u}{\partial t}(t,x) + \mathcal{L}u(t,x) = 0, \qquad & \text{for } (t,x) \in \Omega_T \\ u(0,x) = u_0(x), \qquad & \text{for } x \in \Omega \\ u(t,x) = g(t,x), \qquad & \text{for } (t,x) \in \partial\Omega_T \end{aligned} \tag{4.2.1}$$

Of course the solution $u(t,x)$ is unknown, and our plan is to approximate $u(t,x)$ by $f(t,x)$ which minimizes the cost function J which is defined by

$$J(f) = \left|\left| \frac{\partial f}{\partial t} + \mathcal{L}f \right|\right|^2_{2,\Omega_T} + ||f-g||^2_{2,\partial\Omega_T} + ||f(0,\cdot) - u_0||^2_{2,\Omega} \tag{4.2.2}$$

Here, we will find the optimal f over $\mathcal{N}_n$, the class of neural networks with a single hidden layer and n neurons, which is defined as the following:

$$\mathcal{N}_n = \left\{ h : \mathbb{R} \times \mathbb{R}^d \to \mathbb{R} | h(t,x) = \sum_{j=1}^{n} b_j \sigma(a_{0,j} t + \sum_{k=1}^{d} a_{k,j} x_k + c_j) \right\} \tag{4.2.3}$$

where $\sigma : \mathbb{R} \to \mathbb{R}$ is a non-linear activation function such as a sigmoid, tanh, ReLU, or ELU in Figure 4.2.

Let $f_n \in \arg\min_{f \in \mathcal{N}_n} J(f)$, then [23] proved that, under certain conditions, there exists $f_n \in \mathcal{N}_n$ such that $J(f_n) \to 0$, as $n \to \infty$, and $f_n \to u$, as $n \to \infty$, strongly in, $L^s(\Omega_T)$, with any $s < 2$.

[23] provides proofs for the existence of $f_n \in \mathcal{N}_n$ and the convergence of f_n to u under certain conditions for (4.2.1). Here, we present the results for the case of $\mathcal{L}u(t,x) = -\text{div}(\alpha(t,x,u(t,x),\nabla u(t,x))) + \gamma(t,x,u(t,x),\nabla u(t,x))$:

Theorem 4.2.1. *(Theorem 7.1 in [23])*
Choose the activation function σ in $\mathcal{C}^2(\mathbb{R})$, bounded and non-constant, and let $\mathcal{N} = \bigcup_{n\geq 1} \mathcal{N}_n$. Suppose Ω_T is compact, and assume that

- (4.2.1) *has a unique classical solution in $\mathcal{C}(\overline{\Omega_T}) \bigcap \mathcal{C}^{1+e/2,2+e}(\Omega_T)$ with $e \in (0,1)$ satisfying*

$$\sup_{(t,x)\in\Omega_T} \sum_{k=1}^{2} |\nabla_x^{(k)} u(t,x)| < \infty$$

- *The coefficients of non-linear terms of* (4.2.1) *are locally Lipschitz in $(u, \nabla u)$ with Lipschitz constant that can have at most polynomial growth on u and ∇u, uniformly w.r.t. t, x*

Then, for every $\varepsilon > 0$, there exists a positive $K > 0$ that depends on u such that there exists $f \in \mathcal{N}$ that satisfies

$$J(f) \leq K\varepsilon$$

Theorem 4.2.1 leads us to the existence of Neural network approximation $\{f_n\}$ such that $J(f_n) \to 0$. With this $\{f_n\}$, we have the following result:

Theorem 4.2.2. *(Theorem 7.3 in [23])*
Assuming the boundedness of α, γ, Ω, and smoothness of $\alpha, \gamma, \partial\Omega$, the (4.2.1) *has a unique solution u and f_n converges to u, strongly in $L^s(\Omega_T)$ for every $s < 2$.*

Remark 4.2.1. *Through Theorem 4.2.1 to Theorem 4.2.2, we can use any positive measure to calculate the L^2-norm whenever the measure has support in $\overline{\Omega_T}$.*

