HEC MONTRÉAL

Model Risk in Stock Liquidation Strategies

par

Ali Roshani Moghaddam

Professor Michèle Breton
HEC Montréal
Directrice de recherche

Sciences de la gestion
(Spécialisation Financial Engineering)

Mémoire présenté en vue de l'obtention
du grade de maîtrise ès sciences
(M. Sc.)

August 2021

Résumé

La négociation algorithmique est un système de négociation par ordinateur orienté vers l'optimisation de l'exécution d'importants ordres d'achat ou de vente de valeurs ou de dérivés. Ce système s'est rapidement répandu au sein des institutions financières et a attiré l'intérêt des chercheurs universitaires. Pour les mainteneurs de marché, l'enjeu principal est de déterminer une stratégie optimale, exécutable par un ordinateur, pour liquider (ou acheter) un grand nombre d'actions en un court laps de temps, tout en minimisant l'impact de ces transactions sur le prix du marché.

Les solutions théoriques de ce problème sont principalement basées sur des hypothèses quant au comportement du marché. Par conséquent, les stratégies proposées sont exposées au risque de modèle, c'est-à-dire qu'elles sont sensibles aux hypothèses et à l'estimation initiale des paramètres du modèle de marché.

Dans ce mémoire, nous étudions la sensibilité du prix de marché et des critères de performance de certaines de ces solutions analytiques à la valeur estimée des paramètres du modèle. Pour ce faire, nous évaluons la sensibilité des taux d'exécution, des prix de marché et des prix d'exercice à des erreurs d'estimation. Par la suite, nous utilisons la simulation de Monte Carlo pour analyser l'impact de telles erreurs d'estimation sur la fonction objectif du décideur.

Nos résultats qui montrent que les modèles usuels sont très sensibles à l'estimation de la volatilité des prix, ainsi qu'à celle du coefficient d'impact temporaire sur le cours des prix en bourse, alors qu'ils sont peu influencés par le coefficient d'impact permanent sur le cours des prix en bourse.

Pour remédier au risque de modèle, nous proposons une stratégie de négotiation intelligente basée sur une méthode d'apprentissage par renforcement, pour un environnement de marché spécifique. Nous entraînons des modèles d'apprentissage à partir de données simulées. Nous comparons par la suite les résultats de l'algorithme d'apprentissage à ceux correspondant à l'utilisation d'un taux d'exécution aléatoire et à la solution analytique étudiée précédemment. Nous comparons diverses approches en apprentissage par renforcement, dont l'interpolation linéaire et la technique de relecture et évaluons la stabilité de leur comportement.

Nos résultats montrent que l'algorithme d'apprentissage atteint un comportement très proche de la solution analytique, alors qu'il permet de s'ajuster aux erreurs d'estimation des paramètres, permettant ainsi d'atteindre une valeur de l'objectif plus élevée.

Mots-clés

Négotiation algorithmique, liquidation optimale, risque de modèle, analyse de sensibilité, apprentissage par renforcement

Abstract

Algorithmic trading is an automated trading strategy, which is growing rapidly across both of the financial institutions and universities. In this context, as a market maker, finding an optimum strategy for liquidating a large number of shares within a short period, and yet minimizing its impact on the market price, is a major concern. Theoretical solutions are mainly based on the models of the market environment. Therefore, these trading strategies are exposed to the model risk due to the sensitivity of these models to the initial parametric estimation.

In this thesis, we study the the sensitivity of the market price and performance criteria of these analytical solutions to parameter estimation. We formulate the difference in the trading rate, market and exercise price between the wrong and correct parameters estimations. Then, we use the Monte Carlo method to simulate the market environment, and investigate the impact of the wrong estimation on the utility function. We achieve results, which show that these models are significantly sensitive to the stock price volatility. Also, they are highly sensitive to the trading temporary price impact coefficient, while they are almost not sensitive to the trading permanent price impact coefficient estimation.

According to our results, both of the market price and performance criteria are significantly sensitive to the parameter estimation. To address this, we develop an intelligent trading strategy based on Reinforcement Learning (RL), for a specific market environment. We train Q-learning models on the simulated data. This data is computed using the Monte Carlo method with the same experimental setting as in the model evaluation chapter. For evaluating these models, we introduce two upper and lower baselines, which

are random trading and analytical solution strategies respectively. The results show a very close behavior of this trading strategy to the analytical solutions. Moreover, we use linear function approximation and experience replay technique, and find that they can help the Q-learning models in order to both reach better results, and have more stable behavior. Also, these methods have larger terminal cash comparing using the analytical solutions with the wrong model's parameters estimations. Therefore, these methods can be good alternatives for analytical solutions when we are not completely sure about the model parameters.

Keywords

Algorithmic Trading, Optimal Liquidation, Model Risk, Sensitivity Analysis, Reinforcement Learning

Contents

List of Tables

List of Figures

Acknowledgements

I am extremely grateful to my supervisor, Prof. Michele Breton for her invaluable guidance, endless support, and patience during my M.Sc. study. Additionally, I would like to express my gratitude to my parents. Without their support and encouragement in my whole life, it would be impossible for me to be here. Finally, I gratefully acknowledge the funding received towards the Fin-ML program.

Chapter 1

Introduction

1.1 Objectives

The objective of this research project is to investigate the impact of modeling assumptions in the context of algorithmic trading. More precisely, this thesis aims at

i) presenting the trading model and its solution considering specific assumptions for stock liquidation in a short time horizon;

ii) evaluating the sensitivity of performance criteria of various algorithmic trading models to parameter estimates;

iii) introducing an RL method for stock liquidation in a short time horizon.

1.2 Background

The development of artificial intelligence (AI) approaches for automated financial trading is a challenge that has captured the interest of researchers and analysts for decades (Cartea et al. [2015]). The automatic trading process is an online decision-making problem that can be described by two critical steps: *summarizing the market condition* and *executing*

optimal actions. For many years, algorithmic trading suffered from various challenges due to the difficulty of characterizing evolving market conditions in real time and the complexity of determining optimal decisions in the trading environment.

Advances in machine learning, particularly in deep learning and Deep Reinforcement Learning (DRL), gave rise to numerous novel proposals and architectures, including namely end-to-end approaches to algorithmic trading (Deng et al. [2016]). In this context, end-to-end refers to the direct mapping of high-dimensional raw market and environment observations into optimal decisions in real-time. As data-driven agents, many such algorithms rely on sources of data that are either externally or collectively maintained (e.g., sentiment analysis from Twitter feeds (Kaur [2017]).

However, recent results in AI safety and security (see Behzadan and Munir [2017]) show that DRL algorithms are vulnerable to perturbations in the input data at both the training and the inference phases of their deployment. Evidence to that effect has been provided in various settings, such as video games (Huang et al. [2017]), robots (Clark et al. [2018]), autonomous navigation (Behzadan and Munir [2019]), and cybersecurity (Han et al. [2018]).

In recent years, there has been a significant increase in regulatory activity as a result of the industry movement towards algorithmic trading (see Mattli [2018] for a survey). In Authority [2018], the UK Financial Conduct Authority (FCA) notes that: *"In the absence of appropriate systems and controls, the increased speed and complexity of financial markets can turn otherwise manageable errors into extreme events with potentially widespread implications. As a result, algorithmic trading continues to be an area of focus for the FCA and other regulators across the globe."* While the growing interest in the adoption of AI for algorithmic trading is well-justified by its success in other fields, there seems to be little consensus on how such algorithmic trading models should be validated.

1.3 Outline

The first part of the project consists of assessing the sensitivity of strategies and of performance criteria of algorithmic trading models to parameter estimates (e.g., volatility, liquidity, impact of trading volume on asset prices, etc.). We will use stress testing and scenario testing methods to evaluate trading models. We will first evaluate the impact of estimation errors on the trading strategy using analytical or numerical solutions of optimal trading models. Monte-Carlo simulation will then be used to compare asset price and performance criterion trajectories under the optimal strategy and the strategy implied by the parameter estimation.

In the second part of the project, we will conduct experiments using RL algorithms to adjust parameter values to market observations while trading in real time. We will investigate various monitoring and adaptation strategies to mitigate model risk in algorithmic trading. Again, simulations (under the theoretical model) will be used to assess the reactivity and the adaptability of reinforcement learning approaches for the determination of the optimal trading frequency over a given time horizon when parameter values are uncertain.

1.4 Contribution

From the model risk governance perspective, there are important differences between traditional and algorithmic trading model risk. One salient feature is that the number of model components involved in algorithmic trading can be quite large, so that the composition of individually negligible estimation imprecisions may result in significant system errors. The first part of the project will provide an illustration of the level of risk involved in algorithmic trading under various assumptions. For this purpose, we simulate the impact of the variation in our parameters' estimation on a simple optimal liquidation problem. This could help to have an insight about what could happen in more complex algorithmic trading models.

A second contribution of this research project is to analyze the adaptive nature of data-driven algorithmic trading, such as DRL or RL-based systems where model components (self) learn and retrain. It is important to note that, unlike other areas of quantitative modelling in finance, there is not much information publicly available on the performance of algorithmic trading systems due to competition considerations.

Model performance and model risk are essential parts of any risk governance framework. Methods allowing for real-time robustness testing and safety verification are crucial in the context of algorithmic trading, given the number of components, the complexity and the speed of these systems. This research project can be considered as a step toward the objective of developing a risk analysis framework in the context of algorithmic trading.

1.5 Overview of Results

According to our research, the analytical solutions for the large order optimal liquidation within a short period problem are extremely sensitive to temporary impact coefficient estimation. Also, the model for the risk averse agent is highly sensitive to the stock price volatility estimation. However, these models are not being impacted by the permanent impact coefficient estimation.

In the last part of the thesis, we use three Q-learning methods to develop an intelligent agent, which finds the optimal trading strategy without a need to estimate the parameters. The results show that the behavior of these strategies is similar to the analytical solutions. Moreover, these methods have better performance comparing using analytical solutions with wrong assumptions, and can be good alternatives for them when we are not sure about the model's parameters values.

1.6 Organization of the Thesis

Chapter 2 is a literature review covering the previous researches close to the aim of this project. In Chapter 3, we discuss some key topics related to the financial market, and introduce our basic model and its solutions considering specific assumptions for large order liquidation within a short period problem.

The aim of Chapter 4 is to evaluate the model risk of the mentioned analytical solutions. We discuss the simulation procedure. Then, the analytical and numerical sensitivity tests are presented, and in the final part, we compare their performance in different market scenarios.

The focus of Chapter 5 is to introduce a machine, which takes the market data, tries to adapt to the market environment, and finds the best trading strategy. We firstly introduce our RL model and its specifications. Then, we present the learning, testing, and evaluation process, and discuss the results.

The last chapter is a short conclusion.

Chapter 2

Literature review

Recent developments in computer technologies, such as fast data transferring and high-speed processors, construct the new market environment and market infrastructure, which in turn, result in many alterations in the financial industry. Due to this, since the 1970s, most of the US equity market shifted to the electronic structure. As a result, the financial institutions tried to align their trading strategies with the market evolution. Accordingly, they parameterized the stock market behavior to have the ability to simulate its trend and capture its reaction to many different scenarios. Additionally, they used a system named "Algorithmic trading" to automate the whole or part of the trading process, which performs well in the rapid-changing stock market environment (for more information about the previous changes in financial market see Nuti et al. [2011]).

Algorithmic trading is a computational method to specify the trading decisions based on mathematical and statistical rules. This form of trading has many advantages, including lower commissions, transparency, minimization of information loss, and reduction in transaction costs. However, as discussed in Kissell [2013] and Li [2017], there are serious challenges in implementing such trading strategy in the real stock market. For example, there is an inadequate adaptability in measuring environmental uncertainty. This can weaken the algorithmic trading performance for investors' needs. Furthermore, there is a lack of research in algorithmic trading performance evaluation. Therefore, measuring

the performance of any new algorithmic trading strategy and evaluating its risk is another concern in its implementation process (for more information about algorithmic trading, see Théate and Ernst [2021]).

The earliest studies in algorithmic trading were mostly based on mathematical, statistical, and economical tools. The main part of these works aimed to find the best optimal execution methods to fulfill the clients' trading goals. These methods were based on the initial assumptions about the market environment such as price trajectories, the active market participants, and the existing risks. For instance, Chan [2021] and Narang [2013] used market trend obeying and mean reversion methods in order to develop their algorithmic trading strategies.

Since earlier studies, there has been further development due to the growth in market data and information availability, development in computer processors, and evolution in artificial intelligence (AI), mostly in the years following the 2000s. This has contributed to developing useful tools in computation, optimization, and control, which facilitate development process of many automated financial trading approaches. In this context, new research in machine learning (ML) has been developed to drive the automated system for trading. Machine learning is an application of AI, which allows the machines to learn from data and experience with no explicit human interaction. Due to the compatibility of ML for mapping high-dimensional raw input data to the desired output, this method helps to find an optimal decision according to the market environment inputs behavior (for more information about using ML in algorithmic trading see Deng et al. [2016], and Huang et al. [2019]).

With regards to these developments, RL algorithms, which are one type of ML, aim to simultaneously forecast the market dynamics, and decide the trading rates at any time. Moreover, in most of the researches, deep reinforcement learning methods are used to consider the experienced information of the algorithm during the trading horizon. For instance, Arévalo et al. [2016] used deep neural networks (DNN) to find the best execution decisions according to the multi-dimensional inputs; Bao et al. [2017] deployed long short-term memory (LSTM) neural network for finding the best trading strategy (to

consider the dependency of the market dynamics in sequential time steps); Paiva et al. [2019] derived an automated trading method using support vector machine (SVM) for forecasting the investment return and mean-variance method for portfolio construction; Deng et al. [2016] presented a fuzzy recurrent DL method in order to find an optimum trading rate, which minimizes the uncertainty risk in stock price prediction by using the fuzzy learning technique; Carapuço et al. [2018] used the deep Q-learning method, and Dempster and Leemans [2006] developed an adaptive RL method for finding the optimum trading decisions in the foreign exchange market; Moody and Saffell [2001] introduced a recurrent reinforcement learning (RRL) method to develop a trading strategy, which does not require forecasting model construction.

High-frequency trading is another type of trading, which was developed after the start of the 21st century (Duhigg [2009]). The importance of this trading strategy became more crucial as the trading execution time decreased from several seconds into microseconds in the last two decades (Klepacki et al. [2015]). Therefore, RL and specifically the Q-learning algorithms are also used for trading execution in a high-frequency environment. A good case in point is the research of Spooner et al. [2018] about simulating a limit order book market with high-frequency orders, using a temporal-difference reinforcement learning algorithm to find the best trading policy when an agent wants to liquidate its inventory in a short period. Vyetrenko and Xu [2019] deployed a risk-sensitive RL method to learn the market micro-structure in a simulated market which considers the impact of the trading on the market price, and then finds the best trading strategy. Ning et al. [2018] and Xiong et al. [2018] used a deep Q-learning method to find the best trading policy.

As a classical problem for market makers, finding an optimum strategy to liquidate or acquire a large number of stock shares in a short period is a specific concern (for more information see Cartea et al. [2015]). This problem has some analytical solutions, which are mostly based on dynamic programming (for instance see Ross [2014]). Almgren and Chriss [2001], Cartea et al. [2015], Zhang et al. [2015], and Pemy et al. [2008] used this method to solve the large stock liquidation problem. However, these methods are highly dependent on their strict assumptions about the trading impact on the stock market, market

price volatility, and other features of the market environment. Hence, many researchers such as Cartea et al. [2017] believe that these trading strategies are highly sensitive to any change in the market environment.

In this context, reinforcement learning methods are used to address this liquidation problem too. For example, Nevmyvaka et al. [2006] considered a market with a single big market maker participant and used an RL method to tackle this problem. They trained their RL model on millisecond time-scale market data from NASDAQ for a period of 1.5 years and showed the efficiency of their model performance in the market environment with a single big agent. Also, Bansal et al. [2017], Tampuu et al. [2017], Bao and Liu [2019], and Yang et al. [2018] investigated the impact of multiple big active agents in the same market, and found the optimum execution policy (buy or sell and its quantity) when multiple financial institutions are active in the market.

While the growing interest in the adoption of AI for algorithmic trading is well-justified by its success in other fields, there seems to be little consensus on how such algorithmic trading models should be validated. Within this framework, recent results in AI safety and security (see Behzadan and Munir [2017]) demonstrated that DRL algorithms are vulnerable to perturbations in the input data (called adversarial actions in the AI security domain) at both training and inference phases of their deployment. This discovery is further verified in various settings such as video games (Huang et al. [2017]), robots (Clark et al. [2018]), autonomous navigation (Behzadan and Munir [2019]), and cybersecurity (Han et al. [2018]). Moreover, Sornette and von der Becke [2011] and Moosa [2015] investigated the possible market bubbles and crashes connected to high-frequency trading. Glantz and Kissell [2013] reviewed the risks and evaluation metrics of algorithmic trading. In other works, the adverse impact of automated trading strategies on the stock market price was evaluated by Kirilenko and Lo [2013], and Arnoldi [2016] looked over the impact of misleading on execution strategies on the market behavior.

