DEGREE PROJECT IN THE FIELD OF TECHNOLOGY
INFORMATION AND COMMUNICATION TECHNOLOGY
AND THE MAIN FIELD OF STUDY
COMPUTER SCIENCE AND ENGINEERING,
SECOND CYCLE, 30 CREDITS
STOCKHOLM, SWEDEN 2018

Simulating market maker behavior using Deep Reinforcement Learning to understand market microstructure

MARCUS ELWIN

KTH ROYAL INSTITUTE OF TECHNOLOGY
SCHOOL OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

Simulating market maker behaviour using Deep Reinforcement Learning to understand market microstructure

MARCUS ELWIN

Master of Science in Machine Learning
Date: January 1, 2019
Supervisor: Hamid Reza Faragardi
Examiner: Elena Troubitsyna
Swedish title: En simulering av aktiemarknadens mikrostruktur via självlärande finansiella agenter
School of Electrical Engineering and Computer Science

Abstract

Market microstructure studies the process of exchanging assets under explicit trading rules. With algorithmic trading and high-frequency trading, modern financial markets have seen profound changes in market microstructure in the last 5 to 10 years. As a result, previously established methods in the field of market microstructure becomes often faulty or insufficient. Machine learning and, in particular, reinforcement learning has become more ubiquitous in both finance and other fields today with applications in *trading* and *optimal execution*. This thesis uses reinforcement learning to understand market microstructure by simulating a stock market based on NASDAQ Nordics and training market maker agents on this stock market.

Simulations are run on both a *dealer market* and a *limit orderbook market* differentiating it from previous studies. Using DQN and PPO algorithms on these simulated environments, where *stochastic optimal control theory* has been mainly used before. The market maker agents successfully reproduce *stylized facts* in historical trade data from each simulation, such as *mean reverting prices* and *absence of linear autocorrelations* in price changes as well as beating random policies employed on these markets with a positive profit & loss of maximum 200%. Other trading dynamics in real-world markets have also been exhibited via the agents interactions, mainly: *bid-ask spread clustering, optimal inventory management, declining spreads* and *independence of inventory and spreads*, indicating that using reinforcement learning with PPO and DQN are relevant choices when modelling market microstructure.

Sammanfattning

Marknadens mikrostruktur studerar hur utbytet av finansiella tillgångar sker enligt explicita regler. Algoritmisk och högfrekvenshandel har förändrat moderna finansmarknaders strukturer under de senaste 5 till 10 åren. Detta har även påverkat pålitligheten hos tidigare använda metoder från exempelvis ekonometri för att studera marknadens mikrostruktur. Maskininlärning och *Reinforcement Learning* har blivit mer populära, med många olika användningsområden både inom finans och andra fält. Inom finansfältet har dessa typer av metoder använts främst inom handel och optimal exekvering av ordrar. I denna uppsats kombineras både *Reinforcement Learning* och marknadens mikrostruktur, för att simulera en aktiemarknad baserad på NASDAQ i Norden. Där tränas *market maker* - agenter via *Reinforcement Learning* med målet att förstå marknadens mikrostruktur som uppstår via agenternas interaktioner.

I denna uppsats utvärderas och testas agenterna på en *dealer* - marknad tillsammans med en *limit* - orderbok. Vilket särskiljer denna studie tillsammans med de två algoritmerna DQN och PPO från tidigare studier. Främst har stokastisk optimering använts för liknande problem i tidigare studier. Agenterna lyckas framgångsrikt med att återskapa egenskaper hos finansiella tidsserier som återgång till medelvärdet och avsaknad av linjär autokorrelation. Agenterna lyckas också med att vinna över slumpmässiga strategier, med maximal vinst på 200%. Slutgiltigen lyckas även agenterna med att visa annan handelsdynamik som förväntas ske på en verklig marknad. Huvudsakligen: kluster av *spreads*, optimal hantering av aktielager och en minskning av *spreads* under simuleringarna. Detta visar att *Reinforcement Learning* med PPO eller DQN är relevanta val vid modellering av marknadens mikrostruktur.

Acknowledgements

I like would like to express my gratitude to my thesis advisor Hamid Reza Faragardi as well as my examiner Elena Troubitsyna. For both your support and helpful comments throughout the course of the thesis. I am profoundly grateful to Björn Hertzberg at Nasdaq for introducing me to both the field of market microstructure and the thesis subject. My research would have been impossible without the aid and support of Jaakko Valli at Nasdaq. Finally I would like to thank my family and girlfriend for having patience with me during this long process.

Contents

List of Figures

List of Tables

Listings

Chapter 1

Introduction

1.1 Background

Modern financial markets such as NASDAQ, CME and NYSE have all been effected by the rise and presence of *Algorithmic Trading* and *High-Frequency Trading* (HFT) [2], which are causing, for instance, more fragmented markets. Both types of trading consist of using computer programs to implement investment and trading strategies [1]. These strategies have, according to Abergel [1] and O'Hara [49], raised various questions about their effects on the financial markets, mainly in such areas as: *liquidity, volatility, price discovery, systematic risk, manipulation* and *market organization*. A quite recent example of the proposed effect of algorithmic trading and HFT on financial markets is *the Flash Crash* the 6th of May 2010. In the course of 30 minutes, U.S. stock market indices, stock-index futures, options, and exchange-traded funds, experienced a sudden price drop of more than five percent, followed by a rapid rebound [38, 37] (see an illustration of this in Figure 1.1).

Trading in the financial market can be seen as a search problem where buyers and sellers search for each-other, depending on market structure [1]. *Market microstructure* is a branch of economics, where one tries to understand trading dynamics on the financial market on a microscopic level [50, 30]. Market microstructure theory is used by regulators, traders and organizers of financial markets, in order to make a profit or create more transparent and efficient markets. Recent regulation aiming at making markets more transparent and efficient is *Markets in Financial Instruments Directive* (MiFID) II [10].

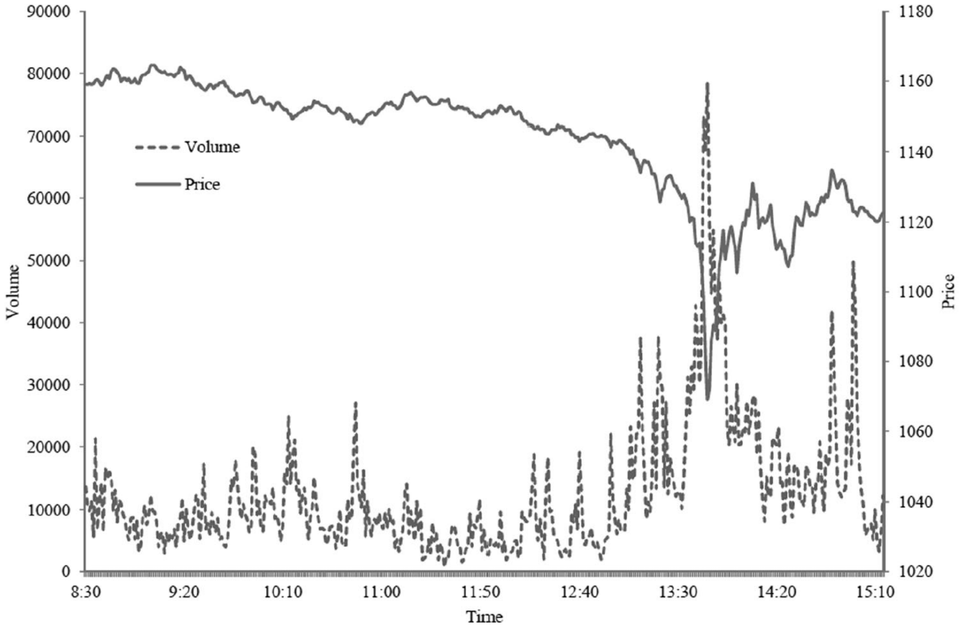


Figure 1.1: Minute-by-minute transaction prices and trading volume on E-mini S& P futures contracts during the flash crash, between 8:30 to 15:15. Notice the distinct drop and rebound at the end of the day. Source: Kirilenko et al. [37]

However, as mentioned by O'Hara [49], due to HFT and algorithmic trading, learning models and empirical models used in market microstructure in the past are deficient and may no longer be appropriate. It calls for the use of new more capable methods. In this respect, *Reinforcement Learning* and other machine learning methods are of great interest. Machine learning and AI have become ubiquitous in finance (see for example [47, 65, 20, 22, 6, 14]), where the main reasons for this are the abundance of available data and computing power.

Recent achievements in the use of Reinforcement Learning has been seen in the game Go. *AlphaGo* and *AlphaGoZero* [59] are programs that were able to win over esteemed Go champions, using reinforcement learning. More complicated strategic games, as Star Craft have also seen successful application of reinforcement learning [64]. Therefore, the focus of this thesis will be to examine the usages of deep reinforcement learning, to understand the market microstructure of a simulated Nordic stock market.