4.2.2 Neural Network Approximation for optimal liquidation problem

Recall that our final aim is to solve (3.3.6):

$$\begin{cases} U_t + \frac{1}{2}\sigma^2(t,S)S^2U_{SS} - \lambda NS + \frac{1}{4\eta}U_N^2 - \frac{NU_N}{T-t} = 0, & \text{on } (0,T)\times\mathbb{R}^2, \\ U(T,S,N) = 0, \qquad (S,N)\in\mathbb{R}^2. \end{cases}$$

And, in case of GBM, which means $\sigma(t,S) = \sigma$ is constant, (3.3.3) gives us the solution

$$\begin{aligned} U(t,S,N) = &-\frac{1}{2}\lambda(T-t)NS \\ &+\frac{\lambda^2}{8\eta\sigma^6}S^2\left(e^{\sigma^2(T-t)} - 1 - \sigma^2(T-t) - \frac{1}{2}\sigma^4(T-t)^2\right) \end{aligned} \tag{4.2.4}$$

In order to use the results in Section 4.2.1 to solve the optimal liquidation problem, we need to fix some troubles. First, we need to restrict (S,N) in a bounded domain $\Omega\subset\mathbb{R}^2$. Theorem 4.2.1 and 4.2.2 are based on the bounded domain $\Omega\subset\mathbb{R}^2$ for x. The natural restriction of domain is setting $\Omega = [0, S_{max}]\times[0, N_{max}]$ for some fixed S_{max} and N_{max}. The very second trouble we face related to the first trouble is that the local volatility $\sigma(t,S)$ can be unbounded on the domain such as CEV model. Therefore, we will form $\Omega = [S_{min}, S_{max}]\times[0, N_{max}]$ so that $\sigma(t,S)$ is bounded on Ω_T to fix these troubles. As a result, we will form (3.3.6) as the following:

$$\begin{cases} U_t + \frac{1}{2}\sigma^2(t,S)S^2U_{SS} - \lambda NS + \frac{1}{4\eta}U_N^2 - \frac{NU_N}{T-t} = 0, & \text{on } \Omega_T, \\ U(T,S,N) = 0, \qquad \text{on } \Omega \end{cases} \tag{4.2.5}$$

where $\Omega = [S_{min}, S_{max}] \times [0, N_{max}]$, $\Omega_T = [0, T) \times \Omega$. At last, to approximate the solution U, the following Neural network approximation problem arises:

$$h_n \in \arg\min_{h \in \mathcal{N}_n} J(h) \tag{4.2.6}$$

where

$$J(h) = \left|\left| h_t + \frac{1}{2}\sigma^2(t,S)S^2 h_{SS} - \lambda NS + \frac{1}{4\eta}h_N^2 - \frac{Nh_N}{T-t} \right|\right|^2_{2,\Omega_T} + ||h(T,\cdot)||^2_{2,\Omega}$$

Here, to avoid the unboundedness of $\frac{1}{T-t}$, let us choose the probability measure $\mathbb{P}_1$ on Ω_T which has probability density function is proportional to $(T-t)^2$ and uniform in S and N, and uniform probability measures $\mathbb{P}_2$ on Ω. And Remark 4.2.1 allows us to choose these measures. In the next section, we will provide an algorithm for solving (4.2.6) and numerical results for GBM and CEV model.

4.3 Algorithm and result

4.3.1 Algorithm

In our thesis, we've chosen to optimize (4.2.6) by using the Mini-Batch Gradient Descent method with batch size $M_b = 10$ described in Section 4.1.2. Even though it's unstable as mention in Section 4.1.2 with small batch size, [22] shows that choosing proper learning rate α_k gives us the critical point in $\mathcal{N}_n$. Thus, we have chosen the Mini-Batch Gradient Descent method which avoids forming a mesh, so that the computation is performed rapidly. The optimization algorithm is as following:

1. Initialize $\theta_0^{(n)}$ a vector of parameters of $\mathcal{N}_n$
2. Generate uniformly M_b random points (t_1^m, S_1^m, N_1^m) from Ω_T, uniformly M_b random points (S_2^m, N_2^m) from Ω.