In this circumstance, many regulations have been enforced on financial institutions regarding algorithmic and high-frequency trading (see Mattli [2018] for a survey) to have a better monitoring ability and controlling their performance. Linton and Mahmoodzadeh

[2018] indicated that these types of trading strategies may result in unreliability and even more volatility in the market behavior, which could be addressed by the correct regulations in the short-term. They demonstrated that abusive trading practices like front running should be forbidden in order to prevent a market crash. However, many researchers such as Fletcher [2021] and Aitken et al. [2013] believe that these regulations did not consider the long-term impact of these trading strategies on the market. Therefore, there is an important concern for the regulators to understand the impact of these computer-based methods on the market quality to have better legislation.

Chapter 3

Optimal Execution Model

One of the typical problem forced by market makers is to find the best trading strategy to liquidate a large number of stock shares within a short period. Moreover, as a market maker, these agents have to minimize their impact on the market price during the trading period. In this chapter, we overview how the market environment behaves, introduce the appropriate model for our problem, and present the analytical solutions considering specific assumptions.

3.1 Market environment

In general, a limit market order is an environment where stock buyers and sellers can place their orders simultaneously. These orders will be executed according to time and price priority. In this context, the highest buy (bid) or sell (ask) prices are being considered as the market price. Therefore, the evolution of the market price depends on how the market participants behave.

The common way for modeling the market in a short time horizon is using a Brownian Motion (BM) process. However, any participant can also impact the price by its own actions. In this project, we assume that each participant can have permanent and temporary impacts on the price. While the permanent impact influences the market price, the temporary impact only changes the agent's execution price at the current step. Therefore,

we can consider the following dynamic for the market and execution prices:

- S_t: is the market price for the share that the agent aims to liquidate and

$$dS_t = -g(v_t)dt + \sigma dW_t, \tag{3.1}$$

  where

  - v_t: is the agent's trading rate.
  - σ: is the stock market price volatility.
  - W_t: is a standard Brownian motion.
  - $g(v_t)$: represents the permanent price impact of the agent's trading on the stock price.

- $\widehat{S_t}$: is the stock price at which the agent can liquidate v_t shares at time t (execution price) and:

$$\widehat{S_t} = S_t - f(v_t), \tag{3.2}$$

  where

  - $f(v_t)$: represents the temporary price impact of the agent's trading on the execution price.

The gained money rate from each decision is equal to:

$$r(S_t, v_t) = v_t(S_t - f(v_t)) = v_t\widehat{S_t}, \tag{3.3}$$

There are various models about how the market and execution prices are impacted by the agent's behavior. Since the agents can only choose their trading rate at each time step, the g and f functions are solely dependent on the trading rate variable.

3.2 Model

As discussed previously, in the optimal liquidation problem, we want to liquidate a large number of stock within a short time with the maximum price. Therefore, our problem is a sequential decision-making procedure, which follows the Markov process, and at each time step, we look at the market price and our remaining inventory, and decide how much to trade. Since according to Equations 3.1 and 3.2, our actions have an impact on the price, we can consider our problem as a dynamic optimization problem, which can be efficiently solved by dynamic programming (DP). For this purpose, we can formulate this problem in a dynamic programming structure (for more detailed information about DP and Markov process, see Appendix A) in discrete time with the following specifications:

- **Steps** are the dates denoted by t_i, where $i \in \{0,1,...,I\}$, the time step between two successive dates is assumed to be constant and equal to δ, and t_I is the horizon T.
- **State** variables are the stock market price (s), the cumulative cash, which is gained by the previous transactions (m), and the remaining inventory of the agent (q).
- **Decision** variable is the number of the shares, which the agent decides to liquidate (v)
- **Value function** (z) is the maximum expected utility of the cumulative cash in the end of the trading horizon T at date t if the number of remaining shares is q, the gained cash is m, and price is s.

$$z_i(s,m,q) = \max_{0\leq v\leq q} \mathbb{E}\{z_{i+1}(F(s,v), m + r(s,v) - h(q), q - v\delta)\} \quad \text{for } i \in \{0,1,...,I-1\}, \tag{3.4}$$

$$z_I(s,m,q) = U(m + r'(s,q)),$$
$$s = s_0, \quad m = 0, \quad q = q_0 \quad \text{at} \quad i = 0,$$

where

- U is the utility function of the cash, which is selected based on the risk-aversion of the agent:
  - Risk-neutral agent:
    $$U(y) = y \tag{3.5}$$
  - Risk-averse agent:
    $$U(y) = -e^{-\gamma y}, \tag{3.6}$$
    where γ parameter is the risk aversion coefficient.
- F function is the stochastic process of the stock market price, which depends on the agents' assumptions about the stock market environment parameters, and their trading impact on the market price. So:
  $$F(s,v) = s - g(v)\delta + \sigma W_\delta, \tag{3.7}$$
- r function is the transacted money, which the agents gain at each step based on their decision, and their assumptions about their impact on the execution price. So, :
  $$r(s,v) = v\delta(s - f(v)) = v\delta\widehat{S}, \tag{3.8}$$
- r' function is the transacted money for liquidating the remained inventory at the terminal step, based on their assumptions about their impact on the execution price. So:
  $$r'(s,q) = q(s - f'(q)), \tag{3.9}$$
  where f' represents the terminal temporary price impact of the agent's trading on the execution price.
- h function is the inventory holding penalty, which is selected based on the risk-aversion of the agent. Cartea et al. [2013] showed that this term is equivalent to consider that our agent is ambiguity-averse, and prefers known risks over the unknown risks.

– Risk-neutral agent:

$$h(q) = \phi \delta q^2, \tag{3.10}$$

– Risk-averse agent:

$$h(q) = 0, \tag{3.11}$$

where ϕ is the agent's urgency for execution coefficient.

This model has a continuous version. In this version we want to maximize the following function:

$$z_t(s,m,q) = \max_v [m + \int_t^T r(s_u, v_u)du - h(q)] \quad \text{for } 0 \leq t < T, \tag{3.12}$$

$$z_T(s,m,q) = U(m + r'(s,q)) \tag{3.13}$$

$$\dot{m}(t) = r(s_t, v_t) - \dot{h}(q_t),$$

$$dS_t = -g(v_t)dt + \sigma dW_t,$$

$$\dot{q}(t) = -v_t,$$

$$s = s_0, \quad m = 0, \quad q = q_0 \quad \text{at} \quad t = 0,$$

where

- U is the utility function of the cash, which is selected based on the risk-aversion of the agent according to Equations 3.5 and 3.6.
- r function is the transacted money rate, which the agents gain at each time based on their decision, and their assumptions about their impact on the execution price. So, :

$$r(s_t, v_t) = v_t(s_t - f(v_t)) = v_t \widehat{S}_t, \tag{3.14}$$

- r' function is the transacted money for liquidating the remained inventory at the terminal time, based on their assumptions about their impact on the execution price. So:

$$r'(s,q) = q(s - f'(q)), \tag{3.15}$$

where f' represents the terminal temporary price impact of the agent's trading on the execution price.

- h function is the inventory holding penalty, which is selected based on the risk-aversion of the agent:
  - Risk-neutral agent:

$$h(q) = \phi \int_t^T q_u^2 du \rightarrow \dot{h}(q_t) = \phi q_t^2, \tag{3.16}$$

  - Risk-averse agent:

$$h(q) = 0 \rightarrow \dot{h}(q_t) = 0, \tag{3.17}$$

In the following section, we introduce the assumptions, which allow us to solve this problem analytically. Then, we discuss that how the specific agent requirements can impact this general solution.

3.3 Solution

When the optimal liquidation model and the market environment characteristics have been defined, the analytical solution can be introduced. In this project, we introduce the analytical solutions based on the book written by Cartea et al. [2015]. This solution considers the following assumptions about the model we have already introduced:

- Both the temporary and permanent trading impact on the stock price are being considered as a linear function of trading rate, so:

$$f(v) = kv; \quad g(v) = bv \quad \text{for finite constants} \quad k \geq 0, \quad b \geq 0, \tag{3.18}$$

  where k and b are the temporary and permanent trading impact coefficients, respectively.

- The stock price volatility (σ) is constant during the trading horizon.

- According to the our model specification, at the terminal step, our trading decision is equal to the remained inventory. Therefore, all of the initial inventory will be liquidated within the trading period.

- If a terminal inventory penalty is being considered, we assume that the temporary trading impact coefficient for the terminal step is equal to $\alpha > k$. The α parameter is named terminal liquidation penalty coefficient.So:

$$f'(q) = \alpha q \quad \text{for finite constants} \quad 0 \leq k < \alpha, \tag{3.19}$$

For the continuous format of our model,the solution, which Cartea et al. [2015] presented is based on our optimal liquidation model, market environment characteristics, and the assumptions mentioned in the previous section. Therefore, the dynamics of optimum trading rate and the agent's remaining inventory are:

$$v_t^* = a_1\left(\frac{a_2 e^{a_1(T-t)} + e^{-a_1(T-t)}}{a_2 e^{a_1(T-t)} - e^{-a_1(T-t)}}\right) Q_t^{v^*} \quad \text{for all} \quad 0 \leq t < T \quad \text{and} \quad v_T^* = Q_T^{v^*} \tag{3.20}$$

$$Q_t^{v^*} = \left(\frac{a_2 e^{a_1(T-t)} - e^{-a_1(T-t)}}{a_2 e^{a_1(T)} - e^{-a_1(T)}}\right) Q_0 \quad \text{for all} \quad 0 \leq t \leq T \quad \text{and} \quad Q_{T+1}^{v^*} = 0 \tag{3.21}$$

Where a_1 and a_2 values depend on our assumptions about the order volume and risk-aversion level of the agent.

3.3.1 Assumptions

The agents' assumptions are based on the stock market environment and their client's objectives. In this regard, we categorize the assumptions based on two aspects. Firstly, we have two versions of the model for the decision-makers who are risk-averse ($U(y) = -e^{-\gamma y}$) or not ($U(y) = y$). Additionally, we assumed two different scenarios for the agents who are big players (an agent who has a large trading rate comparing to the total trading volume in the market, and as a result, the agent's trading impact on the market price is important, so $b > 0$) or small players ($b = 0$).

Scenario 1: Risk-neutral and small player agent

In this type of assumption, the clients are risk-neutral, and their orders are small comparing to the total volume of the market orders for other participants in the market. Therefore, we assume that the orders only have a temporary impact on the stock execution price ($b = 0$). In other words, the trading does not affect the market price; however, it decreases the execution price. For this agent, we neglect the terminal inventory penalty. Hence, the optimum trading strategy for this model is to liquidate the inventory with a constant rate of initial inventory over the number of trading time steps. So:

$$v_i^* = \frac{Q_0^{v^*}}{N}; \quad Q_i^{v^*} = \frac{N-i}{N} Q_0; \quad 0 \leq i \leq N \tag{3.22}$$

Scenario 2: Risk-neutral and big player agent

In this specific situation, our decision-maker is risk-neutral, but because of the large trading rates, the trading impact on the market price is important. Therefore, we assume that not only our trading activities have a temporary impact on the execution price, but also there is a permanent impact on the market price too. Also, in order to prevent the large remained terminal inventory, we assume a terminal inventory penalty for this agent. As a result, the trading strategy tries to liquidate the inventory sooner. Considering the aforementioned assumptions, the a_1 and a_2 parameters in the general solution are equal to:

$$a_1 = \sqrt{\frac{\phi}{k}} \quad \text{and} \quad a_2 = \frac{\alpha - \frac{1}{2}b + \sqrt{k\phi}}{\alpha - \frac{1}{2}b - \sqrt{k\phi}} \tag{3.23}$$

It should be mention that, if we remove the inventory holding penalty ($\phi = 0$), then a_1 and a_2 coefficients are equal to 0 and 1, respectively. Therefore, the optimal strategy is to wait until the end of the trading horizon, and liquidate all the inventory at the terminal step.

Scenario 3: Risk-averse agent

Since this execution problem is for a short time horizon, the money discount factor is not being considered. Therefore, the previous solutions for the previous assumptions are not completely risk-neutral. The risk-averse clients prefer to liquidate their orders sooner. Since keeping the orders for the later time steps, may result in selling them with a very low price (if any market price jump happens). Therefore, in this specific situation, we use an exponential utility function in order to discount the cash values in the later steps. However, all the other assumptions are similar to the previous type (for the small player we assume that the trading permanent impact is zero). Considering the aforementioned assumptions, the a_1 and a_2 parameters in the general solution are equal to:

$$a_1 = \sqrt{\frac{\gamma\sigma^2}{2k}} \quad \text{and} \quad a_2 = \frac{\alpha - \frac{1}{2}b + \sqrt{\frac{1}{2}k\gamma\sigma^2}}{\alpha - \frac{1}{2}b - \sqrt{\frac{1}{2}k\gamma\sigma^2}} \tag{3.24}$$

3.4 AI-based trading strategy

Reinforcement learning is an effective method for algorithmic trading, which has a procedure to learn the market environment, and find an optimum trading strategy. Since the analytical solutions for the specific market situation present the global optimums, in general, these solutions are better than the RL-based methods. However, since the RL-based methods can recognize any change in the market situation (which makes it different from the initial assumptions about the market environment), and make their decisions compatible to these new situations, they might have better (more robust) behavior in unpredictable market changes. In Chapter 5, we introduce an intelligent agent, which finds the optimum trading strategy for the short-term large-stock liquidation problem. For this purpose, we develop an RL model with two updated versions, which learn the market environment, and choose the best trading strategy. In the last step, we compare these models' behavior with our baseline cases to show their advantages.

Chapter 4

Model Risk evaluation

4.1 Model risk

Model risk is a basic type of risk that happens when our decisions and results are based on non-accurate, wrong or inappropriate models. In other words, since most of the models are an approximation of the real environment, if the model cannot perfectly show the behavior of the environment, then all the decisions and accordingly the results, are exposed to the model risk. This type of risk can be categorized into three main groups, including model misappropriation, model misspecification, and model misexecution. Model appropriateness is the capability of the model for describing the real environment. Therefore, the first step is to choose the right model, which describes the market behavior. Model specification mentions to the correctness of the model features and characteristics. As a result, the second step is to estimate the model parameters precisely. Model execution refers to the proper implementation strategy of the model. Therefore, the third step is finding the best strategy to execute the model.

In this chapter we discuss the model misspecification risk of the optimal solutions based on different assumptions. In this context, we analyze the impact of wrong models' parameters estimation on the trading decisions and performance criteria. For this purpose, firstly, we introduce our base case and experimental setting, based on the market environ-

ment, trading model, its assumptions and solution, which were discussed in Chapter 3.

4.2 Base Case and Experimental Setting

The next step for evaluating the optimal execution models is introducing the market environment parameters, stock specifications, and the agent's requirements for the sensitivity tests. In this regard, as a base case, we use the same example as Cartea et al. [2015] considered for the stock (SMH on October 1, 2013) with its market parameters, which are shown in the Table 4.1.

Parameter	Unit	Value
Initial inventory (Q_0)	share	20,000
Initial stock price (S_0)	\$	40.3
Stock price volatility (σ)	%	0.025
Trading permanent impact coefficient (b)	\$/share	0.00007
Trading temporary impact coefficient (k)	\$.sec/share	0.00035
Terminal liquidation penalty coefficient (α)	\$/share	0.01
Agent's urgency for execution coefficient (ϕ)	\$/share.sec	0.0005
Risk aversion coefficient (γ)	1/\$.share.sec	1.5
Number of trials		10,000
Number of discrete intervals		200

Table 4.1: Experimental setting parameters for the market model of SMH on October 1, 2013

It should be noted that the time unit is second. Also stock volatility is defined in the percentage of the initial price in the unit time interval. Having the trading permanent impact coefficient equal to $b = 0.00007\$/share$ means that, if we liquidate 100 shares of SMH at one time step with an initial price of 40.3\$, and we neglect the stochastic trend of the stock market price, the market price changes to $40.3 - 100 * 0.00007 = 40.293\$$.

Since all the execution solutions are defined in the continuous-time format, the first step for simulating the model is to approximate them in the discrete-time horizon intervals (to be programmable in computer-based software). In this context, we assumed that the time horizon is divided into N equal time intervals. To compute the discrete dynamics of

the remaining inventory and the trading rate, at each time step the inventory is computed according to Equation 3.15. Therefore, the trading rate is equal to the difference in the remaining inventory in two successive time steps.