1.2 Problem

Due to the abundance of available high frequency data, fragmented markets and more sophisticated trading algorithms, the financial markets have become harder to understand during the past decade. Traditional methods used in market microstructure might have become obsolete as mentioned in O'Hara [49]. A traditional supervised learning approach is not of use for this thesis, because financial markets are dynamic complex systems. The agents must be able to adapt and dynamically learn optimal behaviour. Therefore, the problem in this thesis is to understand trading dynamics, by using reinforcement learning, on a simulated Nordic stock market.

The thesis will study whether common market microstructure trading dynamics, such as *bid-ask spread clustering, optimal trade execution* and *optimal inventory costs* as described in O'hara [50], will be exhibited and understood by the reinforcement learning agents. Also, it is of interest to see what happens with the agents when changing the market conditions, much like what is happening in the real markets. For example, it can be explored by changing the number of participants, order sizes, prices, volatility, order arrival rates and trading rules.

1.3 Objective

The objective of this thesis is two-fold. Firstly, in the case of the principal the objective is to have a functional and working exchange simulator (EXSIM), where they can change different parameters, policies, reward functions and other things effecting the market structure, in order to study and simulate modern financial markets on a microscopic level. Secondly, from the thesis point of view, the objective is to investigate the following:

- Whether reinforcement learning can be successfully used in a more complicated environment such as the financial market.
- Whether new insights and applications can be provided to the the growing field of market microstructure with the help of reinforcement learning.

1.4 Delimitation

This thesis does not use any real-world data for the simulations and experiments conducted. Instead, only simulated data from the agent's interaction with its environment is used. This is due to security constraints at the principle. No multiagent reinforcement learning has been used, due to time constraints. Instead, several different single agents are tested and evaluated.

Information-based market microstructure models have not been used in this thesis. Instead, inventory-based models and limit order book models have been used, mainly due to time constraints. Exploration of distributed and parallel training of agents to speed up training, using workers and a parameter server have not been performed. This is also due to time constraints and could be something to look at for future work.

1.5 Methodology

The methodology in the thesis is empirical with the focus on the quantitative approach. Data are collected from the agent's interactions with the environments through various experiments testing different scenarios, behaviours and collecting different statistics. More information about the methodology is found in chapter 4 and chapter 5.

1.6 Contribution

The contribution of this thesis is to show the usages of reinforcement learning to more complex environments such as the financial markets with a more realistic limit order book market compared to previous studies. The aim is to provide new insights and methods in the field of market microstructure. The target audience is both academia and the industry, which can benefit from this work.

1.7 Societal and sustainability issues

Financial crises have negative impact on society, where the credit-crisis in 2007-2008 is a clear example of this. Hence, studying market microstructure is important to prevent them.

1.8 Outline of the thesis

The thesis has the following structure:

- In chapter 2, the theoretical background needed for the thesis is presented. It covers *market microstructure, artificial financial markets* and *reinforcement learning*.
- In chapter 3, related work relevant to the thesis is presented. This chapter presents the state-of-the art, and the contribution of this thesis, in comparison with the previous research.
- In chapter 4, an overview of the research methodology employed for this thesis is presented.
- In chapter 5, the implementation of the agents, environments, data collection and the experiments are discussed in detail.
- In chapter 6, the results obtained from the different experiments are illustrated and discussed.
- In chapter 7, the main findings are highlighted and evaluated and compared to previous research. The conclusions of the thesis and future research are also presented.

Chapter 2

Background

This chapter presents relevant theory needed to contextualize the thesis. The following is covered: *market microstructure theory, artificial financial markets* and finally *reinforcement learning*.

2.1 Market Microstructure

Market microstructure is the study of the process and outcomes of exchanging assets under implicit trading rules as mentioned in O'hara [50]. The majority of market microstructure research is according to Madhavan [44] concerned with :

1. *Price formation and discovery*, i.e., looking in to the black box of the market and see how latent demands are translated into prices.

2. *Market structure and design*, i.e., what rules exists, and how they affect the black box of the market.

3. *Information and disclosure*, i.e., how the inner workings of the black box or the market affects the behaviour of traders and strategies.

All of these will be covered in this section, in order to give the reader a comprehensive overview, on how modern financial markets operate. However, for the interested reader more details, regarding market microstructure can be found in O'hara [50], Harris [29], Bouchaud et al. [8], Abrol, Chesir, and Mehta [2], and Abergel [1].

2.1.1 Market Participants

A *market* is the place where traders gather to trade [29] different types of instruments: stocks, bonds, futures, options, derivatives and foreign exchange rates just to mention a few. In today's financial markets, Cartea, Jaimungal, and Penalva [13] broadly identify three primary classes of traders (or strategies) that partake in the market:

- **Fundamental traders:** those who are driven by economic fundamentals outside the exchange.
- **Informed traders:** traders who profit from leveraging information not reflected in market prices and trading assets in the hope of their increase or decrease in value.
- **Market Makers:** professional traders who profit from facilitating the exchange in a particular asset and exploit their skills in executing trades.

Market makers will be the main participant covered in this thesis. This because they have an important impact on providing liquidity to financial markets, albeit the dual nature of optimizing their inventory of stocks and making a profit for themselves.

2.1.2 Trading Mechanisms

Any trading mechanism can be seen as a type of trading game in which players meet virtually or physical at some venue and act according to some rules [50]. The players are some of the participants mentioned in the previous section. The venue or the market is where trades are actually executed, which can be on an exchange or via other intermediaries (often called OTC or off exchange). A common division of market structure is presented in Foucault, Pagano, and Röell [24], where they classify markets as either *limit order markets* (auction markets) or *dealer markets*. In fact, all trading mechanisms can be viewed as variations of these [24]. Today most markets are electronic, and adopt a *continuous-time double auction* mechanism using a limit orderbook [8]. Compared to *Walrasian auctions*[1], in limit order markets the final investors interact directly instead.

[1] Traders communicate their buying and selling intentions via an auctioneer.

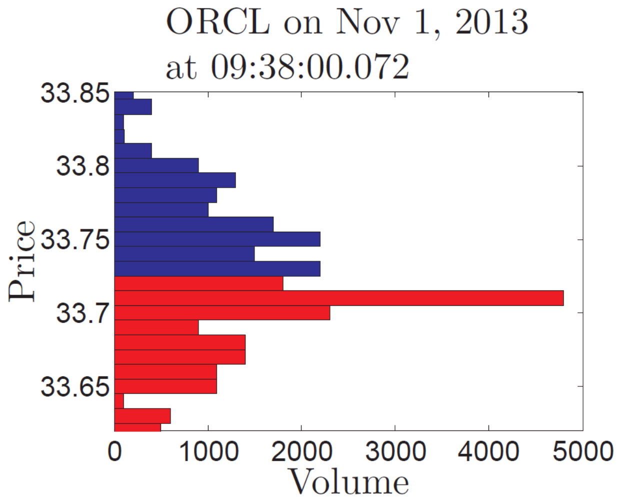


Figure 2.1: Snapshot of the LOB for the ticker ORCL (Oracle) after the 10 000th event during that day. Blue bars indicate sell limit orders, whilst red bars are buy limit orders. Source: Cartea, Jaimungal, and Penalva [13]

In the limit order market, bids and offers are accumulated in the *limit order book* (LOB). The orders in the LOB are accumulated by firstly price priority and secondly time priority [30]. In dealer markets, participants can only trade at the bid and ask quotes posted by specialized intermediaries, i.e., *dealers* or *market makers* [24]. Note that the LOB is very dynamic as it consists of *limit orders*[2], which can be cancelled or modified at any time. Thus, the state of the LOB can be changed extremely often [30]. For this thesis this means that the agents will have a very large state space, with many possible actions that they need to explore, in order to find optimal policies. More formally, according to Bouchaud et al. [8], one can see an order as a tuple consisting of sign/direction (ε_x), price (p_x), volume (v_x) and submission time (t_x):

$$x = (\varepsilon_x, p_x, v_x, t_x) \tag{2.1}$$

where $\varepsilon = \pm 1$ indicates if it is a buy or sell order and $v_x > 0$. An example of a limit order book is shown in Figure 2.1 for the stock ORCL. Market orders are other types of orders found on LOB markets, which are usually considered to be more aggressive because, this type of orders seek to execute a trade immediately [13], compared to limit orders that wants to be executed at a certain limit.

[2] A limit order is an order that specifies a direction, quantity and acceptable price.

For instance, if a market order is placed and the quantity is larger than the quantity available in the book then the order is *re-routed* or said to *walk the book* until the order is filled [30, 13]. When submitting an order x, the trader must chose the size v_x and price p_x according to a relevant *lot size* and *tick size* of the LOB [8]. The lot size v_0 is the smallest amount of the asset that can be traded. Hence the size of each order is a multiple of the lot size $v_x \in \{kv_0 | k = 1, 2, ...\}$. The tick size ϑ is the smallest permissible price interval between different orders within a given LOB [8]. The values of v_0 and ϑ differ a lot between exchanges. However, expensive stocks are often traded with $v_0 = 1$ whilst cheaper stocks are traded with $v_0 \gg 1$. In equity markets, ϑ is often 0.01% of the stock's mid-price [8].