3. Calculate the squared L^2-norm $C(\theta_k^{(n)}, s_k)$ at the randomly sampled points $s_k = \{(t_1^m, S_1^m, N_1^m), (S_2^m, N_2^m)\}_{m=1}^{Mb}$ where $C(\theta_k^{(n)}, s_k)$ is the single cost function at randomly sampled point s_k

4. Take the descent step at s_k:

$$\theta_{k+1}^{(n)} = \theta_k^{(n)} - \alpha_k \nabla_\theta C(\theta_k^{(n)}, s_k)$$

5. Repeat (2)-(4) until $\left|\left|\nabla_\theta C(\theta_k^{(n)}, s_k)\right|\right|$ gets small enough

Here, the learning rate α_k satisfies:

$$\sum_{k=1}^{\infty} \alpha_k = \infty, \qquad \sum_{k=1}^{\infty} \alpha_k^2 < \infty$$

which gives the almost sure convergence of the iteration. And, the single cost function C is defined by:

$$\begin{aligned} C =& C_1 + C_2 \\ C_1 =& \frac{3}{T^2}\left((T-t)(u_t + \frac{1}{2}\sigma^2(t,S)S^2 u_{SS} - \lambda N S + \frac{u_N^2}{4\eta}) - N u_N\right)^2; \text{ on } \Omega_T \\ C_2 =& u^2(T,S,N); \text{ on } \Omega \end{aligned}$$

where C_1 is not same with the cost function in (4.2.6) because, in the calculation algorithm, we have chosen uniform distribution in Ω_T, so we need correction by multiplying $\frac{3}{T^2}(T-t)^2$.

4.3.2 Result for GBM

In this subsection, we present our result on conducting the Neural network approximation algorithm described in section 4.3.1 with the following given constants:

With these constants, we've conducted the Neural network approximation with $10, 50, 100, 200$ neurons. Figure 4.3 shows the error versus iterations. We have observed that the larger neurons, the faster convergence.

σ	0.15	η	0.01
S_0	100	λ	3×10^{-9}
T	1	S_{min}	10
N_0	100000	S_{max}	1000

Table 4.1: GBM model

It may seem that the error is large, but compared to the value of V_0, it is less than $10^{-5}\%$. Therefore, it is valid to approximate solution by Neural network approximation.