Therefore, the remaining inventory is equal to

$$Q_i^* = (\frac{a_2 e^{a_1(\frac{N-i}{N})T} - e^{-a_1(\frac{N-i}{N})T}}{a_2 e^{a_1(T)} - e^{-a_1(T)}})Q_0 \quad \text{for all} \quad 0 \leq i \leq N; \tag{4.1}$$

Hence, the trading rate in the discrete format is equal to

$$v_i^*(\frac{T}{N}) = Q_i^{v^*} - Q_{i+1}^{v^*} \rightarrow v_i^* = \frac{Q_i^{v^*} - Q_{i+1}^{v^*}}{\frac{T}{N}} \quad \text{for all} \quad 0 \leq i < N, \tag{4.2}$$

It should be noted that the trading rate at the terminal time step is equal to the remaining inventory:

$$v_N^* = \frac{Q_N^{v^*}}{\frac{T}{N}} \quad Q_{N+1}^{v^*} = 0 \tag{4.3}$$

For simulating the stock market and execution price trajectories within the liquidation time horizon, the Monte Carlo method is used. Therefore, firstly, according to the execution strategy and the model setting, the trading rate and remaining inventory are computed at each time step. Then, for $10,000$ of trajectories (trials) and 200 time steps, stock market and execution prices are computed by:

$$S_{i+1} = S_i - bv_i^*(\frac{T}{N}) + \sigma W(\frac{T}{N})$$
$$\rightarrow S_{i+1} = S_i - b(Q_i^{v^*} - Q_{i+1}^{v^*}) + \sigma W(\frac{T}{N}) \quad \text{for all} \quad 0 \leq i \leq N$$

$$\widehat{S_i} = S_i - kv_i \quad \text{for all} \quad 0 \leq i < N \tag{4.4}$$

It should be mentioned that the specific solution for the first scenario, which was mentioned in Section 3.3, is not sensitive to any initial parameters estimation. Since, regardless of our assumptions about these values, the optimal trading rate is constant and equal to the initial inventory over the number of trading steps.

For assessing the sensitivity of the model to each parameter in the following section, three low and three high values are being used. To be more precise, according to the model setting, if the actual parameter is assumed to be equal to par_{market}, three low and high parameters values are assumed to be equal to:

$$\text{Low values:} \quad \frac{par_{agent}}{par_{market}} \in \{0.125, 0.250, 0.500\} \tag{4.5}$$

$$\text{High values:} \quad \frac{par_{agent}}{par_{market}} \in \{2.00, 4.00, 8.00\} \tag{4.6}$$

4.3 Model risk for parameters estimation

Now, we can assess the model risk for parameters estimation in our model's specific solutions for different assumptions.

4.3.1 Model parameters estimation sensitivity test

According to our market model and its trading optimum strategy, there are three main estimations including stock price volatility, trading temporary price impact coefficient and trading permanent price impact coefficient. In the following parts, the sensitivity of the model performance criteria and market behavior to these parameters estimation are evaluated.

Let's assume a wrong value for any of the mentioned parameters. All of the a_1', a_2', v', Q', S' and $\widehat{S}'$ values are computed for those wrong parametric estimation. Therefore, the difference between the market and execution prices, when we assume the correct or wrong estimations is (detailed computation is in the Appendix B):

$$S_t - S_t' = b(Q_t^{v^*} - Q_t'), \quad 0 \le t \le T$$
$$\text{and} \quad \widehat{S}_t - \widehat{S}_t' = b(Q_t^{v^*} - Q_t') - k(v_t^* - v_t'), \quad 0 \le t < T$$

Therefore, the parameters misspecification can result in having higher trading rates, and as a result lower remained inventory at the initial steps. Consequently, the market price during the trading horizon decreases.

It should be noted, regardless of the agent's assumption for these parameters, the real stock market price after the trading horizon does not change. Since at the terminal time, all the remaining inventory will be liquidated, there is no market price difference between the wrong or correct assumptions for these parameters after the liquidation horizon:

$$v_T^* = Q_T^{v^*}, \quad v_T' = Q_T' \quad \rightarrow S_{T+1} - S'_{T+1} = 0$$

Furthermore, at the terminal time step, the execution price is impacted by terminal inventory penalty coefficient (instead of the trading temporary price impact coefficient). Hence, the difference between the execution price for these two assumptions in the terminal time step is:

$$\widehat{S_T} - \widehat{S'_T} = (b - \alpha)(Q_T^{v^*} - Q'_T)$$

Therefore, we have all the formulation to start evaluating the models:

Stock price volatility (σ)

According to the trading dynamic and optimum execution solution formulation, which were mentioned in Section 3.3, the stock price volatility (σ) has an impact on the stock market price and the optimum trading rate for the risk-averse and big or small player agent (stock price volatility has an impact on both of a_1 and a_2 parameters, which are used in the trading rate computation).

The parameters setting for sensitivity test, which were discussed in previous section, can help us to investigate the impact of extreme misspecification without having too many samples. Therefore, for each of the two models we have:

Scenario 2

Since our assumption for the stock price volatility has no impact on the trading rate, the market price and the model's performance criteria will not be impacted.

Scenario 3

Considering the model setting, in Figure 4.1, we can see the trajectories of the remaining inventory, and in Figure 4.2 the stock market and execution prices for the average of the simulation trials are shown. As we can see in Figure 4.1, when the agent assumes a higher value for the stock price volatility, the optimal strategy is to liquidate the inventory with a higher trading rate at initial steps, which decrease gradually to very small values until the end of the liquidation period. This behavior is due to the risk-aversion of the agent,which results in liquidating most of the inventory at initial steps, in the highly volatile market. In this manner, the agent, which assumes a wrong higher values for the stock price volatility, has more inverse impact on the market price during the liquidation period, which is shown in Figure 4.2. However, the market price is the same for all stock price volatility values after the trading horizon, since regardless of its value, after the terminal time step, the market price is equal to:

$$S_{T+1} = S_0 - bQ_0 + \sum_{n=0}^{N} W(0, \frac{T}{N})$$

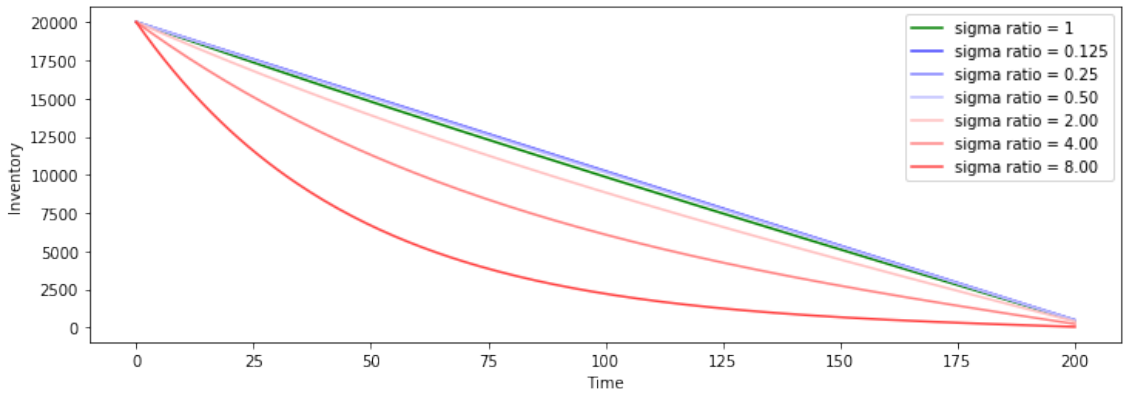

Figure 4.1: Remaining inventory trajectories during the liquidation process for the Risk-averse and big or small player agent

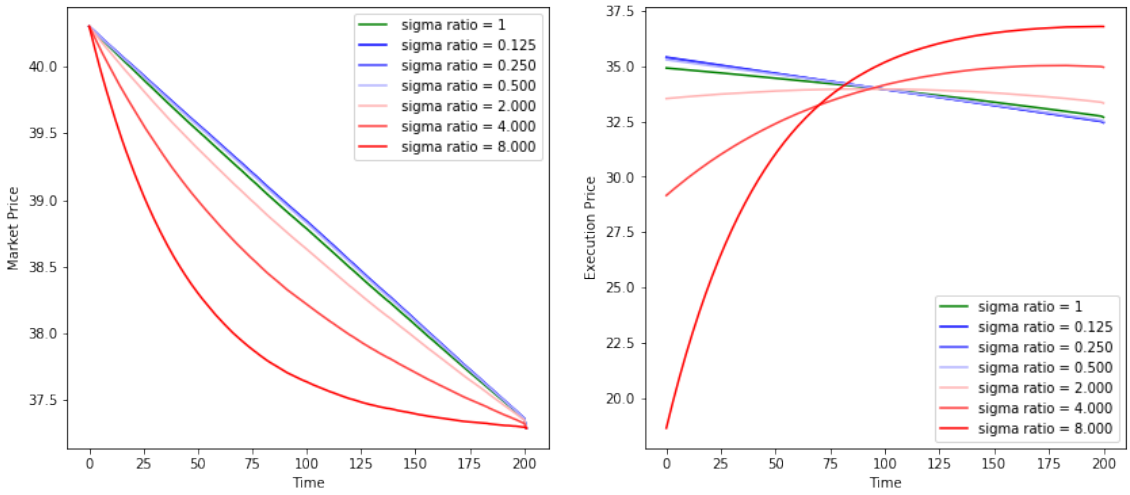

Figure 4.2: Stock market and execution price during the liquidation process for the Risk-averse and big or small player agent

According to the trading temporary impact definition, this impact only changes the execution price at the current step (and then disappears). Therefore, as we can see in Figure 4.2, having too high trading rate at the initial steps, results in the execution prices, which are much smaller than the market price at the initial time steps. While at the terminal steps, having less trading rate (close to zero) results in the execution prices, which are almost the same with the market price.

In addition, in Figure 4.3 we can see the terminal utility function ratio ($\frac{u_{T\,agent}}{u_{T\,market}}$, which is the terminal utility function for the agent, who assumes the wrong low or high stock price volatility $\sigma_{agent} \neq \sigma_{market}$ over the terminal utility function for the agent, who uses the same stock price volatility with market $\sigma_{agent} = \sigma_{market}$) against the sigma ratio ($\frac{\sigma_{agent}}{\sigma_{market}}$, which is the agent low, same or high stock price volatility σ_{agent} over the market stock price volatility σ_{market}). Moreover, in this figure, we assumed two other situations, which have exactly the same specification except for their actual stock price volatilities.

As we can see in the previous figure, when the agent assumes the correct stock price volatility, the solution and as a result, the terminal expected utility function is optimum. Moreover, when the lower stock price volatilities than the actual values are assumed, the trajectories of the remaining inventory, market, and execution prices have very small

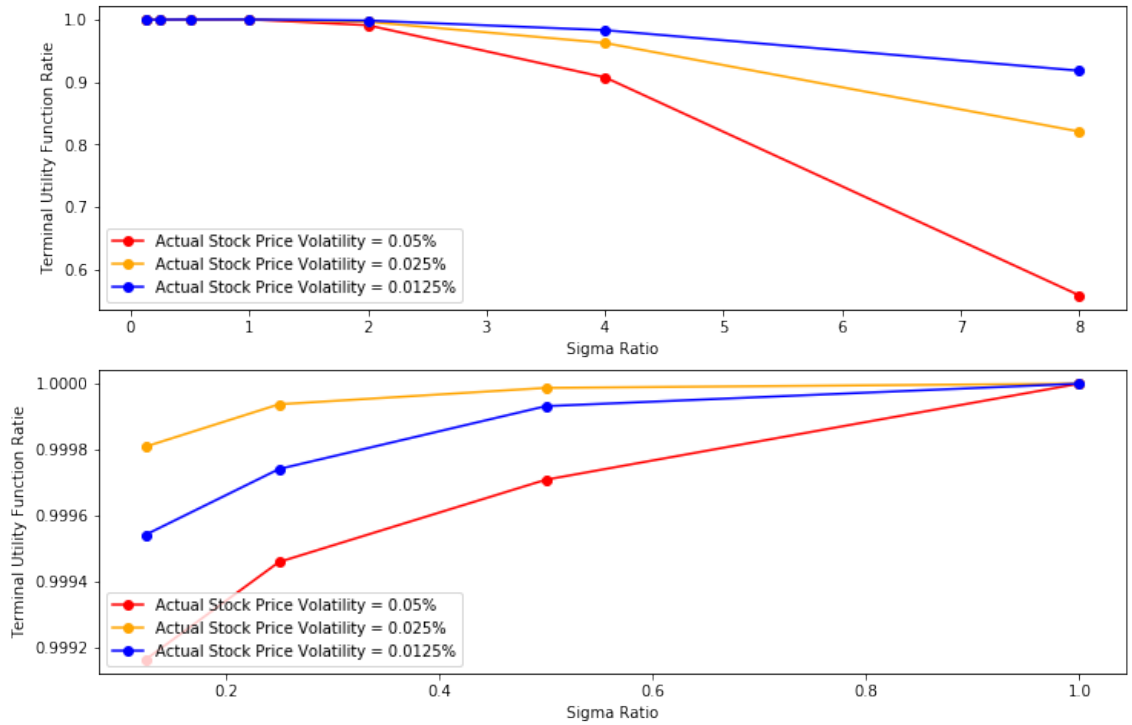

Figure 4.3: Terminal utility function ratio for three actual stock price volatility considering different agent's stock price volatility assumptions for Risk-averse and big or small player agent

changes (the lower plot zooms to the wrong lower stock price volatilities part). Therefore, for these wrong assumptions, the terminal utility function ratio is close to one. On the other hand, when the agent assumes higher stock price volatilities than the actual values, most of the shares are liquidated at the initial time steps with lower execution prices. Hence, assuming higher stock price volatilities results in lower terminal utility function value.

Furthermore, when the actual stock price volatility increases, the remaining inventory and market price trajectories turn to convex. Additionally, assuming higher values for stock price volatility increases their convexity and as a result, almost all of the shares will be liquidated at initial time steps with very low execution prices. Therefore, as the actual stock price volatility increases, the impact of having wrong higher values rises too.

It should be mentioned that according to Figure 4.3, we can compare the relative

variation of terminal cash ratios over the σ ratios. So, as we can see, for two extreme situations when $\sigma = 0.125$ or 8, the absolute value of the relative change in terminal utility function over the change in the σ ratio is equal to 0.091% and 2.111%, respectively. These two values, which show the slope of the terminal utility function to the σ ratios, demonstrate that in very large σ values, the utility function is highly sensitive. This behavior is due to the impact of large σ on increasing the initial trading rate, which are liquidated with lower execution prices.

To be concise, assuming lower stock price volatility has not a significant impact on the market price or the terminal utility function values. However, if higher values are assumed, not only the market price decreases significantly during the trading period, but also the terminal utility function (and as a result terminal cash) drops significantly too. This impact could be more significant as the actual stock price volatility increases.

Trading temporary price impact coefficient (k)

According to the trading dynamic and optimum execution solution formulas, which were mentioned in Section 3.3, the trading temporary price impact coefficient (k) has an impact on the stock execution price and the optimum trading rate parameters for both of the Risk-neutral and big player agent (both a_1 and a_2 parameters) and Risk-averse and big or small player agent (both a_1 and a_2 parameters). Therefore, for each of the two scenarios we have:

Scenario 2

Given the model setting, in Figure 4.4 we can see the trajectories of the remaining inventory, and in Figure 4.5 the stock market and execution price for the average of the simulation trials are shown. Figure 4.4 illustrates that for the agent, who assumes a lower value for the temporary price impact coefficient, the optimal strategy is to liquidate the inventory with a higher trading rate at the initial step, which decreases gradually to very small values until the end of the liquidation period. Hence, this agent has more inverse impact on the market price during the liquidation period, which is shown in Figure 4.5.

However, the market price is the same for all temporary price impact coefficient values after the trading horizon, since regardless of its value, the whole inventory will be liquidated and the cumulative impact does not change after the terminal step. Moreover, having high and low trading rates at the initial and terminal steps, results in the lower and almost the same execution price comparing to the market price. All of these behavior are due to having a big player agent. Since when the agents assume wrong lower values for k, they prefer to liquidate most of their inventory at initial steps, when their permanent impact on the market price is smaller (since at the time passes, the execution price will be decreased because both of the temporary and permanent impact).

On the other hand, when the temporary price impact coefficient is assumed to be greater than the actual value, the trading rate is almost the same at the whole trading horizon and we have a significant number of shares at the terminal time, which should be liquidated at that step. Therefore, the agent has less inverse impact on the market price, and since the trading rate is almost the same at the liquidation period, the execution price follows the same behavior as the market price.

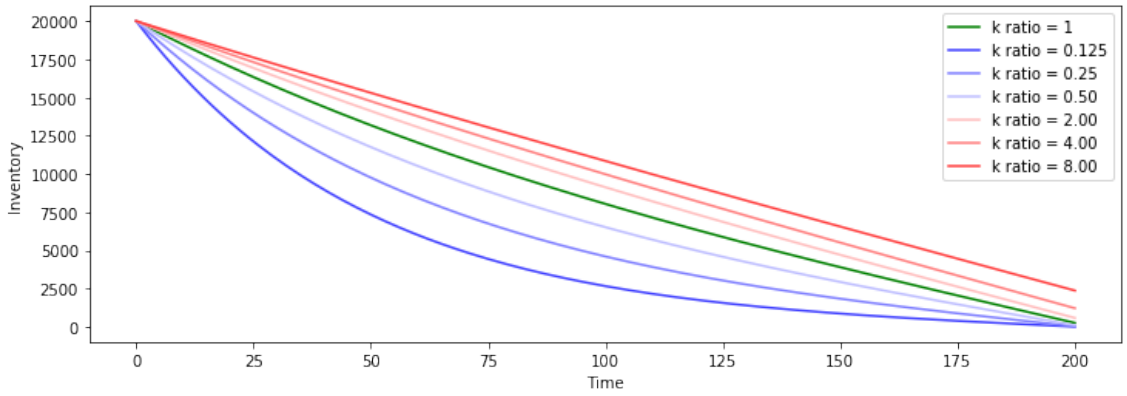

Figure 4.4: Remaining inventory trajectories during the liquidation process for the Risk-neutral and big player agent

Furthermore, since assuming higher temporary price impact coefficients results in having a significant number of shares at the terminal step, we can see a downward jump in the execution price at this step. It should be noted that, according to this method, all the remaining inventory at the terminal step is liquidated with the price of $\widehat{S_T} = S_T - \alpha Q_T$.