Both the tick size and lot size affect trading, where the lot size dictates the smallest permissible order size. The tick size ϑ dictates how much more expensive it is for a trader to gain the priority of choosing a higher or lower price to a buy or sell order [8]. Sometimes it is also useful to consider the *relative tick size* ϑ_r, which is equal to the ϑ divided by the mid-price for a given asset [8]. To make things more complicated modern markets have a lot of different order types. Such as *hidden, reserved, ice-burg* and *Fill-or-Kill* orders just to mention a few, where [24, 13, 30] gives good overviews of these. There also exist *hybrid markets* which are traditional quote-driven markets such as NASDAQ and London Stock Exchange (LSE) [24].

2.1.3 Price Formation & Discovery

The mechanism of price formation is at the very heart of economics, which is also important in order to understand *stylized facts* in financial price series such as *heavy tails* and *volatility clustering* [1]. Price discovery is the speed and accuracy with which transactions prices incorporate information available to market participants [24]. Thus, investigating market makers is a logical starting point for understanding how prices are actually determined in the market [44]. Nevertheless, what is not covered in this thesis and still important is the role of information and who has it [13].

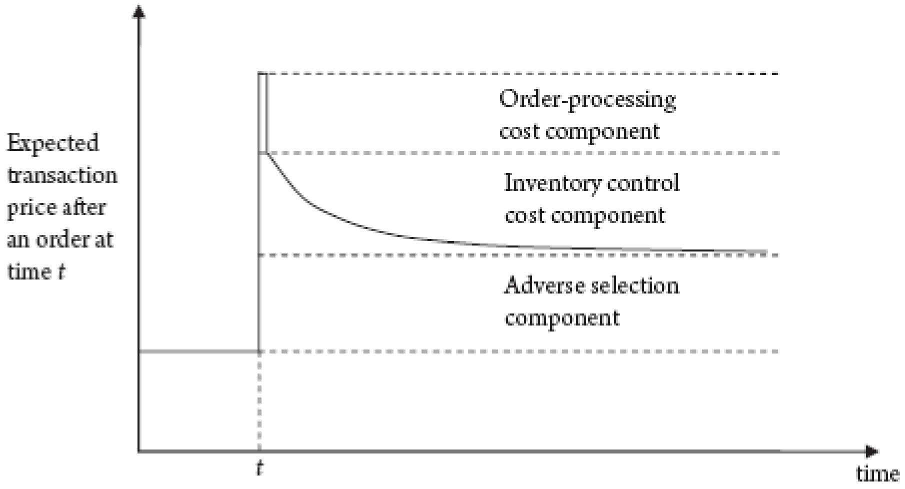

Expected transaction price after an order at time t
Order-processing cost component
Inventory control cost component
Adverse selection component
t
time

Figure 2.2: Three components of bid-ask spread in short-term and long-term response to a market buy order. Source: Foucault, Pagano, and Röell [24]

Bid-Ask Spread. The bid-ask spread is usually decomposed into three components: *adverse selection, order-processing costs* and *inventory holding costs* [24]. Order processing costs consist of the setup price and operating costs of trading. Inventory costs are costs associated with carrying inventory. Adverse selection costs are costs that arise because some traders are more informed then others, and when trading with these *informed traders*, market makers will on average lose money [17]. Therefore, a fraction of the bid-ask spread can be seen as a compensation for having to trade against informed traders [17]. In Figure 2.2 above these are illustrated on a short-term and long-term perspective.

Market makers quote two prices: the bid price and the ask price, where the difference between these is the market makers *spread* [44]. By quoting bid and ask prices, market makers are also providing *liquidity* to the market. Spreads measure the execution cost of a small transaction, by measuring how close the price of a trade is to the *market price*, where the market price is the *equilibrium price*, i.e., the price where demand equals supply [13]. One approach is by using the *midprice* in Equation 2.2:

$$S_t = \frac{1}{2}(a_t + b_t) \tag{2.2}$$

which is the simple average of the bid (b_t) and ask (a_t) prices at time t.

However, the most common spread measures are the *quoted* and the *effective* [13, 24] spread both shown in Equation 2.3 and Equation 2.4:

$$QS_t = a_t - b_t \tag{2.3}$$

$$ES_t = a_t - S_t \text{ or } ES_t = S_t - b_t \tag{2.4}$$

The quoted spread represents the potential cost of immediacy at any point in time as well as the distance from the market price [13]. As mentioned in Foucault, Pagano, and Röell [24] the quoted spread is also a good measure of trading costs for small orders used for measuring liquidity. By normalizing Equation 2.3 with the midprice, the *relative quoted spread* is obtained:

$$RQS_t = \frac{a_t - b_t}{S_t}.$$

On the contrary, the effective spread or half-spread measures the realized difference between the price paid and the midprice, which can also be negative indicating that one is buying at a price below or selling above the *market price* [13]. ES and QS differ in the fact that ES can only be measured when there is a trade while QS are always observable [13]. Some stylized facts known about the bid-ask spread are [30, 43, 8]:

- The trade prices series is a martingale & the order flow is not symmetric.
- The spread declines over time & the bid-ask spread are lower in high volume securities and wider for more riskier securities.
- For large-tick stocks, the spread is almost equal to one tick. Small-tick stocks have a broader distribution of spreads.
- There is a price impact of trades, i.e., on average the arrival of a buy trade causes prices to rise (S_t increases), whilst the arrival of sell trades causes prices to fall (S_t decreases).

Liquidity. If the structure of a securities market is compared to a car design, measuring liquidity is like assessing the car's driving performance [24]. Liquidity impounds the usual economic concept of *elasticity*[3]. In a liquid market, a small shift in supply and demand does not result in large price changes [30]. However, liquidity is also concerned with trading costs. Market makers are seen as liquidity providers (sell-side) whilst liquidity demanders are the customers (buy-side) [30]. A key dimension of liquidity is *immediacy*. This is the ability of an investor to buy or sell an asset without having to wait to find a counterpart with an offsetting position to sell or buy [13]. In fact, the bid-ask spread is a common measure of how liquid a market is [24].

Order imbalance. Another way of describing price dynamics is to see the accumulated best bid and ask quotes in a LOB as *queues* [8]. All price changes then boil down to a *race* between the different queues. The queue that depletes first, i.e., the winner, dictates the next price change [8]. Order imbalance is a measure providing a quantitative assessment of the relative strengths of buying and selling pressure in the LOB [8]. Volume order imbalance is defined as:

$$I_t = \frac{V_t^b}{V_t^b + V_t^a} \tag{2.5}$$

where V_t^b is the bid volume at time t, conversely V_t^a is the ask volume at time t. The denominator (total volume at time t) in Equation 2.5 normalizes the imbalance, therefore $I_t \in [0, 1]$. Bouchaud et al. [8] provide a qualitative understanding of I_t:

- $I_t \approx 0$ corresponds to a situation where the ask-queue is much larger than the bid-queue, meaning that there is a net positive selling pressure in the LOB, likely pushing the price downwards.

- $I_t \approx \frac{1}{2}$ means that the bid- and ask-queues are approximately equally big, meaning that the buying and selling pressures in the LOB are somewhat balanced.

- $I_t \approx 1$ corresponds to a situation where the bid-queue is much larger than the ask-queue, meaning that there is a net positive buying pressure in the LOB, likely pushing the price upwards.

[3] A measure of the responsiveness of one economic variable to another.

An estimate of the probability $p_+(I_t)$ that the ask-queue depletes before the bid-queue has been shown in [8] to be:

$$p_+(I_t) = p_+(V_t^b, V_t^a) = \frac{2}{\pi}\arctan\left(\frac{I_t}{1-I_t}\right) \tag{2.6}$$

note that one can replace $\frac{I_t}{1-I_t} = \frac{V_t^b}{V_t^a}$. Another imbalance measure according to Cartea, Jaimungal, and Penalva [13] is *limit order balance* defined in Equation 2.7:

$$\rho_t = \frac{V_t^b - V_t^a}{V_t^b + V_t^a} \tag{2.7}$$

this measure takes values for $\rho_t \in [-1, 1]$. Usually one computes ρ_t by looking only *at-the-touch*, the best n-levels of the LOB. The first level is the best price, followed by the second price level and so forth. There are more buy orders when the imbalance is high, and there are more sell orders when the imbalance is low. The willingness of an agent to post limit orders is strongly dependent on the value of imbalance [13].