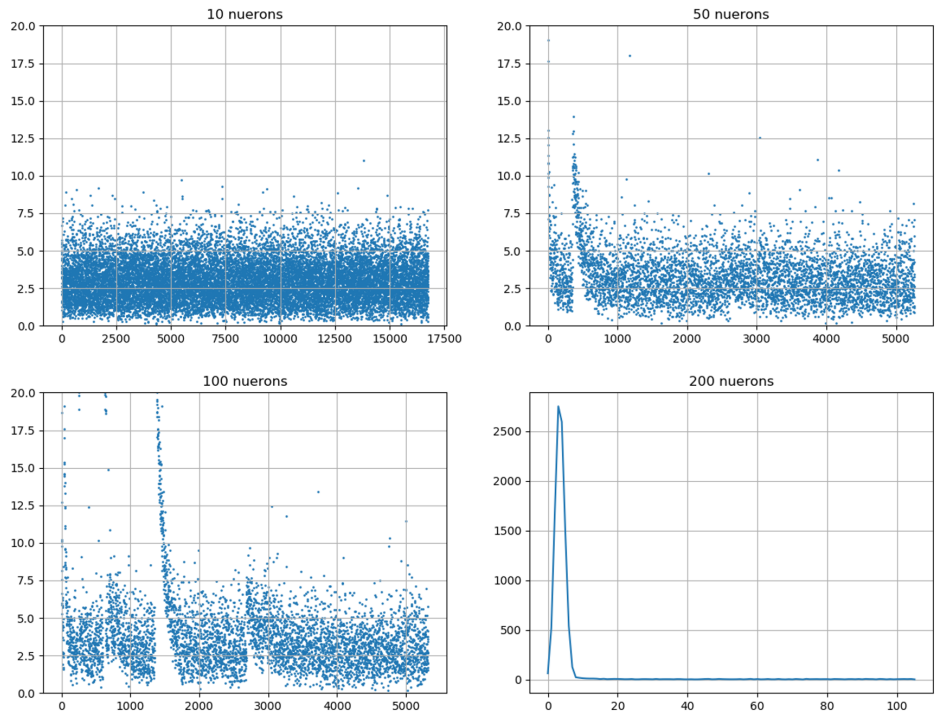

Figure 4.3: Errors versus iterations for GBM model

4.3.3 Result for CEV model

In this subsection, we present our result on conducting the Neural network approximation algorithm described in section 4.3.1 with the CEV model with parameters in Table 4.2.

With this CEV model, we've conducted the Neural network approximation with $10, 50, 100, 200$ neurons. Figure 4.4 shows the error versus iterations. We have observed very slow convergence. It is due to the large volatility near $S = S_{min}$. It may seem that the error is large, but compared to the value of V0, it is less than $10^{-5}\%$. Therefore, it is valid to approximate solution by Neural network approximation.

$\sigma(S)$	$0.15(S_0/S)^{0.5}$	η	0.01
S_0	100	λ	3×10^{-9}
T	1	S_{min}	10
N_0	100000	S_{max}	1000

Table 4.2: CEV model

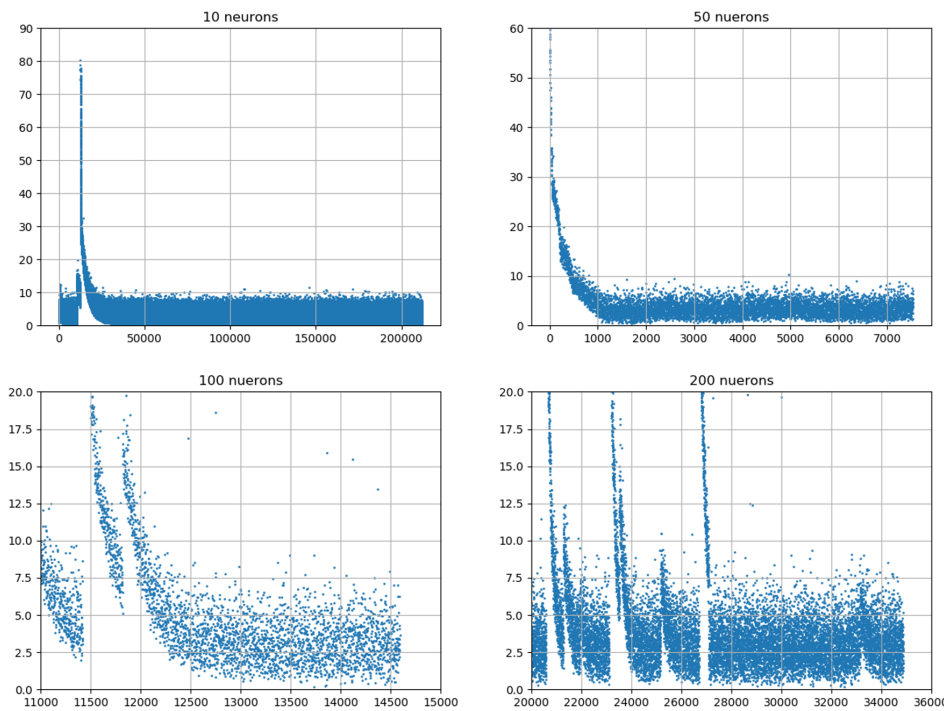

Figure 4.4: Errors versus iterations for CEV model

Chapter 5

Conclusion and future works

In this thesis, we covered three topics related to the volatility of asset price dynamics.

First, we have presented the roughness of implied volatility via the European option data associated to a stock. In previous works, the roughness of volatility process has been detected and a general property of rough volatility process driven by fractional Brownian motion was found. We examined the validity of the assumption that fractional Brownian motion drives the volatility process. Furthermore, we also used smoothing cubic spline to fulfill data between available data points. As a result, Figure 2.3 and 2.4 reveals the rough volatility path of JPMorgan Chase in 2012.