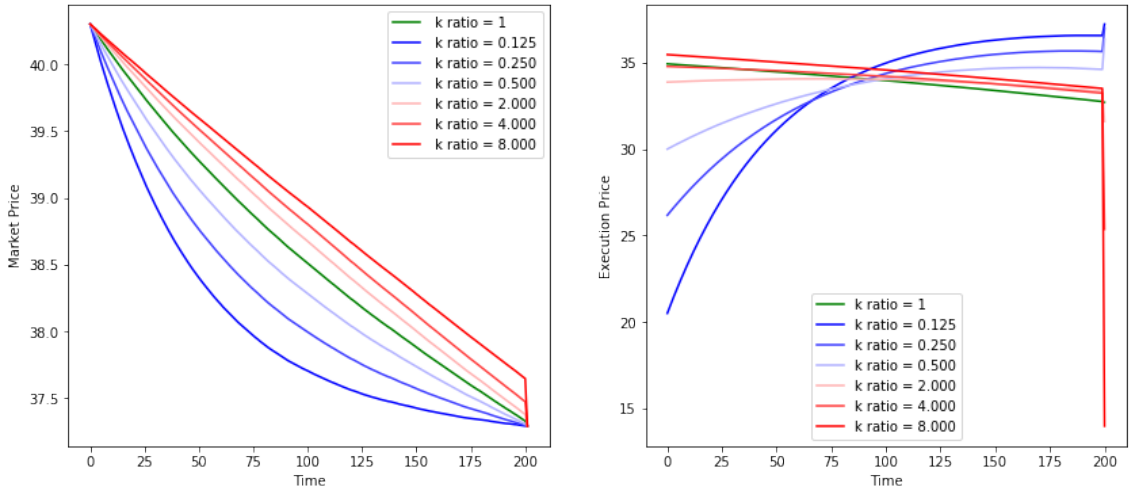

Figure 4.5: Stock market and execution price during the liquidation process for the Risk-neutral and big player agent

Therefore, since for penalizing the terminal inventory, we use a bigger value for α comparing to k parameter, this jump in the execution price happens.

The following figure shows when the agent assumes the correct temporary price impact coefficient, the solution and, as a result, the terminal utility function is optimum. Moreover, when the higher temporary price impact coefficient than the actual value is assumed, the trajectory of the remaining inventory, stock market, and execution prices do not change significantly. However, the terminal inventory will be liquidated with a very low price. Therefore, although for these wrong assumptions, the terminal utility function ratio decreases, the change is not considerable. On the other hand, when the agent assumes a lower temporary price impact coefficient than the actual value, most of the inventory will be liquidated at the initial steps with a lower execution price. Therefore, as we can see in Figure 4.6, the terminal utility function decreases significantly.

It should be mentioned that according to Figure 4.6, we can compare the relative variation of terminal cash ratios over the k ratios. So, as we can see, for two extreme situations when $k = 0.125$ or 8, the absolute value of the relative change in terminal utility function over the change in the k ratio is equal to 21.21% and 1.99%, respectively. These two values, which show the slope of the terminal utility function to the k ratios,

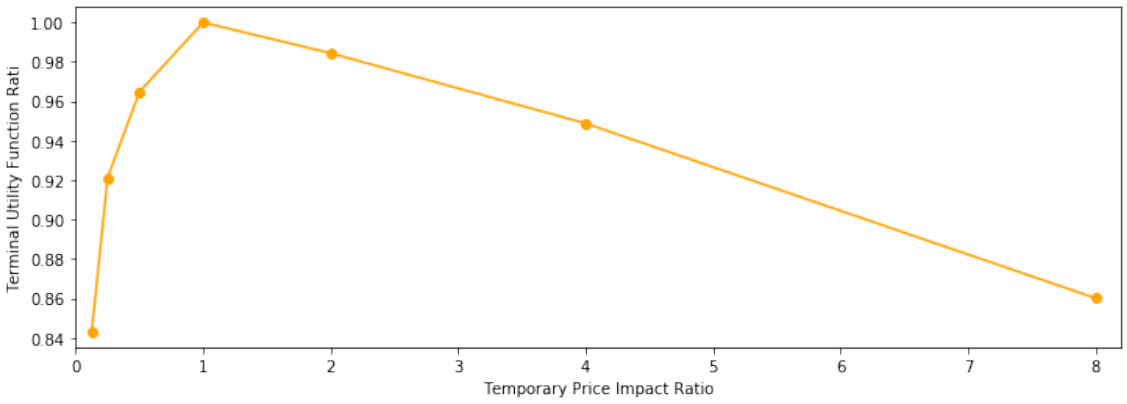

Figure 4.6: Terminal utility function ratio for different wrong agent's temporary trading impact coefficient assumptions for the Risk-neutral and big player agent

demonstrate that in very low k values, the utility function is too sensitive, and the slope is very high. This behavior is due to the impact of small k on increasing the initial trading rate, when the impact of permanent impact is not significant. Therefore, these initial executions are highly impacted by the temporary price impact, and their terminal utility function decreases.

Scenario 3

According to the model setting for the Risk-averse and big player agent, in Figure 4.7 we can see the trajectories of the remaining inventory and in Figure 4.8, the stock market and execution price for the average of the simulation trials are shown. The behavior of this model is almost the same as the previous one. However, assuming a lower temporary price impact coefficient does not make the remaining inventory and stock market price trajectories as convex as the previous method. Therefore, although the initial trading rate for the lower temporary price impact coefficient is higher comparing to the agent, who assumes the actual k, the impact on the execution price is not too much. This behavior is due to having a risk-averse agent. Since the risk-averse agent already liquidates the inventory with higher trading rate at the initial steps. Therefore, although when the agents assume wrong lower values for this parameter, prefer to liquidate their inventory sooner, the change is small. On the other hand, the agent, which assume wrong higher values,

tries to postpone the liquidation to the later steps, which results in having larger remained inventory at the terminal step, and as a result, the execution price of the terminal step is lower.

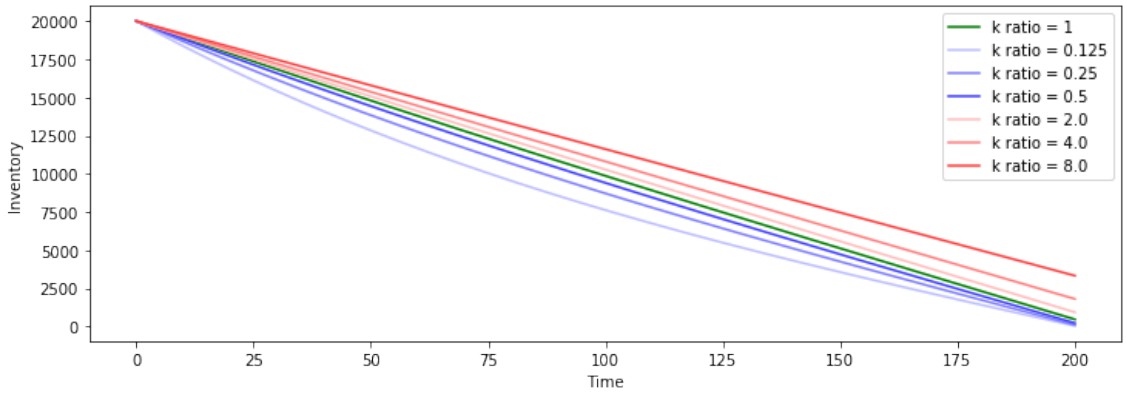

Figure 4.7: Remaining inventory trajectories during the liquidation process for the Risk-averse and big player agent

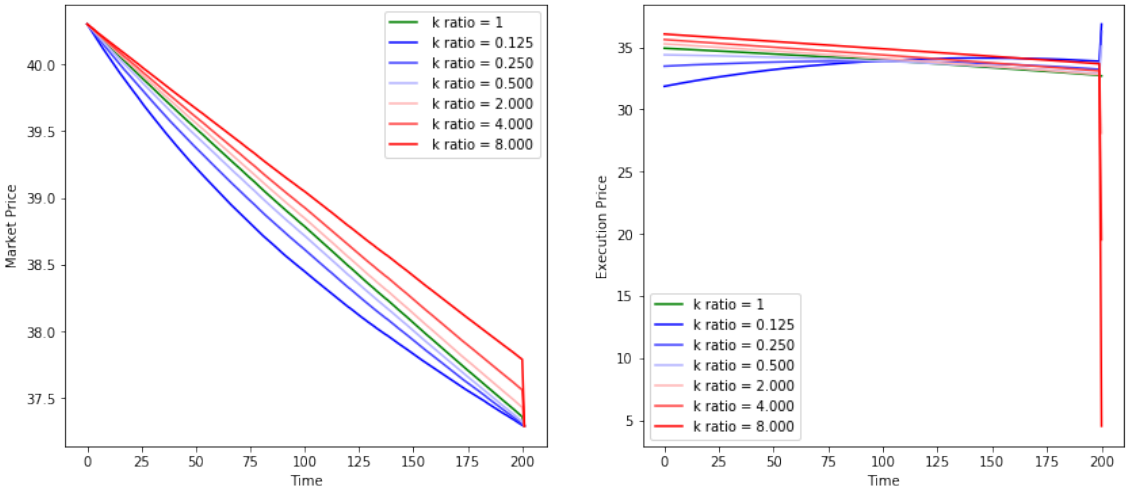

Figure 4.8: Stock market and execution price during the liquidation process for the Risk-averse and big player agent

In Figure 4.9, the terminal utility function ratios for the agents, who assume correct or wrong temporary price impact coefficients are shown. As discussed before, in the wrong higher assumptions for the temporary price impact coefficient, the large amount of terminal inventory is liquidated with very low price and as a result, the terminal utility

function decreases significantly as well. However, assuming lower k does not significantly impact the convexity of the remaining inventory and stock market price (and the initial execution price has not a change similar to the previous method). Therefore, although the terminal utility function decreases, it remains close to 1.

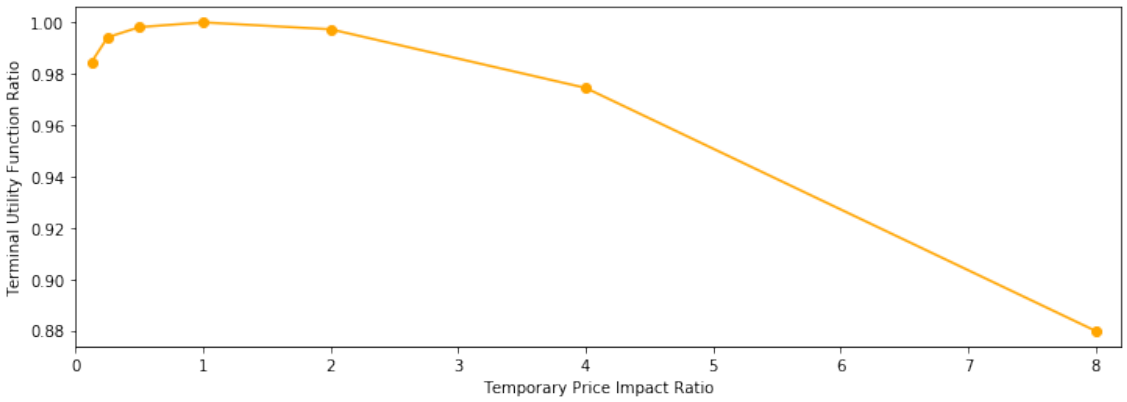

Figure 4.9: Terminal utility function ratio for three upper and lower agent's temporary trading impact coefficient assumptions for the Risk-averse and big player agent

It should be mentioned that according to Figure 4.9, we can compare the relative variation of terminal cash ratios over the k ratios for a risk-averse and big player agent. So, as we can see, for two extreme situations when $k = 0.125$ or 8, the absolute value of the relative change in terminal utility function over the change in the k ratio is equal to 2.286% and 1.717%, respectively. So, for the lower extreme k value, this relative variation value is significantly lower (for the risk-neutral and big player agent, this value was equal to 16.241). This behavior related to risk-aversion of our agent. Therefore, since the risk-averse agents liquidate their shares with higher initial trading rate, choosing wrong k value has smaller impact on their trading rates and as a result the terminal utility.

To be concise, assuming a wrong temporary price impact coefficient has a different impact on the two execution methods. Considering lower temporary price impact coefficient has an inverse impact on the terminal utility function and market price for the Risk-neutral and big player. However, although we can see this behavior for the Risk-averse and big or small player agent too, it is not as significant as the previous method. On the other

hand, assuming a higher temporary price impact coefficient has an inverse impact on the terminal utility function and the terminal execution price for the Risk-averse and big or small player agent. However, although we can see this behavior for the Risk-neutral and big player too, it is not as significant as the previous method.

Trading permanent price impact coefficient (b)

According to the trading dynamic and optimum execution solution formulas, which were mentioned in Section 3.3, the trading permanent price impact coefficient (b) has an impact on both of the market price and the optimum trading rate parameters for both of the Risk-neutral and big player (a_2 parameter) and the Risk-averse and big or small player agent (a_2 parameter). Therefore, for each of the two scenarios we have:

Scenario 2

Given the model setting, in Figure 4.10 we can see the trajectories of the remaining inventory and in Figure 4.11, the stock market and execution price for the average of the simulation trials are shown. As we can see in Figure 4.10, regardless of assuming lower or higher values for the permanent price impact coefficient, the optimal trading strategy does not have any significant change in the remaining inventory trajectory (and as a result trading rate). Therefore, the stock market and exercise price trajectories do not change significantly too.

Moreover, since at the terminal step we have a very small amount of remained inventory, the terminal execution price jumps downward (but the jump is small too). As a result, the terminal utility function for higher or lower trading permanent impact coefficient is very close to the terminal utility function, when the agent assumes the actual trading permanent impact coefficient. This means that having wrong (higher or lower) b value will result in a significant change in non of the stock market price or the terminal utility function (and as a result, terminal cash). This behavior is due to our assumptions for larger values for k comparing to the b value. Therefore, although having larger trading rate at initial steps is in favor of wrong larger assumptions for b (to avoid losing money

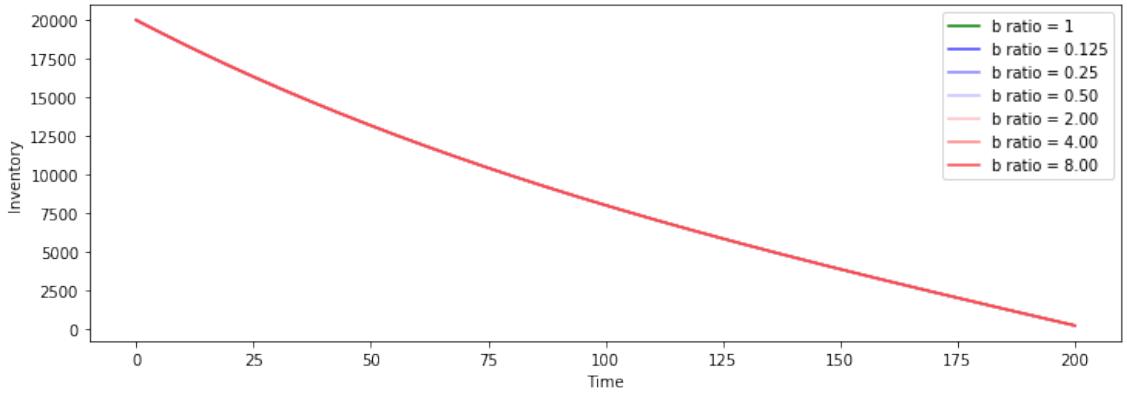

Figure 4.10: Remaining inventory trajectories during the liquidation process for the Risk-neutral and big player agent

because the permanent impact is cumulative and as the time passes, its value is bigger), it also results in liquidating the stock with very lower price because of the temporary impact. So, since the k value is 5 times larger than the b value, wrong larger b assumption, has a very small impact on the trading strategy. On the other hand, if we assume wrong smaller value for b, and postponing the trades to the later steps, we will have the same situation at the terminal step. As a result, wrong smaller b assumption has a very small impact on the trading strategy too.