Price Impact. A concern for participants that wish to execute large orders is that they will have an adverse *price impact*. Meaning that an agent will be increasing the price when buying aggressively and lowering it when selling [13, 8]. A way of measuring this is by running a regression on the change of the midprice of the form below:

$$\Delta S_n = \lambda q_n + \varepsilon_n \tag{2.8}$$

here $\Delta S_n = S_{n\tau} - S_{(n-1)\tau}$ for a time interval $[(n-1)\tau, n\tau]$ [13]. Note that other forms of this regression exist including inventory (see Foucault, Pagano, and Röell [24] for more information). Using Equation 2.8 it is possible to estimate λ, where the parameter λ, also called *Kyle's lambda* [8] is capturing the market's price reaction, i.e., its price impact. On the other hand, q_n is the order imbalance or net order flow [4] and ε_n the error term assumed to be normally distributed [13, 24].

In terms of measuring liquidity, a lower λ indicates that the market is more liquid due to greater competition, lower risk tolerance or lower volatility [13]. Thus a lower λ means that prices are less sensitive to order imbalance [24]. A larger λ indicates that the given volume impacts the prices, and trading is thus more expensive [8].

[4] Difference between buy and sell orders during an interval.

2.1.4 Inventory-based models

A large (positive) inventory causes the dealer or market maker to face a higher cost for observing more inventory, which lowers both bid and ask prices by the same amount [50]. Vice versa situation holds for negative inventory. The model by Ho and Stoll [33] is one of the most popular inventory-based models, presented below.

Ho & Stoll Model. Ho and Stoll [33], present a model that handles the risk which the market maker faces when providing his service. In the model the following assumptions are made: *transactions follow a Poisson process, the dealer faces uncertainty over the future* and, *the arrival rate of orders depend on the bid and ask prices*. The objective of the dealer is now to maximize the expected utility of his total wealth $E[U(W_T)]$ at time horizon T, where:

$$W_T = F_T + I_T + Y_T \tag{2.9}$$

Equation 2.9 is called *the dealers pricing problem*. F_T, I_T and Y_T is the dealers cash account, inventory and base wealth, this is, in fact, an optimization problem where the aim is to maximize the value function $J(\cdot)$ using dynamic programming. Thus, yielding the optimization problem below in Equation 2.10:

$$J(t, F, I, Y) = \max_{a,b}[E[U(W_T)]|t, F, I, Y] \tag{2.10}$$

where U is the utility function, a and b are the ask and bid adjustments and t, F, I, Y are the state variables time, cash, inventory and base wealth [50]. The function $J(\cdot)$ gives the level of utility given that the dealer's decisions are made optimally [50]. There is no intermediate consumption before time T in this model. The recurrence relation found by using the principle of optimality is:

$$\max_{a,b} dJ(t, F, I, Y) = 0 \text{ and } J(T, F, I, Y) = U(W_T) \tag{2.11}$$

Solving Equation 2.11 one finds a solution to the dealer's problem, which requires stochastic calculus. For further elaboration and more details see Appendix C. There is no closed form solution for this problem. However, via approximations the bid and ask quotes have been shown to be found by:

$$b^* = \alpha/2\beta + (J - BJ)/2BJ_FQ \tag{2.12}$$

$$a^* = \alpha/2\beta + (J - SJ)/2SJ_FQ \tag{2.13}$$

Finally, from Equation 2.12 and Equation 2.13 one gets the bid-ask spread as:

$$s = \alpha/\beta + (J - SJ)/2SJ_FQ + (J - BJ)/2BJ_FQ \tag{2.14}$$

The first term of Equation 2.14 is the spread which maximizes the expected returns from selling and buying stocks. The rest of the terms are seen as *risk premiums* for sale and purchase transactions. This shows how the dealer or market maker sets the spread without knowing what side the transaction will have, i.e., bid or ask [33].

Ho and Stoll [33] demonstrates three important properties of the dealer's optimal pricing behavior:

1. The spreads depends on the time horizon of the dealer.
2. It can be decomposed in a risk-neutral and risky part.
3. The spread is independent of inventory levels.

Inventory based models are just one set of models used in the market microstructure literature, which is the most relevant one for this thesis. However, another important family is *information-based models*, allowing for examination of market dynamics, thus providing insights into the adjustment process of prices [50]. For popular models see more, in, for instance, Glosten and Milgrom [25] or Das* [18] and Das [19].

2.2 Artificial Financial Markets

Agent based modelling. In agent-based modelling (ABM), appropriate parts of a complex system are modeled as autonomous decision-making entities called *agents* [17]. In ABM market investors or traders are modelled as agents trading together via an orderbook [42]. However, the decision of what type of agents to use is paramount. Where, a wide range of different types of agents exist from *zero-intelligence agents* to *reinforcement learning agents* [45]. Broadly speaking there is a price disagreement between the agents, and often pricing is done randomly (also called *noise traders*), or via some real-world strategy [42]. According to, Martinez-Jaramillo and Tsang [45] some important design issues to think about are:

- *Decision making*, i.e., is it rule based, or based on something else.
- *Objective function*, i.e., explicit, implicit, utility or profit maximization.
- *Heterogeneity*, i.e., types of agents, parameters, information basis and learning.
- *Learning*, i.e., zero intelligence or more complex, as reinforcement learning.

Validation is an important issue in agent-based modelling, where simulated markets should be able to replicate realistic quantitative features of the real market with reasonable calibration [45]. In order to validate, multiple parameters need to be user-defined. However, one way to overcome this, is by using a benchmark in which the behaviour of the market is well defined. Another way according to Martinez-Jaramillo and Tsang [45] is to use parameters in the simulated market derived from experimental or real markets. It is also common to test whether the prices created via the interaction of the agents exhibit certain *stylized facts*, that real-world markets do [9]. With stylized facts one means a set of properties that is common across many instruments, markets and times, which have been observed by independent studies [16].

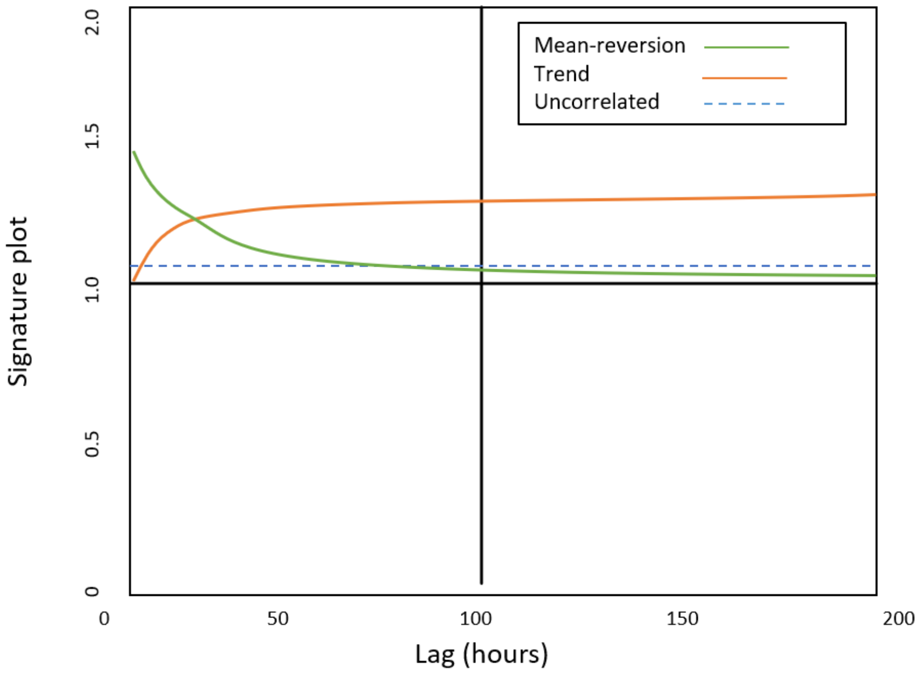


Figure 2.3: Example of volatility signature plots based on [8], showing expected pattern for mean reverting, trend and uncorrelated returns.

Stylized facts. In Cont [16] a large selection of stylized facts are presented. If $S(t)$ is the price of a financial asset, and $X(t) = \ln S(t)$ in time scale Δt, then the log-return of the asset can be defined in Equation 2.15:

$$r(t, \Delta t) = X(t + \Delta t) - X(t) \tag{2.15}$$

The time series in Equation 2.15 can then be used to study certain properties of financial time series that seems to hold in many independent studies, which should hold in realistic simulations of financial markets as well. Therefore, it is of importance for this thesis to be able to replicate some of these stylized facts, to validate the simulations. For a full list and further definitions see, for instance, Cont [16] and Bouchaud et al. [8]. The most important stylized facts for this study found in Cont [16] and Boer-Sorban [7] are:

- *Correlated Returns*, i.e., as prices are assumed to follow a *Gaussian random walk*, where the price returns for financial time series can exhibit patterns shown in Figure 2.3.

This shows a *signature plot* of the sampled volatility $\sigma(\tau)$ of the price returns against a time period, (τ). Many financial time series exhibit either strong or weakly *mean-reverting* behaviour [16, 8].