Second, we have introduced the optimal liquidation problem with linear effect of transaction. Under the local volatility model, the optimal liquidation strategy problem becomes a continuous-time Markov Decision Process problem. To find an adapted trading rate strategy, we restrict the trading rate to be a function of state, the pair of time, stock price, and remaining shares. We have called the strategy as a policy, and defined the optimal policy as the maximizer of sum of expected total cash flow and expected total VaR over the set of policies that liquidate all stocks at final time T. By dynamic programming, we obtained the corresponding PDE, which

is referred as the Hamilton-Jacoib-Bellman (HJB) equation. In previous works, an accurate solution is given for GBM, which means a constant volatility model. In this thesis, we extended the optimal liquidation problem to the local volatility model. Observing that the optimal value function V_0 satisfying the HJB equation for GBM without risk measure VaR also satisfies the HJB equation for local volatility without risk measure VaR, we suggested a solution V for the HJB equation for local volatility with risk measure VaR as the form of $V_0 + U$. Since V_0 satisfies the terminal condition of HJB equation, we have obtained another non-linear equation for U with bounded terminal condition. Therefore, we reformed the optimal liquidation problem into a partial differential equation with bounded condition. However, two issues occurred. First issue is to determine the bounded terminal condition. In this thesis, we set the bounded terminal condition 0 value. Second issue is that the obtained PDE is a non-linear PDE which is difficult to solve. These issues remain unsolved, and will remain in future works.

Finally, we studied a deep learning method to solve a partial differential equation, and applied it to solve the optimal liquidation problem. We studied deep learning methods such as Neural network approximation and Gradient Descent method. In contrast with usual additive approximation theory, the Neural network approximation uses compositions of non-linear functions. And naturally, the Neural network approximation leads us to an optimization problem. The Gradient Descent method is an optimizing method by moving the parameters to the direction of steepest difference. Previous works provide the validity of neural network approximation with Gradient Descent optimizing method for solving PDE on a bounded domain. As a result, we applied the Neural network approximation with Gradient Descent optimizing method to solve the PDE obtained in Section 4.2.2. However, during the optimization, there arises an issue. This issue is about the learning-rate. Using small batch size, the computation is very rapid, but it is still unstable. The issue concerning the learning rate will

be discussed in future works.

Bibliography

[1] Bennedsen, M., Lunde, A., Pakkanen, M. S.: *Decoupling the short- and long-term behavior of stochastic volatility.* Working Paper (2016)

[2] Bayer, C., Friz, P., Gatheral, J.: *Pricing under rough volatility.* Quantitative Finance (2016), Vol. 16, No. 6, 887–904.

[3] Fukasawa, M.: *Short-time at-the-money skew and rough fractional volatility.* Quantitative Finance (2017), Vol. 17, No. 2, 635–654

[4] Dupire, B.: *Pricing with a smile.* Risk Magazine (1994), Vol. 7, No. 1, 18–20

[5] Black, F., Scholes, M.: *The Pricing of Options and Corporate Liabilities.* Journal of Political Economy (1973), Vol. 81, No. 3, 637-654.

[6] Shreve, S.: *Risk-Neutral Pricing.* Stochastic Calculus for Finance II: Continuous-Time Models (2004), 209-261.

[7] Bertsimas, D., Lo, A.: *Optimal control of execution costs.* Journal of Financial Markets (2018), Vol. 1, Issue 1, 1-50

[8] Gatheral, J., Jaisson, T., Rosenbaum, M.: *Volatility is rough.* Quantitative Finance (2018), Vol. 18, No. 6, 933-949

[9] Mandelbrot, B., Ness, J.: *Fractional Brownian Motions, Fractional Noises and Applications.* SIAM Review (1968), Vol. 10, No. 4, 422–437

[10] Glasserman, P., He, P.: *Buy Rough, Sell Smooth.* Quantitative Finance (2020), Vol. 20, No. 3, 363–378

[11] Fukasawa, M.: *Asymptotic analysis for stochastic volatility: martingale expansion.* Finance and Stochastics (2011), 635–654

[12] Almgren, R., Chriss, N.: *Optimal execution of portfolio transactions.* Journal of Risk (2001), 5-40.