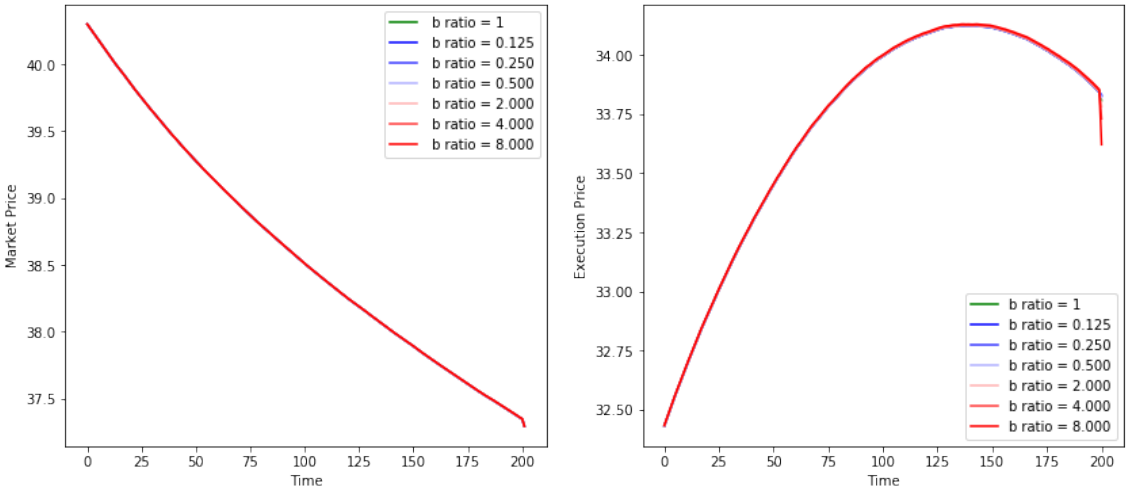

Figure 4.11: Stock market and execution price during the liquidation process for the Risk-neutral and big player agent

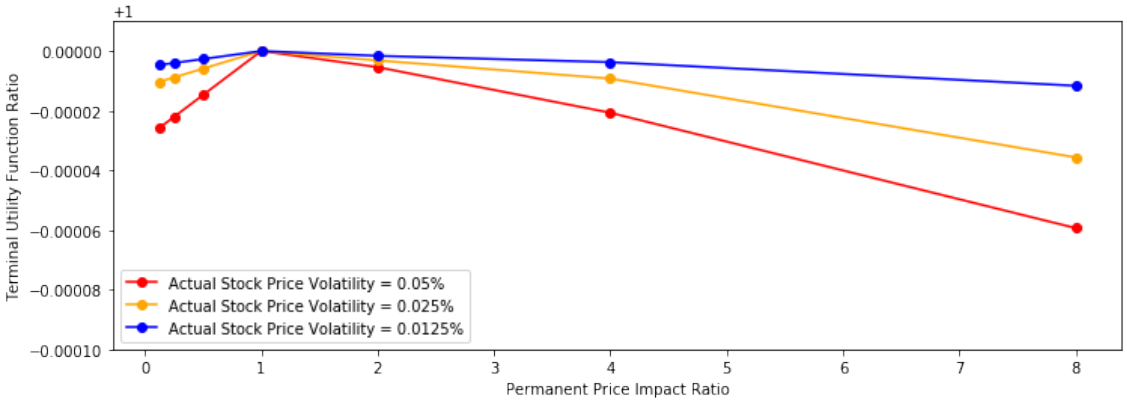

Figure 4.12: Terminal utility function ratio for different wrong agent's permanent trading impact coefficient assumptions for the Risk-neutral and big player agent

Scenario 2

According to the model setting, in Figure 4.13 we can see the trajectories of the remaining inventory, and in Figure 4.14, the stock market and execution price for the average of the simulation trials are shown As we can see in Figure 4.13, similar to the previous method, regardless of assuming lower or higher values for the permanent price impact coefficient, the optimal trading strategy does not have any significant change in the remaining inventory trajectory (and as a result trading rate). Therefore, the stock market and exercise price trajectories do not change significantly too. This behavior has a same reason with the previous method. Moreover, since at the terminal step we have a small amount of remained inventory, the terminal execution price jumps downward.

Therefore, as it is shown in Figure 4.15, the terminal utility function for higher or lower trading permanent impact coefficient is very close to the terminal utility function, when the agent assumes the actual trading permanent impact coefficient. This means that having wrong (higher or lower) b value will result in a significant change in non of the stock market price or the terminal utility function (and as a result, terminal cash).

To be concise, non of the two models have been impacted by the wrong high or low assumptions for the trading permanent impact coefficient. Therefore, these models are not sensitive to the assumptions for the trading permanent impact coefficient. Also, as we

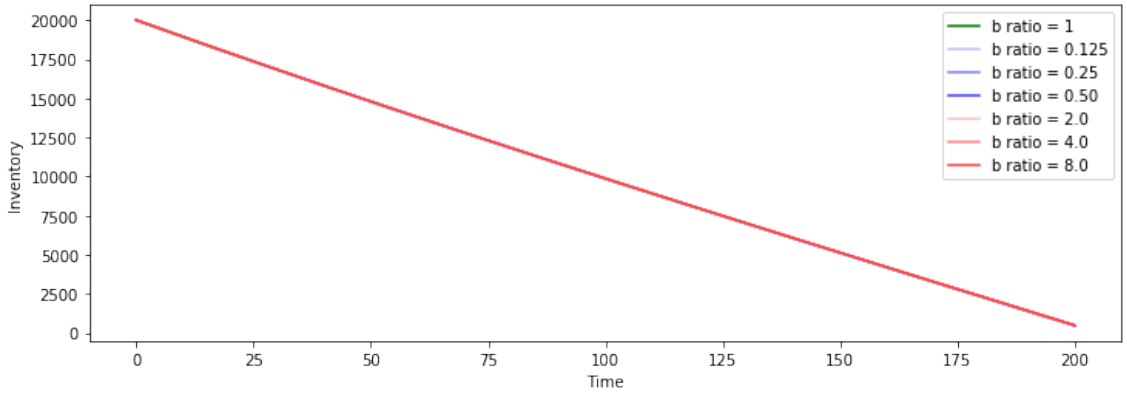

Figure 4.13: Remaining inventory trajectories during the liquidation process for the Risk-averse and big or small player agent

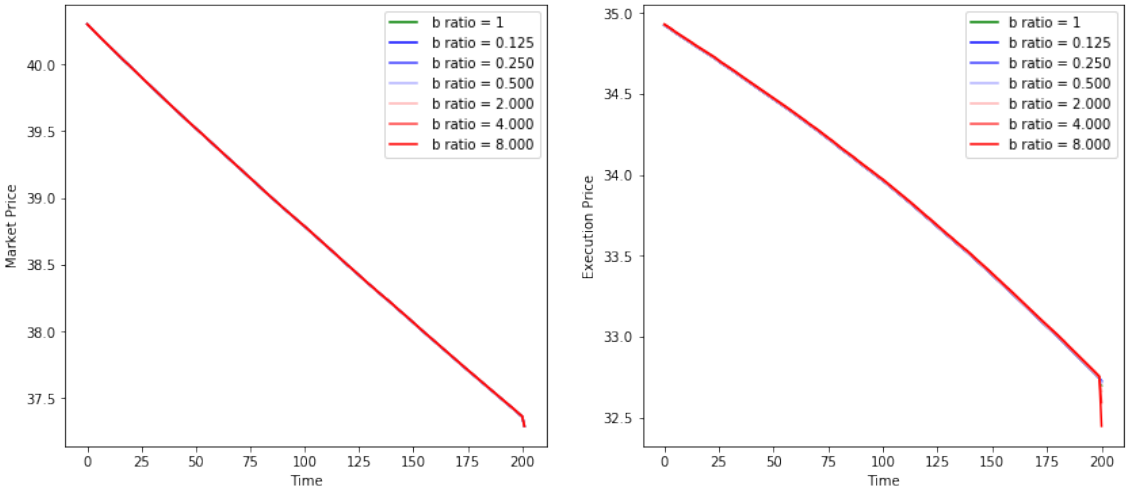

Figure 4.14: Stock market and execution price during the liquidation process for the Risk-averse and big or small player agent

can see in Figures 4.12 and 4.15, for the extreme 0.125 or 8 values, the relative variation of terminal cash ratios over the b ratios is almost zero. This behavior is due to the very small impact of b estimation in trading strategy. Since, for both of the risk-averse and risk-neutral agents, if we consider wrong large b, it should be better to liquidate the stock earlier to prevent losing money due to large permanent impact. However, since the k is larger than b (since by passing the time, the other market participants can cancel our impact on the market price by new buy or sell orders), earlier liquidation will not be efficient.

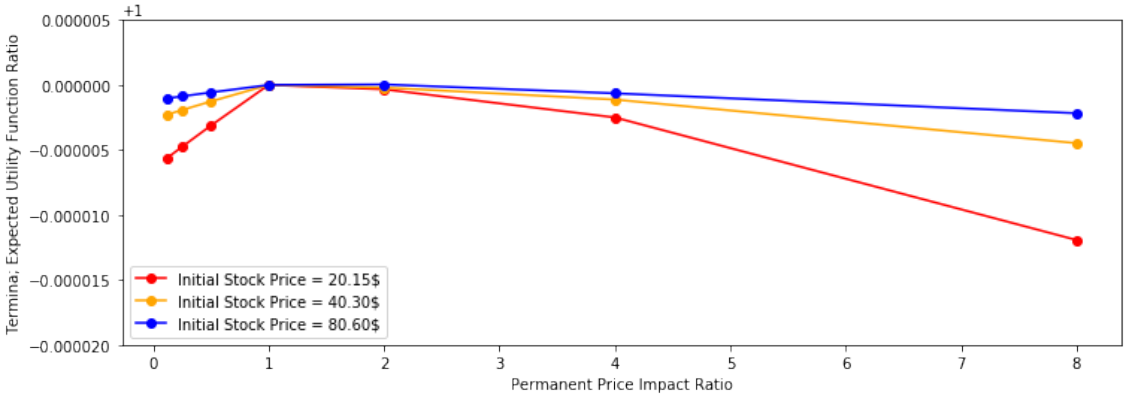

Figure 4.15: Terminal utility function ratio for three upper and lower agent's permanent trading impact coefficient assumptions for the Risk-averse and big or small player agent

Therefore, assuming wrong larger b has very small impact on the trading strategy. Also, assuming wrong smaller b, may result in keeping the stocks to liquidate in later steps. However, if we keep the stock for the terminal step, since the terminal liquidation penalty is larger than k, we lose money. Hence, assuming wrong smaller b has almost no impact on the trading strategy too.

4.4 Models comparison

As discussed before, the Risk-neutral and big player agent is not sensitive to stock price volatility estimation. Therefore, for any stock price volatility, this agent suggests the same trading rates during the trading period. So, it is important to compare its behavior with the Risk-averse and big or small player agent in different stock price volatilities.

For this purpose, we used the same experimental setting as the previous section. Additionally, we assume three low and three high σ values, when the other parameters remained unchanged.

$$\text{Low values:} \quad \frac{\sigma}{0.025\%} \in \{0.125, 0.250, 0.500\} \tag{4.7}$$

$$\text{High values:} \quad \frac{\sigma}{0.025\%} \in \{2.00, 4.00, 8.00\} \tag{4.8}$$

Therefore, in Figure 4.16 we can see the trajectories of the remaining inventory, trading rate, and the cumulative cash during the trading horizon for the different stock price volatilities. In this figure, the green and blue lines refer to the Risk-averse and big or small player agent and a Risk-neutral and big player agent, respectively.

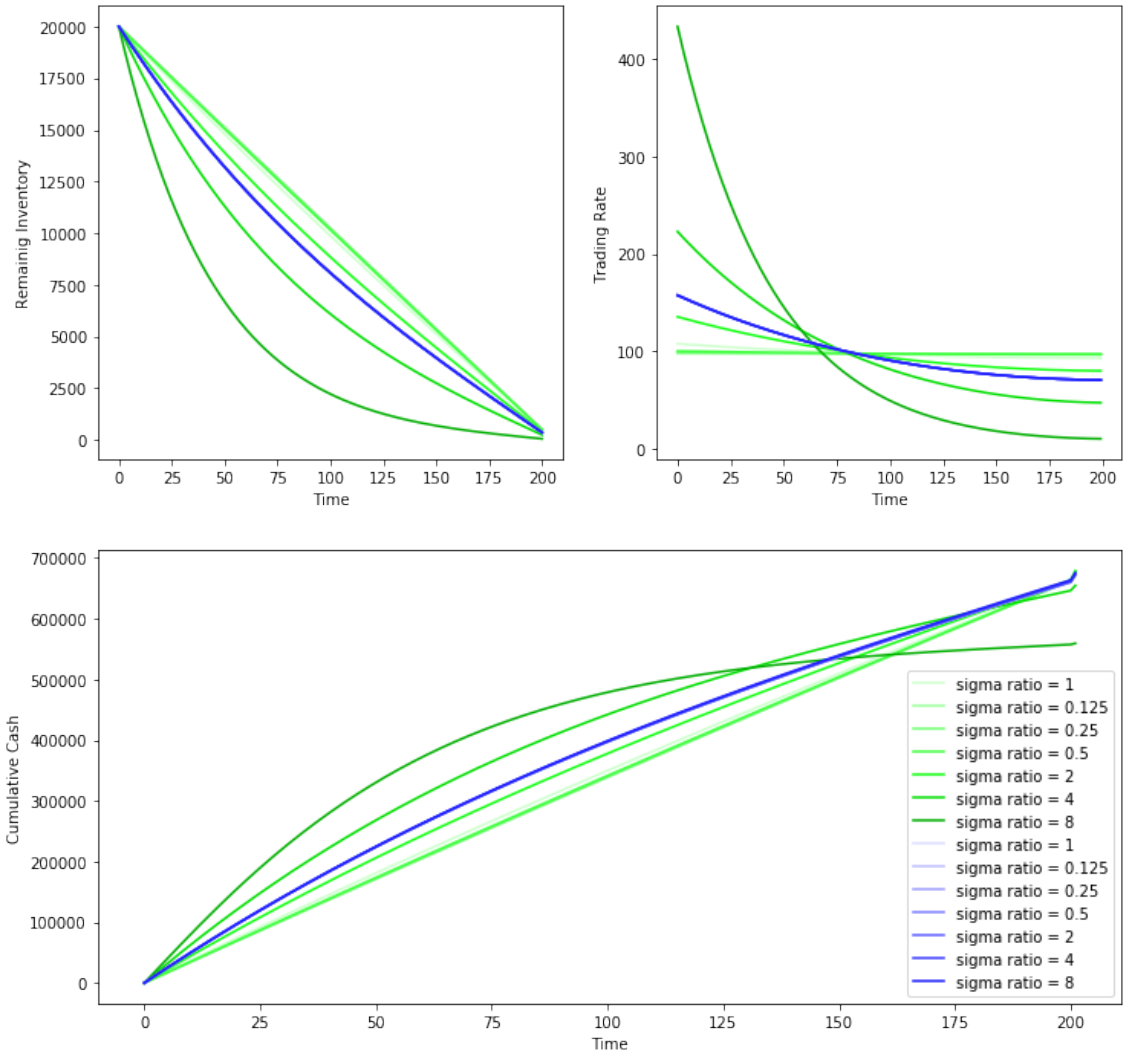

Figure 4.16: Remaining inventory, trading rate, and the cumulative cash trajectories during the liquidation process for both agents with different stock price volatilities. Green: Risk-averse and big or small player, Blue: Risk-neutral and big player

According to the previous figure, as the stock price volatility increases, the Risk-averse and big or small player agent, has higher initial trading rates. As a result, the initial transactions are being executed with lower prices, and the terminal cash decreases. However, for all the stock price volatilities the cumulative cash trajectories are almost the same when we use the Risk-neutral and big player agent.

This behavior can be explained considering the risk-aversion definition. Since for satisfying the risk-aversion of the clients, they should accept the price, which can be very large (in the higher stock price volatility, the terminal cash for the risk-neutral agent is almost 16% higher than the risk-averse agent).

4.4.1 Stock price downward jump

All the scenarios for our solution for liquidation a large number of stock in a short time are for the stock price, which follows the Brownian motion. However, in the real market, we have more complex model for the stock market, which can include up or down jumps. Therefore, in this part, we compare the behavior of our two agents, when a market jump happens. For this purpose, we use the experimental setting and the stock price volatility values, which are mentioned in the previous part. Moreover, we assume a downward jump at the midpoint of our liquidation time horizon, which is equal to half of the price at that time. Since the liquidation period is short, we assume this jump remains until the end of our trading period.

Therefore, in Figure 4.17 we can see the trajectories of the cumulative cash during the trading horizon for the different stock price volatilities. In this figure, the green and blue lines refer to the Risk-averse and big or small player agent and a Risk-neutral and big player agent, respectively.

As we can see in Figure 4.17, when the stock price volatility is too high, the Risk-averse and big or small player agent has a higher terminal cumulative cash. This is due to the having a risk-averse agent, and as a result, having large initial trading rates for this method in the liquidation period.