- *Intermittency*, i.e., returns display at any time scale a high degree of variability.
- *Absence of autocorrelations*, i.e., linear autocorrelations are often insignificant.
- *Non Gaussian returns & Heavy tails*, i.e., the unconditional distribution of returns seems to display a power-law or Pareto-like tail.
- *Volatility clustering*.

2.3 Reinforcement Learning

Reinforcement learning (RL) is learning what to do, i.e., mapping situations to actions, in order to maximize a numerical reward function [61]. It is quite different from other machine learning methods based on *supervised* or *unsupervised* learning, where one in fact have the true labels or not.

2.3.1 The Main Concepts

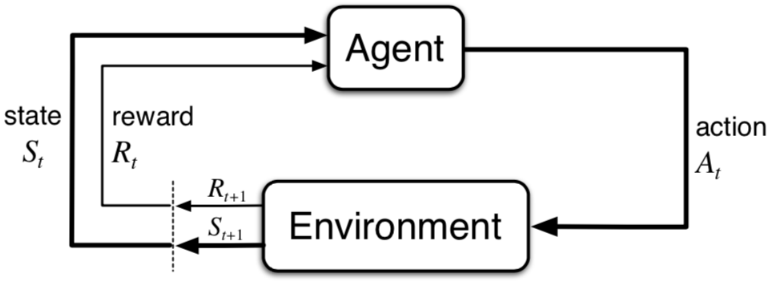


Figure 2.4: Basic overview of the reinforcement learning setting with an agent interacting via actions (A_t) with its environment moving through states (S_t), and gaining different rewards (R_t). Source: Sutton and Barto [61]

Simply put, RL is an agent interacting via actions with its environment, and by doing so eventually learning an optimal policy or behaviour. This learning process is done by trial and error for sequential decision making [41]. Figure 2.4, illustrates a simple agent interacting with its environment. A RL agent interacts with its environment over time, and at, each time step t the agent receives a *state* S_t in a state space $\mathcal{S}$ and makes an *action* A_t from an action space $\mathcal{A}$ [41]. As a consequence of its action, the agent receives *reward* R_t which is a scalar value.

The agent's goal is to maximize the total amount of cumulative reward it receives, which is known as the *reward hypothesis* [61]. The agent's behavior is modelled by the *policy* $\pi(s|a)$. A policy is a mapping from a state to an action, that is the probability of selecting action $A_t = a$ in state $S_t = s$. This also includes transitioning to the next state S_{t+1} according to the environments dynamics or *model* for a *reward function* $\mathcal{R}(s, a)$ and *state transition probability* $\mathcal{P}(S_{t+1}|S_t, A_t)$ [41].

Markov Decision Process. More formally the RL problem is formulated as a *Markov Decision Process* (MPD) [61, 41]. A MDP is a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$ [41] where:

- $\mathcal{S}$ is a finite set of states
- $\mathcal{A}$ is a finite set of actions
- $\mathcal{P}$ is a transition probability matrix,

$$\mathcal{P}^a_{ss'} = P[S_{t+1} = s'|S_t = s, A_t = a] \tag{2.16}$$

- $\mathcal{R}$ is a reward function,

$$\mathcal{R}^a_s = E[R_{t+1}|S_t = s, A_t = a] \tag{2.17}$$

- $\gamma \in [0, 1]$ is a *discount factor*

In general, one seeks to maximize the *expected return*, where the return, denoted G_t, is the total discounted reward from time step t:

$$G_t = R_{t+1} + \gamma R_{t+2} + ... = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \tag{2.18}$$

The discount factor or discount rate in Equation 2.18 determines the present value of future rewards k steps in the future.

Note that a γ close to 0 leads to *myopic* behavior, i.e., the agent only cares about immediate rewards. If γ is close to 1, the agent is more *far-sighted*. Then the agent-environment interaction breaks naturally into sub-sequences, also called *episodes*, which explains Equation 2.18. Another key concept underlying RL is the *Markov property*, i.e., only the current state affects the next state [4]. More formally this means:

$$P[S_{t+1}|S_t] = P[S_{t+1}|S_1, ..., S_t] \tag{2.19}$$

Equation 2.19 states that the future is conditionally independent of the past given the present state. According to, Arulkumaran et al. [4] this assumption is somewhat unrealistic because it requires the states to be fully observable. A generalization of a MDPs are partially observable MDPS (POMDPS), in which the agent receives an observation $O_t \in \mathcal{O}$. The distribution of the observation is $P(O_{t+1}|S_{t+1}|A_t)$[4], which is dependent on the current state and the previous action. The problem explored in this thesis is a POMDPS, as the agent observes, for instance, prices, volumes and other scalar values.

Value Functions. Most RL algorithms involve estimating *value functions* of states or state-action pairs. These estimate how good it is for an agent to be in a certain state [61, 41]. The value of a state s under a policy π is denoted $v_\pi(s)$. This is the expected return when starting in s and following π. Below is the *state-value function for policy* π:

$$v_\pi(s) = E_\pi[G_t|S_t = s] = E_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1}|S_t = s\right], \forall s \in \mathcal{S}. \tag{2.20}$$

Note that the value of the terminal state is always zero [61]. Similarly one can define the *action-value function for policy* π [61], which is the value of taking action a in state s under policy π, denoted $q_\pi(s, a)$. This is the expected return starting from s, taking the action a and therefore following policy π:

$$q_\pi = E_\pi[G_t|S_t = s, A_t = a] = E_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1}|S_t = s, A_t = a\right] \tag{2.21}$$

Equation 2.20 and Equation 2.21 can be estimated from experiences or by using *Monte Carlo Methods* [61].

Bellman equations & Optimality. Both Equation 2.20 and Equation 2.21 satisfies recursive relationships, which are commonly known as *Bellman equations* [61]:

$$v_\pi = E[G_t|S_t = s] = E_\pi[R_{t+1} + \gamma G_{t+1}|S_t = s] \tag{2.22}$$

$$q_\pi = E_\pi[R_{t+1} + \gamma G_{t+1}|S_t = s, A_t = a] \tag{2.23}$$

From Equation 2.22 and Equation 2.23 the Bellman equations expresses a relationship between the value of the state and its successor states. However, solving a RL task implies finding a policy that achieves a lot of reward over the long run, i.e., looking for *optimal policies* ($\pi \geq \pi'$). It is common to denote all optimal policies with π_*. They share the same state-value and action-value functions [61]. Therefore, it is of interest to maximize the following:

$$v_*(s) = \max_\pi v_\pi(s) \tag{2.24}$$

$$q_*(s,a) = \max_\pi q_\pi(s,a) \tag{2.25}$$

Using Equation 2.24 and Equation 2.25 together with the Bellman equations in Equation 2.22 and Equation 2.23 yields the *Bellman optimality equations*:

$$\begin{aligned} v_*(s) &= \max_a E_{\pi_*}[R_{t+1} + \gamma v_*(S_{t+1})|S_t = s, A_t = a] \\ &= \max_a \sum_{s',r} p(s',r|s,a)[r + \gamma v_*(s')] \end{aligned} \tag{2.26}$$

$$\begin{aligned} q_*(s,a) &= E[R_{t+1} + \gamma \max_{a'} q_*(S_{t+1,a'})|S_t = s, A_t = a] \\ &= \sum_{s',r} p(s',r|s,a)[r + \gamma \max_{a'} q_*(s',a')] \end{aligned} \tag{2.27}$$

For a finite MDP Equation 2.26 and Equation 2.27 have a unique solution independent of the policy [61]. Nevertheless, in practice there is no closed form solution for these equations. Therefore, one must resort to approximate and iterative methods using *dynamic programming* or Monte Carlo methods.

2.3.2 Exploration versus Exploitation

There is a trade-off between *exploration* and *exploitation* when training a reinforcement learning agent. Exploration refers to taking actions that come from the current best version of the learned policy. Exploration instead is concerned with taking more actions to obtain more training data [26]. The dilemma is that neither exploration nor exploitation can be pursued exclusively without failing at the task [61]. There exist some strategies to balance the trade-off between exploration and exploitation, e.g, by employing a *greedy* approach shown in Equation 2.27 choosing the action with the highest payoff [61, 62]:

$$A_t = \underset{a}{\operatorname{argmax}} Q_t(a) \tag{2.28}$$

A simple modification of Equation 2.28 is to fix $\varepsilon > 0$ and choose a random selected action, with probability ε and go with the greedy choice otherwise. This strategy is called ε-greedy. Another approach is *Boltzman exploration* [61, 62] which is given the sample means of the actions at time t. The next action is then drawn from the multinominal distribution of π_t, where:

$$P_t(a) = \frac{\exp\left(q_t(a)/\tau\right)}{\sum_{i=1}^{n} \exp\left(q_t(i)/\tau\right)} = \pi_t(a) \tag{2.29}$$

Here in Equation 2.29 τ is a temperature parameter which is annealed over time. Finally, there also exist approaches based on the concept of *optimism in the face of uncertainty*. This basically means that the learner should choose the action with the best upper confidence bound (UCB) [61, 62]:

$$A_t = \underset{a}{\operatorname{argmax}} \left[Q_t(a) + c\sqrt{\frac{\ln t}{N_t(a)}} \right] \tag{2.30}$$

In Equation 2.30 t is the time, c controls the degree of exploration and $N_t(a)$ is the number of times that action a has been selected [61].