[13] Björk, T.: *Stochastic Optimal Control.* Arbitrage Theory in Continuous Time (2009), 282-293.

[14] Gatheral, J., Schied, A.: *Optimal Trade Execution under Geometric Brownian Motion in the Almgren and Chriss Framework* International Journal of Theoretical and Applied Finance (2011), 353-368.

[15] Gueant, O.: *The Almgren-Chriss framework.* The Financial Mathematics of Market Liquidity (2016), 39-64.

[16] Gatheral, J.: *No-dynamics-arbitrage and market impact* Quantitative Finance (2016), 749-759.

[17] Lagaris, I., Likas, A., Fotiadis, D.: *Artificial neural networks for solving ordinary and partial differential equations.* IEEE Transactions on Neural Networks (1998), Vol 9, No. 5, 987-1000.

[18] Lagaris, I., Likas, A., Papageorgiou, D.: *Neural-network methods for boundary value problems with irregular boundaries.* IEEE Transactions on Neural Networks (2000), Vol 11, No. 5, 1041-1049.

[19] Lee, HI.: *Neural Algorithm for Solving Differential Equations.* Journal of Computational Physics (1990), Vol 91, 110-131.

[20] Rudd, K.: *Solving Partial Differential Equations using Artificial Neural Networks.* Duke University (2013), PhD Thesis.

[21] Saltelli, A., Ratto, M., Andres, T., Campolongo, F., Cariboni, J., Gatelli, D., Saisana, M. and Tarantola, S.: *Introduction to sensitivity analysis.* Global sensitivity analysis. The Primer (2008), 1-51.

[22] Bertsekas, D., Tsitsiklis, J.: *Gradient convergence in gradient methods via errors.* SIAM Journal of Optimization (2000), Vol.10, No. 3, 627-642.

[23] Sirignano, J., Spiliopoulos, S.: *DGM: A deep learning algorithm for solving partial differential equations.* Journal of Computational Physics (2018), Vol. 375, 1139-1364.

국문초록

본 학위 논문에서는, 함축된 변동성을 이용하여 거친 변동성에 대한 분석, 그리고 국소 변동성 모델에서 최적의 유동화 전략과 최적의 유동화 전략을 구하기 위한 심층 학습 방법을 제시한다. 실제 금융 시장에서, 변동성의 역학은 특히 파생상품 시장에서 그 자체로 매우 중요한 주제이다. 따라서 파생상품에 대한 이해를 위해, 변동성의 메모리 효과를 나타내는 변동성 역학의 거친 정도를 고려할 필요가 있다. 그리고, 실제 포트폴리오 관리에서, 대량의 주식을 조심스럽게 거래해야할 필요가 있다. 따라서 유동화 전략에 대한 평가와 최적화를 연구하는 것은 필수적이다.

이를 달성하기 위해, 우리는 우선 변동성 역학의 거친 정도가 작은 브라운 운동에서 나온다고 생각하고, 옵션 데이터로부터 거친 정도를 뽑아냈다. 그리고 최적의 유동화 문제를 연구하기 위해, 우리는 먼저 국소 변동성 모델에서 최적 유동화 문제를 제시하였다. 나아가, 역학적 프로그래밍을 통해 우리는 이 문제에 해당하는 해석적으로 풀기 힘든 편미분방정식을 얻었다. 최종적으로 우리는 편미분방정식을 풀기 위한 심층 학습 방법을 공부하고, 유동화 전략을 최적화 한다.

주요어휘: 변동성, 함축된 변동성, 거친 변동성, 작은 브라운 운동, 변동성 왜곡, 기하 브라운 운동, 국소 변동성 모델, 최적 조작, HJB 방정식, 딥러닝, 신경망 네트워크, 기울기 하강 방법
학번: 2018-26732