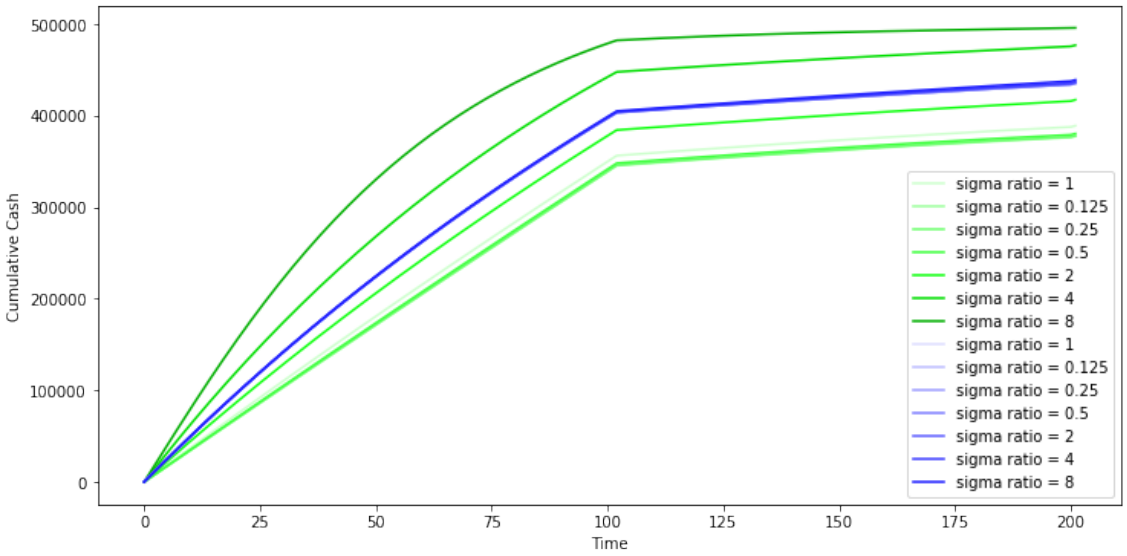

Figure 4.17: Cumulative cash trajectories during the liquidation process for both agents with different stock price volatilities. . Green: Risk-averse and big or small player, Blue: Risk-neutral and big player

Therefore, although both of our scenarios are not the solution for the market environment, which includes jumps, for the lower stock price volatility, it is better to use the Risk-neutral and big player agent assumptions. However, for the higher stock price volatility, if the agent uses the previous assumptions, and any significant jump happens during the liquidation period, the money loss is more comparing the Risk-averse and big or small player agent.

Chapter 5

Reinforcement Learning for Optimal Liquidation

In this chapter, we solve the large stock liquidation problem in short time horizon by RL method. For this purpose, we first idea of the RL model, which we want to use in our problem. Then, the methods and models' specifications are being introduced in detail. Finally, the results of using these models for the optimal liquidation problem are discussed.

5.1 Reinforcement Learning

To introduce the optimal execution method with reinforcement learning, we should first understand the theoretical concepts in RL.

Consider we want to liquidate a large number of stock in a short time horizon in a market, which follows the behavior, which was discussed in Section 3.1. However, although the market follows this environment, we don't know its behavior. Therefore, we are trading in an unknown environment, and we receive some rewards for each of our interactions with the environment. Additionally, we aim to choose decisions in order to maximize our cumulative reward. Hence, we should take a strategy to figure out the

environmental behavior, and take a sequence of actions in order to meet our goals. RL is a method, which learns a proper strategy according to its experiences from interaction with the environment, in order to maximize future rewards.

Reinforcement learning includes three concepts: state, action, and reward. Figure 5.1, describes the general procedure in RL models. At the first stage, the environment initiates the first state S_0. Then, the agent specifies the action according to S_0, chosen from a set of actions A. Consequently, the environment returns the reward r_0, and updates the state to S_1. This process will continue repeatedly until the terminal time step (or until reaching a specific goal).

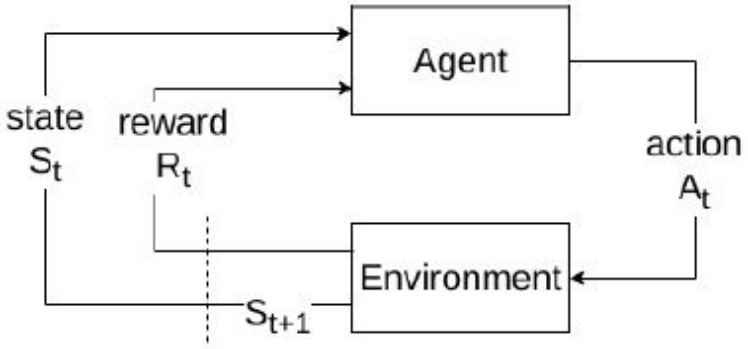

Figure 5.1: Reinforcement Learning Process

As mentioned in the literature review, RL has many applications, especially in optimal trading problems. Most of these problems can be considered as control problems with a specific goal, such as decreasing the risk and cost or increasing the wealth, and clear actions such as the type of the assets for the buy/sell orders, their volume, and timings. In this project, we aim to find the optimal trading strategy for liquidating a determined number of shares within a short time horizon.

5.2 Model

In this project, we use Q-learning as our RL method. This RL algorithm is a procedure for finding the optimal action-value function $Q^*(s,a)$. Q-learning consists of three important

components. The first one is state S, which includes the tuple of buy price (the market price at step i), and the remained inventory (the remained inventory at step i). The second component is action A, which is the number of the shares that we decide to liquidate at each stage. The final component is reward r, which is equal to the money that the agent receives after any action. In the following parts, the details specifications of our Q-learning models are discussed:

Q-learning (QL)

The first method is a tabular Q-learning method. Tabular means that there is no statistical inference between the different states. Therefore, each state's part is behaved separately. Considering remaining inventory as q and stock market price as the agent's state, the trading rate as its action, and the transacted money in each execution as the reward, allows us to construct the Q-learning function, which follows the Bellman's equation.

$$Q^*(s,a) = \mathbb{E}[r(s,a) + \max_{a'} Q^*(s',a')|s,a], \tag{5.1}$$

and

$$Q^*(s,0) = 0 \quad \text{for all of the terminal states } s \in S, \tag{5.2}$$

where s, a, s', a' refers to the current state, current action, the successive state and the successive action, respectively.

It should be mentioned that the model, which we introduced in Equation 3.4 is the same with our model in this chapter. Since, when we have a linear utility function for the risk-neutral agent, there is not any difference between the sum of the utilities and utility of the sum.

Hence, we can create our Q-function table, and initialize it randomly.For our Q-function, we used a soft updating strategy, which is:

$$Q(s,a) = Q(s,a) + \alpha[r(s,a) + \gamma \max_{a'} Q(s',a') - Q(s,a)], \tag{5.3}$$

$$0 < \alpha \leq 1, \quad \text{and} \quad 0 < \gamma \leq 1,$$

where

- α is the learning rate, which specifies how large the updates are affected by new values and in this model $\alpha = 0.5$
- γ is the discount factor and in this model $\gamma = 1$
- $r(s,a)$ is the reward function, which is computed by Equation 3.3

At the terminal state, we use $Q(s,q) = Q(s,a) + \alpha r(s,q)$.

It should be mentioned that we have fine-tuned two of the hyper-parameters including the size of the sampling batch for the action selection (0 to 500), and the percentage of exploration and exploitation to improve our outcome (0.1 and 0.9). Wrong sampling size for action selection, can result in non-optimum solutions. Because choosing a bigger sampling size can result in losing money when the agent randomly chooses a large trading rate (a large trading rate has more impact on the execution price). While, choosing a smaller sampling size may cause remaining more inventory for the last time period, and losing money due to the impact of liquidating them on the execution price.

Moreover, in 90% of the times, the agent chooses an action a^* from $a^* = arg\max_a Q(s,a)$, and in 10% of the times, the agents selects a random action a^* from $a^* = Uniform[0,...,1000]$.

Q-learning is a model-free RL model, which allows the agent to learn directly from its experience of interaction with the environment (with no need for modelling the environment). Also, Q-learning is an off-policy method. Therefore, we estimate the reward for subsequent actions and add it to the new state without actually following any greedy policy.

Q-learning has two major disadvantages. Firstly, this algorithm approximates the optimal policy on a point-wise basis. Therefore, this behavior seriously restricts its ability for interpolation between possible actions, which results in ineffective generalization.

Secondly, in Q-learning, the sample is not being updated at each epoch. So, we lose the previous experience. In this project, we have used two methods to address these problems.

Q-learning with linear approximation (QLLA)

As mentioned in Section 5.2, for using the tabular Q-learning, we have to discretize our state space. This discretization may result in two main issues. Firstly, if we choose completely wrong bins, all the states could be sent to the same bin, and there is a possibility that all of the information became demolished! Secondly, the binning system adds extra hyperparameters to the RL model. So, since reinforcement learning algorithms, in general, are very sensitive to hyperparameters, the binning system makes our model more sensitive to any change.

To tackle these problems, we used a linear function approximation. Therefore, although in a previous model, $Q(s,a)$ used to be a table, now for every $a, s \rightarrow Q(s,a)$ will be a linear regression model. Then, we explore the environment as discussed in the previous part, and construct the transition (s,a,r,s').

Accordingly, the only difference with the previous method is the way we update our Q-function at each stage, after observing the transition. in the previous part, the linear model is fitted with all data in one shot. However, by using the linear approximation, we update the transition after observing each of the steps. The procedure is like the stochastic gradient descent in which we train a model by using an online scheme. So, if we have a tuple of (s,a,r,s') (including the state and action at time t, the reward according to the action, and the next state at time $t+1$), we can update our linear model by $(s, r + \max_{a'} Q(s',a'))$.

Q-learning with linear approximation and experience replay (QLLAER)

The next drawback of the Q-learning is that we lose our useful previous information and experience. Experience replay is a method to address this issue, and increase the sample efficiency. In this method, we store the transitions in a buffer, and in the updating

procedure, we batch some of the samples in our buffer.

Experience replay stores the previous samples in a buffer, which can be useful in the updating process at each stage. So, at each stage, the buffer memory grows, and after several steps, since the buffer includes many of the previous transactions, the exploitation is more exploration. So, as our buffer samples increase, the weight of the new observation decreases.

To be more precise, in experience replay, we store the agent's experiences $e_i = (s_i, a_i, r_i, s_{i+1})$ in a table. Then, in the exploration of the learning phase, we gain experience based on taking random samples from the table, which include all the past experiences. Therefore, as we gain more experience, and have the table including many tuples of previous data, we explore actions closer to the optimal ones.

5.2.1 Data

As mentioned in Chapter 3, the trading rates have an impact on the stock price. In this project, we assumed that the market follows the scenario 2. Therefore, our trades have permanent and temporary impact on the market price.

For the numerical setting, we assume that the market has same specifications and numerical setting as mentioned in Table 4.1 and Section 4.2. So, we want to liquidate $Q_0 = 20,000$ stock shares within the limited time period $T = 200$.

Moreover, for the market data generation, we used the Monte-Carlo method. There are two reasons for using this method instead of using actual or historical data. The first reason is that, for the historical data, it is difficult to consider the impact of our trades on the market price (since collecting the historical bid/ask table data with desirable frequency is difficult and expensive). In addition, using real-time market data for the training or even testing steps, could result in losing money, which is not affordable as an academic research (or even for the financial institutions).

Since we consider the permanent impact of our trading in the market model, according to Equation 3.1, the market price follows follows the following formula:

$$S_{i+1} = S_i - g(v_i)\delta + \sigma W_\delta \tag{5.4}$$

Moreover, at each state the reward is computed with the following formula:

$$r(S_i, v_i) = (S_i - kv_i)v_i \tag{5.5}$$

So, according to our action v_t, and the market price (state) S_t, the reward function is computed.

5.2.2 Evaluation

In this project, we have used the random and analytical solutions as the lower and upper bounds for evaluating the RL model. Therefore, as the result of our RL solution converges to the lower or upper bounds, we can show the efficiency or inefficiency of our model.

Random solution

The random solution is the strategy, which we consider as our lower bound for the RL model performance evaluation. This unintelligent agent determines its random trading rate at each time. This value can be a random integer number between 0 and Q_t. Therefore, since this agent does not follow any certain strategy, we can consider it as a lower bound for the performance evaluation of our model.

Analytical solution

As we discussed in Chapter 3, for the risk-neutral and big-player, there is an analytical solution. According to this solution, we should liquidate our inventory with a trading rate, which discussed in Section 3.3. Since the analytical solution has the global maximum outcome, we considered it as an upper bound for our trading strategy.

5.2.3 Hyperparameters tuning

The implementation of Q-learning method has three hyperparameters; learning rate, discount factor and the number of episodes. These parameters were partially chosen by heuristic and experimentation. Therefore, we chose 10000 for the number of episodes for having a reasonable running time. Additionally, since we are liquidating our inventory within a short time, we chose 1 for discount factor. Also, we selected 0.5 for the learning rate to have a soft, Bellman-style update to our Q-function estimate.

Furthermore, as discussed in Section 5.3, for the the action selection in our Q-learning model, we chose a $\{1, ..., 500\}$ list, which results in the best outcomes. Also, for in this method, for 90% of the time, we chose the action by exploitation, and the other 10%, the action is selected by exploration.

5.3 Results and Discussion

In this section, we discuss and compare the performance of the three RL models.

5.3.1 Lower bound: Random trading strategy

For the lower boundary, we randomly select the trading rate for each period, and the histogram of the terminal cash is depicted in Figure 5.2. It is observed that the distribution of the terminal cash is very dispersed with an average and standard deviation of $687,572.09$\$ and $19,164.70$\$, respectively. Having a large standard deviation comparing to the average is due to the random selection strategy from a very broad range of 0 to Q_t.

5.3.2 Upper bound: Analytical solution

According to the analytical solution, we chose the trading rate value, and the histogram of the terminal cash of this strategy is presented in Figure 5.3. According to this graph, the distribution of the terminal cash is almost normal with an average and standard deviation of is $788,508.15$\$ and $1,674.21$\$ respectively. Since by following the analytical solution,

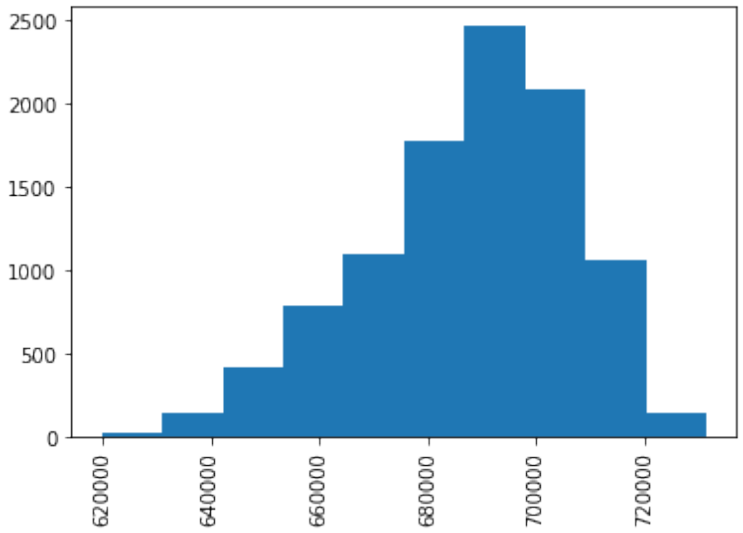

Figure 5.2: The terminal cash histogram for the randomly trading strategy of the simulated stock market price for 10,000 trajectories in the 200 time steps

the trading strategy does not vary, which results in smaller standard deviation comparing to the random strategy. We consider the outcome of this strategy as the upper bound of our RL methods.

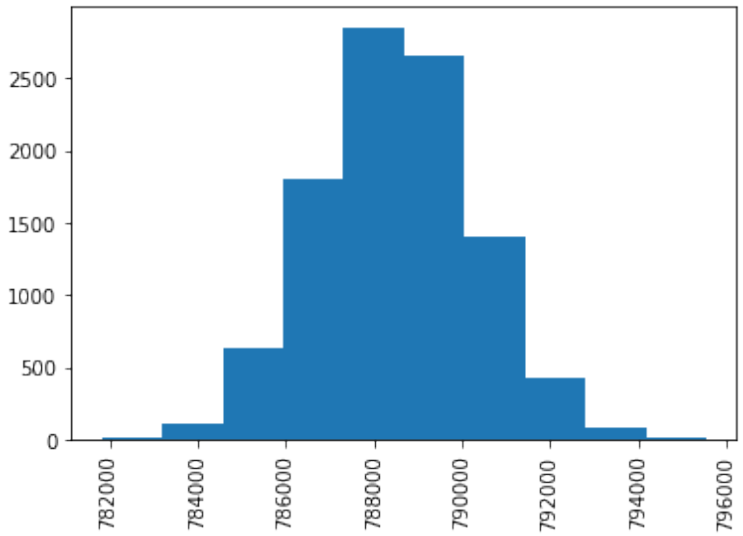

Figure 5.3: The terminal cash histogram for the analytical solution's trading strategy of the simulated stock market price for 10,000 trajectories in the 200 time steps

Therefore, now we have the lower and upper bounds to assess our RL agents' perfor-

mance. It is interesting to note that the terminal cash for analytical solution (upper bound) is almost 12.80% larger than the random strategy (lower bound).