2.3.3 Algorithms & Learning in RL

One can characterize RL problems into two main classes: *prediction* and *control* where each is followed by different approaches as *value iteration, policy iteration* and *policy search* [62]. These different approaches use different algorithms. Note that all these algorithms are implemented in both [52, 21, 56] libraries used in the thesis. In general deep reinforcement learning is based on the algorithms below but with deep neural networks to approximate V^*, Q^* and A^* [4].

Dynamic Programming & Monte Carlo Methods. In order to find optimal solutions for Equation 2.26 and Equation 2.27 one must resort to using approximate methods as dynamic programming. The key idea of dynamic programming is to use the value functions to organize and structure the search for good policies [61, 62]. Once a policy π has been improved using v_π to yield a better policy π', one can then compute v'_π and improve it again to yield policy $\pi^{''}$ [61]. This is what *policy iteration* is about and is described more in algorithm 2 found in Appendix B.

One drawback with algorithm 2 is that it is quite computationally expensive as it involves policy evaluation over the full state space [61]. A remedy for this is *value iteration*. The basic idea is to use early stopping, i.e., one update of each state, which is shown in algorithm 1 in Appendix B. Conversely *Monte Carlo methods* (MCM) require only experience, i.e., sample sequences of states, actions, and rewards, from actual or simulated interaction with the environment [61]. MCM solves reinforcement learning problems based on averaging sample returns.

Q-Learning & SARSA. *Temporal difference* (TD) learning can be seen as a combination of Monte Carlo methods and dynamic programming. The algorithm learns directly from raw experience without a model of the environment's experience [61, 4]. The simplest TD method makes the following update in Equation 2.31:

$$V(S_t) \leftarrow V(S_t) + \alpha \left[R_{t+1} + \gamma V(S_{t+1} - V(S_t)\right] \tag{2.31}$$

This idea is applicable both to *Q-Learning* and *State-action-reward-state-action* (SARSA) which are both cases of TD learning.

However, Q-Learning is an *off-policy* method whilst SARSA is an *on-policy method* [61, 4]. On-policy methods attempt to evaluate or improve the policy used to make decisions. Off-policy methods instead evaluates or improve a policy different from that used to generate the data [61]. In other words, on-policy methods estimate the value of a policy while using it for control, whilst off-policy methods uses two separate policies instead. One called *behavioral policy* and the other *target policy* [61]. Moreover, the goal is to learn an *action-value* function which is estimated with Q, instead of a state-value function. The update rule for SARSA is given below in Equation 2.32:

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \left[R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)\right] \tag{2.32}$$

Q-learning instead estimates q_* [5] with the learned action-value function Q independent of the policy being followed [61]. The main update rule for *Q-learning* is given below in Equation 2.33

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \left[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)\right] \tag{2.33}$$

Deep Q Networks (DQN). In Mnih et al. [46] they present Deep Q Learning (DQN), which replaces the Q function with a neural network called *Q-network*. This method also keeps track of some observations in memory which is called *experience replay* [46]. The agents experience is stored in $e_t = (s_t, a_t, r_t, s_{t+1})$ at each time step in a data set $D_t = \{e_1, ..., e_t\}$. In the Q-learning algorithm, updates from the experience memory is drawn uniformly [46]. Mnih et al. [46] use a deep convolutional network to approximate the optimal action-value function Q^*. The Q-learning update at iteration i uses the loss function in Equation 2.34 below:

$$L_i(\theta_i) = \mathbb{E}_{(s,a,r,s')\sim U(D)}\left[\left(r + \gamma \max_{a'} Q(s', a', \theta_i^-) - Q(s, a; \theta_i)\right)^2\right] \tag{2.34}$$

here γ is the discount factor, θ_i are the parameters of the Q-network. θ_i^- is the parameters of the target network [46].

[5] The optimal action-value function.

Policy Gradients & Proximal Policy Optimization (PPO). Policy gradients methods work by directly computing an estimate of the gradient of policy parameters in order to maximize the expected return, by using stochastic gradient descent [5, 57]. The most common estimator is shown below in Equation 2.35:

$$\hat{g} = \hat{\mathbb{E}}_t \left[\nabla_\theta \log \pi_\theta(a_t|s_t) \hat{A}_t \right] \tag{2.35}$$

where π_θ is a stochastic policy and $\hat{A}_t$ is an estimator of the advantages function at time-step t. The advantage function is shown below in Equation 2.36:

$$\hat{A}_t = -V(s_t) + r_t + \gamma r_{t+1} + ...\gamma^{T-t+1} r_{T-1} + \gamma^{T-t} V(s_T). \tag{2.36}$$

Schulman et al. [57] proposes a new family of policy gradient methods, that alternates between sampling data through interaction with the environment and optimizing a "surrogate" objective function using stochastic gradient ascent. The surrogate function is given by Equation 2.37 below:

$$L_t^{CLIP+VF+S}(\theta) = \hat{\mathbb{E}}_t = \left[L_t^{CLIP}(\theta) + c_1 L_t^{VF}(\theta) + c_2 S[\pi_\theta(s_t)] \right] \tag{2.37}$$

where c_1, c_2 are coefficients, and S is an entropy bonus. L_t^{CLIP} is the clipped surrogate objective and L_t^{VF} is a squared-error loss [57].

2.3.4 Issues with deep RL

The use of reinforcement learning, and especially deep reinforcement learning, has been glorified for some time due to a lot of promising results as the ones presented in Silver et al. [59, 60]. With this in mind, these impressive results would not have been possible without the use of reinforcement learning. However, training and using reinforcement learning is not trivial, where some of the major issues are presented below:

1. *Deep RL can be sample inefficient*. Reinforcement learning has its own planning fallacy. Learning a policy usually needs more samples then what one thinks [34]. Simulations run on the popular MuJoCo physics environment need for instance between $[10^5, 10^7]$ time steps to learn different task [31].

2. *Performance compared to other methods*. In theory RL can work for everything, including experiments where a model of the world is not known. However, this comes with the price, that it is hard to exploit any problem specific information that could help learning [34]. The rule-of-thumb is that domain-specific algorithms tend to work better and faster than RL, except for rare cases.

3. *Reward design is hard*. It seems to be hard to design a reward function that encourages the behaviour one wants the agent to learn. Badly designed rewards can lead to overfitting of the agent to the reward. This is why it is important to design a relevant reward signal [61, 34]. Often *shaped rewards*[6] are easier to learn compared to *sparsed rewards*[7]. Some possible remedies however are to use sparsed rewards or careful shaping of the reward [34]

4. *Generalization*. If you want to be efficient in one specific environment, you tend to overfit. However, generalizing an agent to another environment would result in bad results [34] meaning that transfer learning is not easy to archive.

[6] Increasing rewards in states that are close to the end goal.

[7] Only gives reward in goal state.

Chapter 3

Related Work

In this chapter previous related work is presented, to give an overview over what has been done previously, as well as current state of the art. Also the difference between this work and previous work is discussed. The chapter is divided into the following sections: *Agent based Financial Markets*, *Market Microstructure* and *Reinforcement Learning in Finance*. An overview of previous research is presented in Table 3.1.

Table 3.1: Summary and comparison of most important previous related work by category, with comparison to this work.

Agent based Financial Markets			
Study	**Techniques**	**Strategies**	**Difference to this work**
Raberto et al. [54]	na	Chartist Fundamentalist Random	Not realistic prices behaviour
Martinez-Jaramillo and Tsang [45]	Genetic Programming Decision Trees	Technical Fundamentalist Noise	Not realistic environment
Brandouy, Mathieu, and Veryzhenko [9]	Genetic Programming Human agent	Technical Risk-averse Mean-variance random	Matching engine
Market Microstructure			
Ho and Stoll [33]	Dynamic Programming	Market Maker	Not focused on limit order book (LOB)
Glosten and Milgrom [25] Das* [18] and Das [19]	Dynamic Programming Bayesian Learning	Market Maker	Focused on Dealer Market Information based models
Reinforcement Learning in Finance			
Moody and Saffell [47]	Q-Learning Recurrent Reinforcement Learning (RRR)	Algorithmic Trading	Trading Dynamics not main focus Different value functions
Kearns and Nevmyvaka [36] Hendricks and Wilcox [32]	Q-Learning Reinforcement Learning	Dealer / Market Maker	Optimized Trade execution Price Prediction No LOB
Yang et al. [68]	Q-Learning Inverse Reinforcement Learning	HFT Trader	Trader Identification
Darley [17]	Q-Learning Reinforcement Learning Dynamic Programming	Dealer with and without inventory Parasitic Dealer Dynamical Dealer, Q-Learner Spread Learner	Tick size changes Not using PPO or DQN No LOB
Pastore, Esposito, and Vasilaki [51] Rutkauskas and Ramanauskas [55]	Q-Learning Reinforcement Learning Genetic algorithm	Naive / Short-term	Number of agents No DQN or PPO agent Market data No LOB

3.1 Agent based Financial Markets

The field of finance and economics have used various approaches to model financial markets. Among these, three main groups of approaches can be distinguished: *statistical models*, *Dynamic Stochastic General Equilibrium (DSGE) models* and *Agent-Based Models (ABM)* [42]. Agent-based financial markets of different characteristics have been developed for some time. An early model is given in [54] where they, in their *Geona market*, have agents adopting strategies as chartist, fundamentalist or at random. A drawback with the model is that it does not manage to correctly represent the price behavior in the market [54]. In this thesis the author aims to do so by using reinforcement learning instead.