5.3.3 Q-learning strategy

In this trading strategy, we followed Section 5.3 methodology to find the Q-learning parameters (state, action, reward, and Q-function). Figure 5.4 shows the terminal cash histogram of the agent, which trades according to the Q-learning strategy.

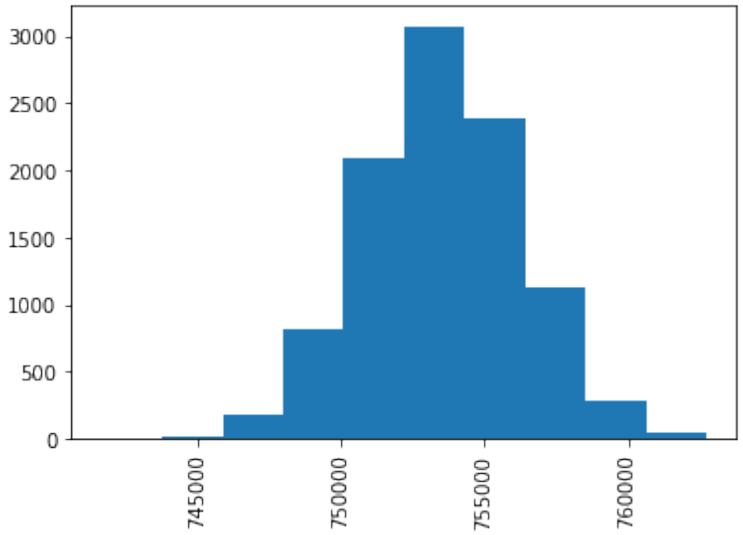

Figure 5.4: The terminal cash histogram for the Q-learning trading strategy of the simulated stock market price for 10,000 trajectories in the 200 time steps

As we can see in Figure (histogram) 5.4, the terminal cash's average and standard deviation for the agent follows the QL model are $753,315.27$ and $2,827.55$ respectively. Therefore, not only we have smaller standard deviation comparing to the random solution, but also we gain almost 95.54% of the terminal cash of the analytical solution.

Also, Figure 5.5 shows the training evolution of this method. As we can see in this figure, this model has a better performance as it trains more.

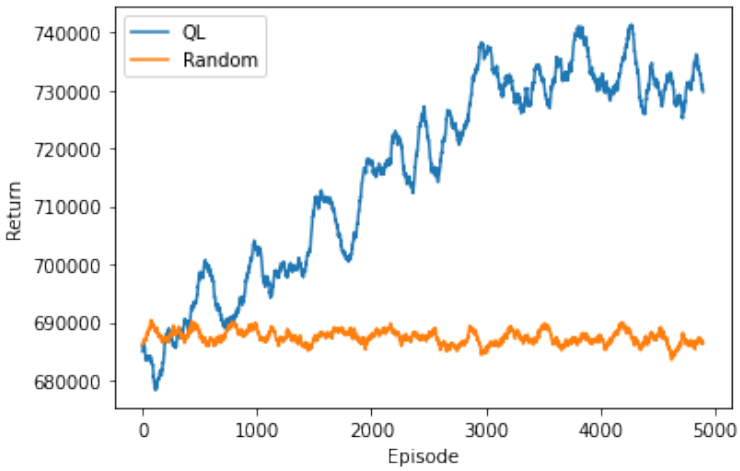

Figure 5.5: The training evolution of the Q-learning trading strategy

5.3.4 Q-learning strategy with linear approximation

In this strategy, we used the linear approximation method for updating our Q-functions. Figure 5.6 presents the terminal cash histogram of the agent, which trades according to this strategy.

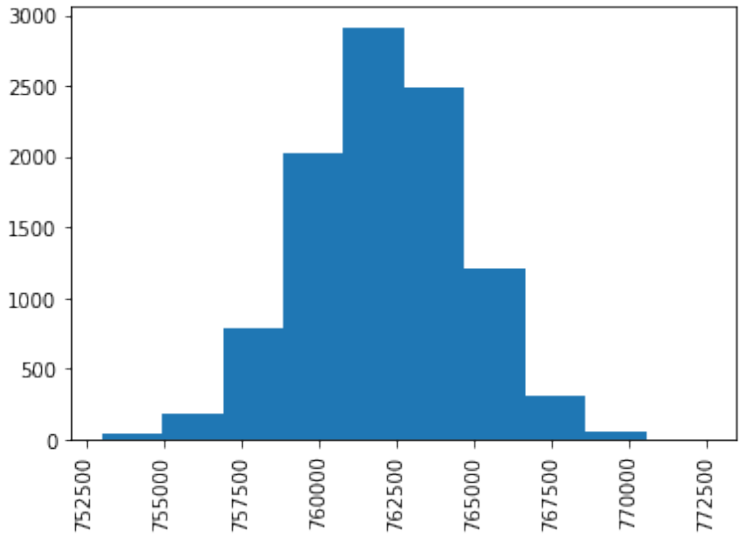

Figure 5.6: The terminal cash histogram for the Q-learning with linear approximation trading strategy of the simulated stock market price for 10,000 trajectories in the 200 time steps

The agent who uses the QLLA strategy has the terminal cash with an average and standard deviation of $762,120.29\$$ and $2,558.21\$$. As we can see, this method has a slightly better performance comparing to the Q-learning method (the terminal cash is almost 96.68% of the analytical solution). As we can see in Figure 5.7, this method has a faster training speed comparing to the Q-learning method. This behavior is due to the updating procedure of the Q-functions with linear approximation.

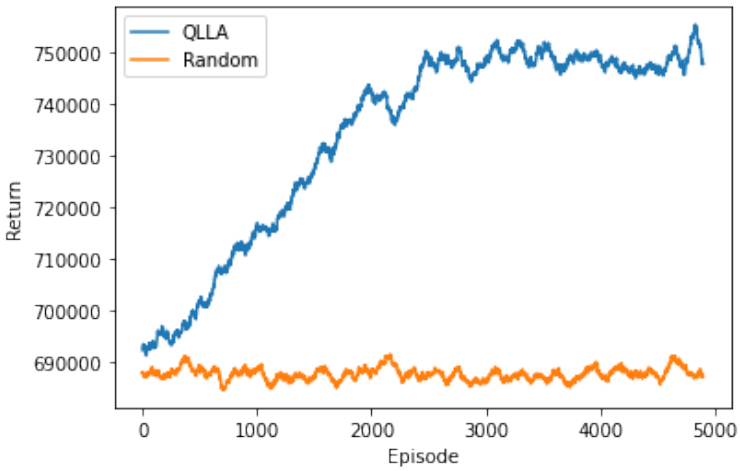

Figure 5.7: The training evolution of the Q-learning with linear approximation trading strategy

5.3.5 Q-learning strategy with linear approximation and experience replay

In this part, we followed the QLLAER trading strategy. Figure 5.8 shows the histogram of the agent, which trades according to the QLLAER strategy. All the settings for this strategy are exactly the same as the previous method, and the only change is related to using experience replay for the sampling buffer updating. The outcome of this strategy has an average and STD of $777,326.80\$$ and $2,029.91\$$, which shows that this strategy has the highest terminal cash and lowest STD. This behavior is due to using both of the linear approximation for Q-function updating, and experience replay for keeping the past information.

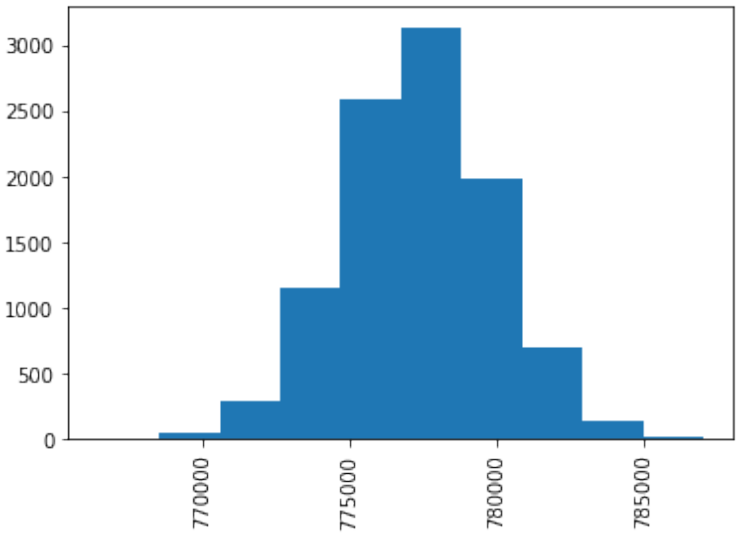

Figure 5.8: The terminal cash histogram for the Q-learning with Linear Approximation and Experience Replay trading strategy of the simulated stock market price for 10,000 trajectories in the 200 time steps

5.3.6 Discussion

In general all the three Q-learning methods showed a great behavior in short-term large stock liquidation. To be more detailed, we consider the terminal cash TC as the main indicator of the solution's performance, and introduce $\beta = \frac{TC_{model} - TC_{random}}{TC_{analytical} - TC_{random}}$ as a parameter, which shows this indicator. Therefore, for the random, Q-learning, QLLA, QLLAER, and analytical methods, the β value is equal to 0%, 65.13%, 73.86%, 88.92%, and 100% respectively. However, our initial hyperparameter choice about the action selection range has a significant impact on this indicator parameter. Since choosing larger numbers for the right side of this range allows the RL agent to select very large values at the exploration, and as a result the terminal cash is lower, and vice versa.

It should be mentioned that this performance is for the risk-neutral RL agent. Therefore, if we have a risk-averse agent, we should consider a discount ratio lower than 1, which force the agent to liquidate the stocks sooner.

Moreover, in order to have a better understanding of our three RL methods, in the following figure, we compare the terminal cash of the RL models with analytical solution

when we estimate wrong k values. As we can see, all three methods have outcomes better than the random strategy, and almost close to the analytical solution outcome when we assume the correct k value. However, the QLLER ,method has the best performance comparing to the other ones. Therefore, using the RL methods can decrease the model's parameters estimation risk of these solution. Since, although in the RL models, we do not need to estimate model's parameters, their performance is highly close to the analytical solution.

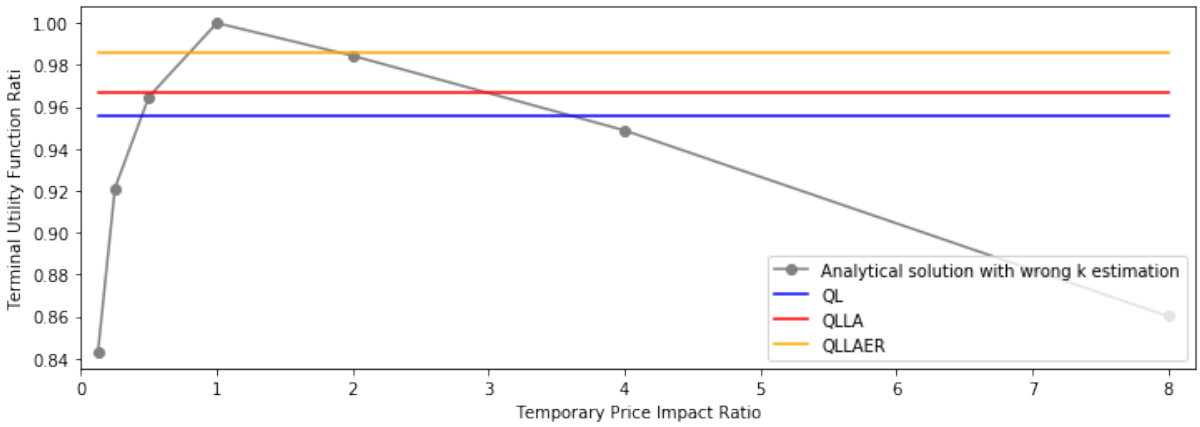

Figure 5.9: Comparison of the terminal cash of the RL models with analytical solution when we estimate wrong k values

Chapter 6

Conclusion

1: Firstly, we evaluated the sensitivity of analytical solutions for optimal liquidation problem to their parameters estimation.

- 1.1: Our assessment demonstrates that stock market price is significantly sensitive to stock price volatility estimation for the risk-averse and big or small player agent. Moreover, this impact is more intense when we estimate wrong larger values for this parameter (for instance, when we estimates the $\sigma_{agnet} = 8\sigma_{market}$, the market price decreases up to 4% during the trading period). However, the market price for the risk-neutral and big or small player agent, is not sensitive to the σ estimation.

  In addition to above, the execution price, and as a result the utility function are highly sensitive to σ estimation for the risk-averse and big or small player agent. However, this impact is more serious, when we estimate wrong larger values for this parameter (For instance, when we estimates the $\sigma_{ganet} = 8\sigma_{market}$, the terminal utility function decreases up to 18%). This impact is more intense as the actual σ increases (For example, when we doubled the actual σ in our experimental setting, and estimates the $\sigma_{agent} = 8\sigma_{market}$, the terminal utility function decreases up to 42%). However, the execution price and the utility function for the risk-neutral and big or small player agent, are not sensitive to the σ estimation.

- 1.3: Our results show that the stock market price is significantly sensitive to trading temporary price impact coefficient estimation, for both of the risk-neutral and big player, and risk-averse and big or small player agents. Additionally, this impact is negative or positive when we estimate wrong larger or smaller values for this parameter (For instance, when we estimates the $k_{ganet} = 8k_{market}$ or $k_{ganet} = 0.125k_{market}$, the market price decreases up to 2.1%, or increases up to 1% during the trading period for the risk-neutral and big player agent).

  Furthermore, the execution price, and as a result the utility function are highly sensitive to k estimation for both of the risk-neutral and big player, and risk-averse and big or small player agents. While for the risk-neutral and big player agent this impact is more serious, when we estimate wrong smaller values for this parameter, for the risk-averse and big or small player agent, this impact is more intense, when we estimate wrong larger values for this parameter (For instance, when we estimates the $k_{ganet} = 0.125k_{market}$, the terminal utility function decreases up to 16% for the risk-neutral and big player agent, and when we estimates the $k_{ganet} = 8k_{market}$, the terminal utility function decreases up to 12% for the risk-averse and big or small player agent).

  Our finding shows that the market and execution price, and as a result the utility function are not sensitive (have very small changes) to the trading permanent price impact coefficient estimation for both of the risk-neutral and big player, and risk-averse and big or small player agents.

  The parameters misspecification can result in having higher trading rates, and as a result lower remained inventory at the initial steps. Consequently, the market price during the trading horizon decreases. This market price inverse impact can increase the regulatory risk of the agents. Since any deliberate attempts in order to impact the market price is realized as a market manipulation, which is illegal, and it is difficult to show that these impacts are due to the nondeliberate parameters misspecifications.

Our findings showed that the risk-averse agent has better performance when there is a big downward jump in the market price.

2: Secondly, we developed three RL solutions for the optimal liquidation problem. The comparison of obtained results with the analytical solution and random method indicates that the performance of the Q-learning methods outperforms the random method, while the second and third methods are very close to the upper bound. It is worth mentioning that the Q-learning with linear approximation and experience replay demonstrated slightly better terminal average cash and more stability. Although, since we do not estimate any parameters in these strategies, they have a really good results comparing to the analytical solution. Therefore, they could be considered as effective alternatives for the analytical solutions when we are not completely sure about our estimation for their parameters.

Moreover, this research could be developed in several directions:

- 1: Since the stock market environment, which we have used in our problem is one of the existing models in the literature, we suggest that more market environment models like the stock market with both temporary and permanent trading impact and execution strategy will be considered.

- 2: In this study, the impact of existing another agent in the market is not considered. However, this assumption is not appropriate for some popular stocks like AAPL. Therefore, assuming another big financial institutions in the market (multi agent scenario) during the trading period and find the optimal trading strategy when all of them have impact on the market, could be an interesting future study.

- 3: Since the stock market has many instantaneous regime shifts, doing the stress and scenario testing for any new method is crucial. Therefore, one line of study for any new proposed trading strategy is to evaluate it (for example the model, which we proposed here) under many financial scenarios.

- 4: A potential extension for this study can be applying other Machine Learning (ML) methods such as counterfactual ML method in order to compare the performance of these strategies with the current study.

Bibliography

M. Aitken, A. Aspris, S. Foley, and F. d. B. Harris. The effects of algorithmic trading on security market quality. Technical report, SSRN Working paper, 2013.

R. Almgren and N. Chriss. Optimal execution of portfolio transactions. *Journal of Risk*, 3:5–40, 2001.

A. Arévalo, J. Niño, G. Hernández, and J. Sandoval. High-frequency trading strategy based on deep neural networks. In *International conference on intelligent computing*, pages 424–436. Springer, 2016.

J. Arnoldi. Computer algorithms, market manipulation and the institutionalization of high frequency trading. *Theory, Culture & Society*, 33(1):29–52, 2016.