Good overviews of agent-based modelling in a financial setting, are given in [45, 7, 40]. A more recent overview of agent based modelling in finance and economics is presented in [42]. In Martinez-Jaramillo and Tsang [45] they introduce CHASM which is a software platform that allows users to experiment with different market scenarios. This market is composed of technical, fundamental and noise traders, where genetic programming seems to be the core learning process employed and decision trees. A difference between the study by Martinez-Jaramillo and Tsang [45] and this thesis, is that the environment used will be more complex, with a matching engine and different algorithmic trading strategies as market making. To successfully study market microstructure phenomena, the author will use reinforcement learning instead of genetic programming or decision trees for training the agents in the complex simulated environment.

There are two major approaches to agent-based financial market simulations, where the first one focuses on specific market structure. The second approach is more focused on generating a flexible environment with flexible settings for the agents employed [9]. This thesis will focus more on the second approach in order to see what type of behaviour the agents exhibit exposed to different market conditions. Namely, the dealer and limit order book markets. In Brandouy, Mathieu, and Veryzhenko [9] they design an artificial stock market called ATOM based on the Euronext-NYSE Stock exchange with the possibility to add *human agents*.

They have, for instance, a limit order book as this thesis aims to use. However, it is unclear whether they employ a functional matching engine, which this study will use by using a simulated limit orderbook market. They also use different types of agents, for instance, *zero intelligence traders, technical traders, evolutionary agents, risk averse agents* and *mean-variance agents*. Nevertheless, none of these are based on reinforcement learning that will be used in this thesis.

One of the main benefits with using agent-based modelling is that they demonstrate the ability to produce realistic system dynamics, which are comparable to those observed empirically [53]. In the study by Platt and Gebbie [53] they calibrate their agents by using heuristic optimization, Nelder-Mead simplex algorithm and a genetic algorithm. They use a simpler form of matching engine in their study, and they also use a low-frequency trader and high-frequency trader. This thesis focuses on simulating the behaviour of a market maker.

3.2 Market Microstructure

Market microstructure studies the process by which investors' latent demands are ultimately translated into prices and volumes [44]. It is of importance for this thesis to understand market microstructure both theoretically and empirical in order to replicate and even find new interesting behaviour of the agents, employed in this kind of environment. In Haferkorn [27] and Agarwal [3] the effects of HFT trading and fragmentation are discussed, which is of interest for market microstructure, as fragmentation leads to more competition on the markets, that might be demonstrated by the agents. Some common references that are used in the state of the art market microstructure research are found in [50, 30, 44, 43].

In [50, 30, 44, 43] both *inventory-based* and *information based* models are discussed. However, in this thesis the author focuses on inventory-based models for the learning agents. Moreover, information-based models can be used in future studies. A common inventory-based model, that uses dynamic programming to find the optimal dealer price in a one-dealer market is given in Ho and Stoll [33]. This model is the core idea for the simulated dealer markets used in this thesis.

A problem with the model in [33] is that there is no closed form solution, thus approximations are used to get arbitrarily close to the real solution. By using reinforcement learning the author hopes to be able to replicate known behaviour of financial agents, that have been shown in previous papers, extending it and see how the agents perform and behave in a more complex market with a limit orderbook. The information-based models on the other hand tries to incorporate the element that some traders are more informed than others, which is called *asymmetric information* or *adverse selection* in the market microstructure literature [50, 30].

A recent model handling asymmetric information is the one presented in [19, 18], which is an extension of the model presented in Glosten and Milgrom [25]. The Glosten and Milgrom model derives the market makers price by setting equations under asymmetric information to be such that the bid/ask quotes are the expectations conditioned on if the order is a buy or sell order. In Das* [18], on the other hand, they employ a non-parametric density estimate of the true value or fundamental value of the stock by using Bayesian Learning.

The author will be training agents on an environment similar to the one described in [18] but with reinforcement learning instead, with the aim of seeing the same market dynamics as they did in their study. This to further validate the agents learning of possible trading dynamics before using them on the more complex model with the matching engine. Finally, in O'Hara [49] they mentioned that in a high-frequency world, older empirical models may no longer be appropriate. This thesis aims to provide a step in a new direction for this type of research by employing reinforcement learning.

3.3 Reinforcement Learning in Finance

Reinforcement learning or *neuro-dynamic programming* as it also have been called in the literature, have been used previously in a financial setting. Some of the earlier applications have been mainly in trading. In Moody and Saffell [47] the authors explore the opportunity of using reinforcement learning for trading using Recurrent Reinforcement Learning (RRL) and Q-learning whilst looking at the S& P 500 index. Not that much focus is on trading dynamics, which will be the focus of this thesis on a completely different market. In [20, 22, 65, 14, 6] they discuss the usages of different types of reinforcement learning algorithms for trading, where trading is done in, for example, foreign exchange or the energy market.

However, none of these use a matching engine or a more complex environment with several HFT/market maker agents, or with the possibility of completely changing the market conditions. This thesis aims to do this with the EXSIM simulations. In Kearns and Nevmyvaka [36] they give a good overview of how reinforcement learning is used in market microstructure and high-frequency trading, with some use-cases in optimized trade execution, prediction of price movements and optimized execution in dark pools. In [48, 32] optimized trade execution is studied with the help of reinforcement learning, which is of importance in market microstructure and might be examined in this thesis. However, for this thesis one of the main objectives is to understand and successfully replicate known behaviour of traders in a dynamic market. Some recent studies concerned with understanding the behaviour of traders are presented in [68, 69], where they use *inverse reinforcement learning* (IRL) on trading decisions.

The main idea underpinning IRL is to find the reward function given the observations of optimal trading behaviour and then use it for trader identification. Though it is interesting, this thesis is not interested in capturing key characteristic of already optimal HFT strategies. Instead, the author wants to see what optimal or not optimal behaviour the agents will use, and how they will react to changed market conditions. Therefore, reinforcement learning is used instead. Some previous studies exist using reinforcement learning for market making see, for instance, [35, 15, 23, 58].

Market making strategies will hopefully be implemented and learned by the agents trained in this thesis. However, these are not exclusive, as they agents might demonstrate other strategies. In Cartea, Jaimungal, and Penalva [13] some common algorithmic trading strategies are presented, focusing on stochastic optimal control. A previous study has also been done at NASDAQ investigating the effects of tick size changes Darley [17], where the strategies employed were a simpler form of reinforcement learning together with dynamic programming approaches. This study works as inspiration for the thesis, however the thesis will be different in that it incorporates a more realistic simulated market, With the matching engine, and several different types of agents. Other studies that have been modelling stock agents are [51, 55]. This study aims to do the something similar but on the Nordic stock market, as this study wants to understand the trading dynamics created by market makers.

Nevertheless, in order to train the agents in the trading environments, a good understanding of reinforcement learning is needed. In [4, 46, 41] good overviews of the current state of the art deep reinforcement learning (RL) is presented. Recently, some of the state of the art research, has been driven by DeepMind and their work with the game Go (see, for instances, [59, 60, 64]), that serves as inspiration of best practices. Competitive self-play is introduced and examined in [5, 57], which can be used for future research of using multiagent reinforcement learning. In [11, 12] some of the most common multiagent reinforcement learning strategies are discussed.

Chapter 4

Research Summary

In this chapter, a summary of the research methodology is given. An overview of the research process is shown in Figure 1.1.