F. C. Authority. Algorithmic trading compliance in wholesale markets, 2018.

T. Bansal, J. Pachocki, S. Sidor, I. Sutskever, and I. Mordatch. Emergent complexity via multi-agent competition. *arXiv preprint arXiv:1710.03748*, 2017.

W. Bao and X.-y. Liu. Multi-agent deep reinforcement learning for liquidation strategy analysis. *arXiv preprint arXiv:1906.11046*, 2019.

W. Bao, J. Yue, and Y. Rao. A deep learning framework for financial time series using stacked autoencoders and long-short term memory. *PloS one*, 12(7):e0180944, 2017.

V. Behzadan and A. Munir. Vulnerability of deep reinforcement learning to policy induction attacks. In *International Conference on Machine Learning and Data Mining in Pattern Recognition*, pages 262–275. Springer, 2017.

V. Behzadan and A. Munir. Adversarial reinforcement learning framework for benchmarking collision avoidance mechanisms in autonomous vehicles. *IEEE Intelligent Transportation Systems Magazine*, 2019.

J. Carapuço, R. Neves, and N. Horta. Reinforcement learning applied to forex trading. *Applied Soft Computing*, 73:783–794, 2018.

A. Cartea, S. Jaimungal, and D. Kinzebulatov. Algorithmic trading with learning. ssrn elibrary, 2013.

Á. Cartea, S. Jaimungal, and J. Penalva. *Algorithmic and high-frequency trading*. Cambridge University Press, 2015.

Á. Cartea, R. Donnelly, and S. Jaimungal. Algorithmic trading with model uncertainty. *SIAM Journal on Financial Mathematics*, 8(1):635–671, 2017.

E. P. Chan. *Quantitative trading: how to build your own algorithmic trading business*. John Wiley & Sons, 2021.

G. Clark, M. Doran, and W. Glisson. A malicious attack on the machine learning policy of a robotic system. In *2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*, pages 516–521. IEEE, 2018.

M. A. Dempster and V. Leemans. An automated fx trading system using adaptive reinforcement learning. *Expert Systems with Applications*, 30(3):543–552, 2006.

Y. Deng, F. Bao, Y. Kong, Z. Ren, and Q. Dai. Deep direct reinforcement learning for financial signal representation and trading. *IEEE transactions on neural networks and learning systems*, 28(3):653–664, 2016.

C. Duhigg. Stock traders find speed pays, in milliseconds. *Sat*, 1(05), 2009.

G.-G. S. Fletcher. Deterring algorithmic manipulation. *Vand. L. Rev.*, 74:259, 2021.

M. Glantz and R. Kissell. *Multi-asset risk modeling: techniques for a global economy in an electronic and algorithmic trading era*. Academic Press, 2013.

Y. Han, B. I. Rubinstein, T. Abraham, T. Alpcan, O. De Vel, S. Erfani, D. Hubczenko, C. Leckie, and P. Montague. Reinforcement learning for autonomous defence in software-defined networking. In *International Conference on Decision and Game Theory for Security*, pages 145–165. Springer, 2018.

B. Huang, Y. Huan, L. D. Xu, L. Zheng, and Z. Zou. Automated trading systems statistical and machine learning methods and hardware implementation: a survey. *Enterprise Information Systems*, 13(1):132–144, 2019.

S. Huang, N. Papernot, I. Goodfellow, Y. Duan, and P. Abbeel. Adversarial attacks on neural network policies. *arXiv preprint arXiv:1702.02284*, 2017.

S. Kaur. Algorithmic trading using sentiment analysis and reinforcement learning. *positions*, 2017.

A. A. Kirilenko and A. W. Lo. Moore's law versus murphy's law: Algorithmic trading and its discontents. *Journal of Economic Perspectives*, 27(2):51–72, 2013.

R. Kissell. *The science of algorithmic trading and portfolio management*. Academic Press, 2013.

J. Klepacki et al. Innowacyjne techniki inwestycyjne na przykładzie ultraszybkich transakcji (high frequency trading, hft). *Przedsiębiorczość i Zarządzanie*, 16(11.2): 205–215, 2015.

Y. Li. Deep reinforcement learning: An overview. *arXiv preprint arXiv:1701.07274*, 2017.

O. Linton and S. Mahmoodzadeh. Implications of high-frequency trading for security markets. *Annual Review of Economics*, 10:237–259, 2018.

W. Mattli. *Global algorithmic capital markets: high frequency trading, dark pools, and regulatory challenges*. Oxford University Press, 2018.

J. Moody and M. Saffell. Learning to trade via direct reinforcement. *IEEE transactions on neural Networks*, 12(4):875–889, 2001.

I. Moosa. The regulation of high-frequency trading: A pragmatic view. *Journal of Banking Regulation*, 16(1):72–88, 2015.

R. K. Narang. *Inside the black box: A simple guide to quantitative and high frequency trading*, volume 846. John Wiley & Sons, 2013.

Y. Nevmyvaka, Y. Feng, and M. Kearns. Reinforcement learning for optimized trade execution. In *Proceedings of the 23rd international conference on Machine learning*, pages 673–680, 2006.

B. Ning, F. H. T. Lin, and S. Jaimungal. Double deep q-learning for optimal execution. *arXiv preprint arXiv:1812.06600*, 2018.

G. Nuti, M. Mirghaemi, P. Treleaven, and C. Yingsaeree. Algorithmic trading. *Computer*, 44(11):61–69, 2011.

F. D. Paiva, R. T. N. Cardoso, G. P. Hanaoka, and W. M. Duarte. Decision-making for financial trading: A fusion approach of machine learning and portfolio selection. *Expert Systems with Applications*, 115:635–655, 2019.

M. Pemy, Q. Zhang, and G. G. Yin. Liquidation of a large block of stock with regime switching. *Mathematical Finance: An international Journal of Mathematics, Statistics and Financial Economics*, 18(4):629–648, 2008.

S. M. Ross. *Introduction to stochastic dynamic programming*. Academic press, 2014.

D. Sornette and S. von der Becke. Crashes and high frequency trading: An evaluation of risks posed by high-speed algorithmic trading. *The Future of Computer Trading in Financial Markets*, 2011.

T. Spooner, J. Fearnley, R. Savani, and A. Koukorinis. Market making via reinforcement learning. *arXiv preprint arXiv:1804.04216*, 2018.

A. Tampuu, T. Matiisen, D. Kodelja, I. Kuzovkin, K. Korjus, J. Aru, J. Aru, and R. Vicente. Multiagent cooperation and competition with deep reinforcement learning. *PloS one*, 12(4):e0172395, 2017.

T. Théate and D. Ernst. An application of deep reinforcement learning to algorithmic trading. *Expert Systems with Applications*, 173:114632, 2021.

S. Vyetrenko and S. Xu. Risk-sensitive compact decision trees for autonomous execution in presence of simulated market response. *arXiv preprint arXiv:1906.02312*, 2019.

Z. Xiong, X.-Y. Liu, S. Zhong, H. Yang, and A. Walid. Practical deep reinforcement learning approach for stock trading. *arXiv preprint arXiv:1811.07522*, 2018.

Y. Yang, R. Luo, M. Li, M. Zhou, W. Zhang, and J. Wang. Mean field multi-agent reinforcement learning. In J. Dy and A. Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 5571–5580. PMLR, 10–15 Jul 2018. URL `http://proceedings.mlr.press/v80/yang18d.html`.

C. Zhang, G. YIN, and Q. Zhang. Liquidation of a large block stock under a markov chain model. *SCIENTIA SINICA Mathematica*, 45(5):497–514, 2015.

Appendix A – Markov process and Dynamic programming

To introduce the optimal execution model formulation for liquidating a large number of stock, we should understand the theoretical concepts about the Markov decision process and dynamic programming, which are discussed in the following part:

Markov decision process

Markov decision process (MDP) is a fundamental framework for our model. Therefore, as a first step, the essential variables and definitions related to this framework are presented:

Markov process: A Markov process is a stochastic system with sequential steps, in which the future outcomes' probabilities are determined solely by their present state. Therefore, a process is Markov only and only if:

$$\begin{aligned}\mathbb{P}^{a}_{ss'} &= \mathbb{P}[S_{t+1} = s' | S_t = s, A_t = a] \\ &= \mathbb{P}[S_{t+1} = s' | S_t = s, S_{t-1} = s_{t-1}, ..., S_1 = s_1, S_0 = s_0, A_t = a] \quad (1)\end{aligned}$$

Where S refers to the state space of the stochastic process, A is the possible actions at each state, $\mathbb{P}^{a}_{ss'}$ corresponds to the transition probability matrix, and:

$$\mathbb{P}^{a}_{ss'} \geq 0; \quad \text{for all } a \in A, \quad s, s' \in S \quad (2)$$

$$\sum_{s' \in S} \mathbb{P}^{a}_{ss'} = 1 \quad (3)$$

Reward: A reward function R_s^a is the expected reward when we are at state s, and taking the action a:

$$R_s^a = \mathbb{E}[R_t|S_t = s, A_t = a] \tag{4}$$

Return: A return is the sum of discounted rewards from the current stage, until the end of the process.

$$F_t = \sum_{i=0}^{i=N} \gamma^i R_{t+i} \tag{5}$$

Where, $0 < \gamma \leq 1$ refers to the discount factor.

Therefore, the Markov decision process is defined as a set of (S, A, P, R, γ), which sequentially describes the current and future of the stochastic process.

Value and policy functions

Policy: A policy is a way of the decision-making process based on the state.

$$\pi(a|s) = \mathbb{P}[A_t|S_t = s] \tag{6}$$

State-Value function: A value function represents the expected return, when we are at state s, and given policy π.

$$v_\pi(s) = \mathbb{E}_\pi[F_t|S_t = s] \tag{7}$$

Action-Value function: An action-value function represents the expected return, when we are at state s, take the action a, and given policy π.

$$g_\pi(s, a) = \mathbb{E}_\pi[F_t|S_t = s, A_t = a] \tag{8}$$

It should be mentioned that for any MDP, there is an optimal policy π_*, which results in the optimal value function.

Bellman equation

The state-value and action-value functions can be represented by decomposing the return into the immediate reward and the expected value of the successive state. This type of value function representation is called Bellman's equation, and will be used in the iterative solutions.

$$v_\pi(s) = \mathbb{E}_\pi[R_t + \gamma v_\pi(S_{t+1})|S_t = s] \tag{9}$$

$$g_\pi(s,a) = \mathbb{E}_\pi[R_t + \gamma q_\pi(S_{t+1}, A_{t+1})|S_t = s, A_t = a] \tag{10}$$

Bellman's Principle of Optimality: The Bellman's principle of optimality states that whatever the initial state and action (decision) are, the remaining decisions of an optimal policy is an optimal policy with regard to the state that results from the first decisions in the absence of disturbances.

Dynamic programming

Dynamic Programming (DP) is defined as a solution algorithm for a complex optimization problem. Therefore, the problem is broken down into simpler sequential sub-problems. Then, each part will be solved accordingly.

According to Bellman's principle of optimality, the solution could be found for the optimal control problem with a known transition matrix and reward function. The decisions at each stage can be found by working either forward or backward at each stage.

Appendix B – Wrong parameters estimation impact on the market and execution price

As discussed in Chapter 2, the dynamic of the market and execution prices for the linear permanent and temporary trading impact functions are:

$$dS_t = -bv_t dt + \sigma dW_t, \tag{11}$$

$$\widehat{S}_t = S_t - (kv_t), \tag{12}$$

Moreover, according to the solutions for optimal liquidation, the dynamic of remaining inventory and the trading rate are:

$$v_t^* = a_1\left(\frac{a_2 e^{a_1(T-t)} + e^{-a_1(T-t)}}{a_2 e^{a_1(T-t)} - e^{-a_1(T-t)}}\right)Q_t^{v^*} \tag{13}$$

And:

$$Q_t^{v^*} = \left(\frac{a_2 e^{a_1(T-t)} - e^{-a_1(T-t)}}{a_2 e^{a_1 T} - e^{-a_1 T}}\right)Q_0 \tag{14}$$

Where a_1 and a_2 parameters are computed according to our assumptions in Equations 2.11 and 2.12.

Therefore:

$$v_t^* = a_1\left(\frac{a_2 e^{a_1(T-t)} + e^{-a_1(T-t)}}{a_2 e^{a_1(T-t)} - e^{-a_1(T-t)}}\right)\left(\frac{a_2 e^{a_1(T-t)} - e^{-a_1(T-t)}}{a_2 e^{a_1 T} - e^{-a_1 T}}\right)Q_0 \tag{15}$$

$$\rightarrow v_t^* = a_1\left(\frac{a_2 e^{a_1(T-t)} + e^{-a_1(T-t)}}{a_2 e^{a_1 T} - e^{-a_1 T}}\right)Q_0 \tag{16}$$

Let's assume the wrong value for any of initial parameters, then:

$$v_t' = a_1'\left(\frac{a_2' e^{a_1'(T-t)} + e^{-a_1'(T-t)}}{a_2' e^{a_1'(T-t)} - e^{-a_1'(T-t)}}\right)Q_t' \tag{17}$$

Where a_1' and a_2' are computed for the wrong initial parameters estimations.

And:

$$Q_t' = \left(\frac{a_2' e^{a_1'(T-t)} - e^{-a_1'(T-t)}}{a_2' e^{a_1 T} - e^{-a_1' T}}\right)Q_0 \tag{18}$$

Therefore:

$$v_t' = a_1'\left(\frac{a_2' e^{a_1'(T-t)} + e^{-a_1'(T-t)}}{a_2' e^{a_1' T} - e^{-a_1' T}}\right)Q_0 \tag{19}$$

Therefore, the difference between the market price when we assume an actual or a wrong initial parameters' values is:

$$dS_t = -bv_t^* dt + \sigma dW_t \quad dS_t' = -bv_t' dt + \sigma dW_t$$

$$\rightarrow S_t - S_t' = \int_0^t -b(v_u^* - v_u')\,du$$

$$\rightarrow S_t - S_t' = \int_0^u -b\left[a_1\left(\frac{a_2 e^{a_1(T-u)} + e^{-a_1(T-u)}}{a_2 e^{a_1 T} - e^{-a_1 T}}\right)Q_0 - a_1'\left(\frac{a_2' e^{a_1'(T-u)} + e^{-a_1'(T-u)}}{a_2' e^{a_1' T} - e^{-a_1' T}}\right)Q_0\right]du$$

$$\rightarrow S_t - S_t' = -bQ_0 a_1\left(\frac{\int_0^t a_2 e^{a_1(T-u)} + e^{-a_1(T-u)}\,du}{a_2 e^{a_1 T} - e^{-a_1 T}}\right) + bQ_0 a_1'\left(\frac{\int_0^t a_2' e^{a_1'(T-u)} + e^{-a_1'(T-u)}\,du}{a_2' e^{a_1' T} - e^{-a_1' T}}\right)$$

$$\rightarrow S_t - S_t' = -bQ_0\left(\frac{-a_2 e^{a_1(T-t)} + e^{-a_1(T-t)}}{a_2 e^{a_1 T} - e^{-a_1 T}} + 1\right) + bQ_0\left(\frac{-a_2' e^{a_1'(T-t)} + e^{-a_1'(T-t)}}{a_2' e^{a_1' T} - e^{-a_1' T}} + 1\right)$$

$$\rightarrow S_t - S_t' = -bQ_0\left(\frac{-a_2 e^{a_1(T-t)} + e^{-a_1(T-t)}}{a_2 e^{a_1 T} - e^{-a_1 T}}\right) + bQ_0\left(\frac{-a_2' e^{a_1'(T-t)} + e^{-a_1'(T-t)}}{a_2' e^{a_1' T} - e^{-a_1' T}}\right)$$

$$\rightarrow S_t - S_t' = b(Q_t^{v^*} - Q_t'), \quad 0 \le t \le T$$

It should be noted, the agent's assumptions for the initial parameters' values have no impact on their actual value. Also, at the terminal time, all the remaining inventory will be liquidated. Therefore, there is no market price difference between these two assumptions after the liquidation horizon:

$$v_T^* = Q_T^{v^*}, \quad v'_T = Q'_T \quad \rightarrow S_{T+1} - S'_{T+1} = 0$$

And, the difference between the execution price when we assume an actual or a wrong initial parameters' values is:

$$\begin{gathered}
\widehat{S_t} = S_t - (kv_t), \quad \widehat{S'_t} = S'_t - (kv'_t), \\
\rightarrow \widehat{S_t} - \widehat{S'_t} = (S_t - S'_t) - (kv_t - kv'_t) \\
\rightarrow \widehat{S_t} - \widehat{S'_t} = b(Q_t^{v^*} - Q'^{)}_t) - k(v_t - v'_t), \quad 0 \leq t < T
\end{gathered}$$

Furthermore, similar to the market price, the execution price difference between these two assumptions at the terminal time step is:

$$\begin{gathered}
\widehat{S_T} - \widehat{S'_T} = (bQ_T^{v^*} - \alpha Q_T^{v^*}) - (bQ'_T - \alpha Q'_T), \\
\rightarrow \widehat{S_T} - \widehat{S'_T} = (b - \alpha)(Q_T^{v^*} - Q'_T)
\end{gathered}$$