4.1 Research Methodology

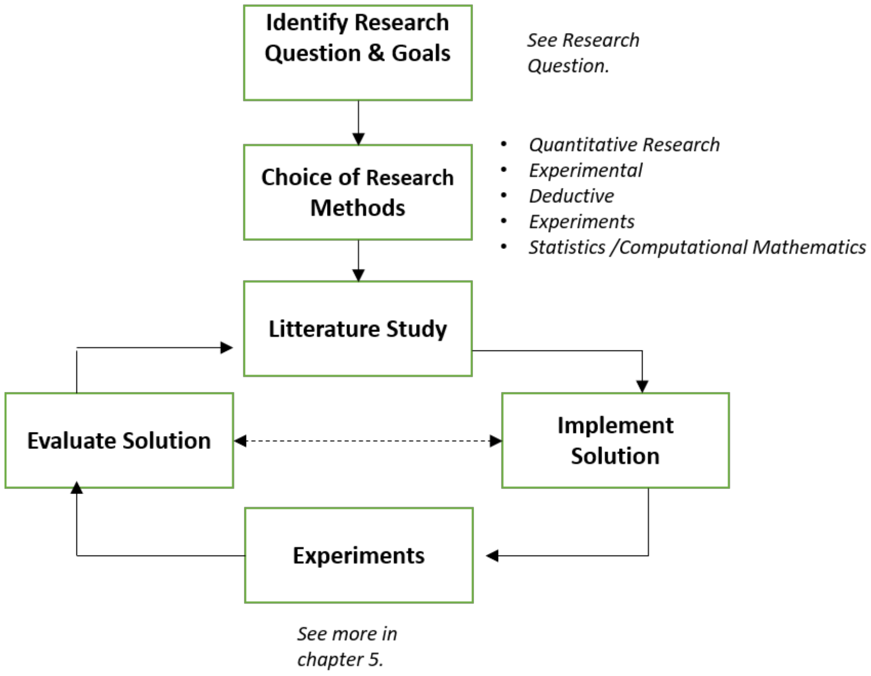


Figure 4.1: Overview of the different research methodology stages conducted during the thesis.

4.1.1 Research Question

The research question for this thesis is shown below:

Research Question 4.1.1. Will trading dynamics such as bid-ask spread clustering, optimal trade execution and optimal inventory costs be exhibited & learned by reinforcement learning agents on a simulated Nordic stock market.

4.1.2 Research Goals

The overall *goal* of this thesis is to understand trading dynamics on a simulated market when market conditions are changed. This goal can be divided into two research goals:

Research Goal 4.1.1. Be able to simulate the effect of any change to market structure.

Research Goal 4.1.2. Applying reinforcement learning to a more complicated environment such as the financial markets.

4.1.3 Research Challenges

While conducting this research some research challenges have been identified and somewhat dealt with:

Research Challenge 4.1.1. Simulating a realistic limit orderbook.

With the current implementations of different reinforcement learning libraries, there does not exist any pre-built limit order market environments. Therefore, the first idea was to use parity a JAVA based exchange engine [1]. However, it turned out to be quite difficult to use this as an environment because it would require some extensive coding to create a wrapper. In order to target this challenge, an own limit order market environment was built with an orderbook and matching engine, inspired by how the NASDAQ limit orderbook market works, namely the INET [2] trading system.

Research Challenge 4.1.2. Training of the agents and accurate learning of agents.

[1] https://github.com/paritytrading/parity

[2] https://business.nasdaq.com/trade/trade-management/technical-information/inet-nordic.html

Training reinforcement learning agents can be quite difficult and requires both time and tuning of parameters, reward functions and other quantities of interest, as well as a lot of computing power to facilitate training. However, to target this challenge, the author tried to monitor how the agents were thinking by visualizing the agent's actions at each time step. More details regarding the visualization are in chapter 5. The author also monitored the mean, standard deviation and other algorithm related metrics at the end of each episode, to tune hyper-parameters random search was used. The author also looked at how many time steps each episode had, in order to understand if the agent solved the problem faster or lost more often.

Research Challenge 4.1.3. Creating accurate rewards to guide the agents in learning optimal actions.

Shaping rewards in an accurate way so that an agent learns a wanted behaviour is not easy. This is still an active research field today, as bad design of rewards can lead to overfitting. See more regarding shaping of the rewards in chapter 2. Furthermore, to target this challenge, the author have used *sparse rewards*. First, in the earlier experiments, and then later the author switched to shaping of rewards in later experiments, as this seemed to yield better results.

4.2 Research Methods

The choice of methods shown in Figure 4.1 are based on using the portal methodology in [28]. The work in this thesis is performed using quantitative research by conducting various experiments in the different agent environments later, to test the behaviour of these agents using different tests and hypothesis [28]. The different environments are covered in more depth in chapter 5. However, as the author is also trying to establish relationships between different variables, the research can also be seen as experimental research [28]. Running all simulations results in big amounts of data that needs to be examined, processed and analyzed. Therefore, this thesis uses deductive reasoning to compare and test the results of this thesis to previous studies. Finally, statistics are used to analyze the collected data and evaluating its significance.

4.2.1 Literature Study

For the literature study, mainly available commercial and non commerical databases such as IEEE, arkivX.org, Google Scholar and Science-Direct have been used, to get access to the state of the art. Also some searching has been done throughout the web on, for example, different blogs, that recommended certain papers or cases, where relevant references were identified. Some books on market microstructure have also been provided by the principal. Three information streams have been identified as relevant *Artificial Financial Markets, Market Microstructure* and *Reinforcement Learning in Finance*. After a first initial search, some 25 papers have been found to be relevant, where the most relevant papers have already been covered in chapter 3.

4.2.2 Implementation, Experiments & Evaluation

Regarding the technical implementation, experiments and evaluation, this is covered in more depth in chapter 5. However, from a methodological standpoint, some discussion about these is presented in the next sections.

4.3 Validity

In this section a discussion about validity is provided covering both construct, internal and conclusion validity. This is of importance when needing to devise different tests, in order to make sure that the simulated data from each simulation is valid. Validity in short indicates the degree to which an instrument measures what it is supposed to measure [39].

4.3.1 Construct Validity

A measure is said to possess construct validity to the degree that it confirms to predicted correlations with other theoretical propositions [39]. In other words, if the measure behaves as the theory says. In this thesis the author compares the obtained results with what is stated in previous studies and literature to enforce more construct validity.

4.3.2 Internal Validity

Internal validity is avoiding *experimental artifacts* in experiments, which is an interpretation of an experiment that is a mere illusion. However, also avoiding confounding variables [3] that can introduce biases and increased variance. Indeed, internal validity is connected to experimental control of background conditions. By following research paradigms and using relevant features, one hopes to avoid this. When doing simulations in the thesis, the author has the possibility to limit and change the number of relevant parameters. Thus, by changing the parameters throughout the experiments the author examines the effect of these closely, in order to handle and avoid any experimental artifacts.

4.3.3 Conclusion Validity

Conclusion validity is a measurement of the extent to which conclusions about relationships of variables are reasonable, which is connected to the analysis of the collected data [63]. In this thesis the author employs regression analysis to look at, for instance, price impact, to determine relationships between some of the variables. The author also looks at the correlation between some of the variables, as well as discussing the obtained results with people having subject matter in market microstructure, with the main goal of increasing the conclusion validity of the thesis results.

4.4 Ethics

Ethics independently of quantitative or qualitative research is the moral principles in planning, conducting and reporting results of research studies [28]. There have not been identified any possible ethical violations by conducting this research, as all the data used is simulated without any connection to real-world market participants. The author also believes that the results of this thesis cannot be used for any unethical behavior.

[3] An independent variable that has not been taken into account effecting dependent variables.

Chapter 5

Implementation

5.1 Overview

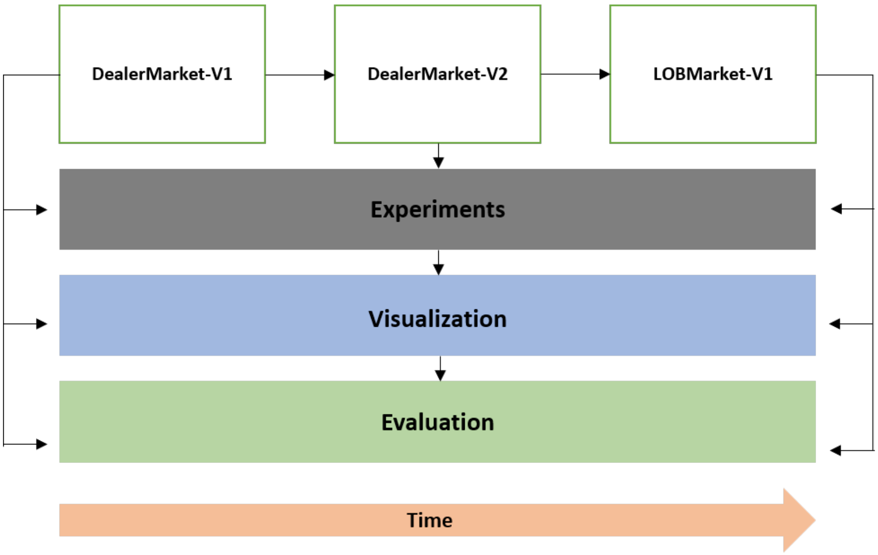


Figure 5.1: Overview of the implementation of the different agents and environments in the thesis.

In this chapter the implementation and experiments on the different agents and environments used are described. In Figure 5.1 an overview is provided, which will be addressed more thoroughly in the following sections.

5.2 Environments & Experiments

Table 5.1: Major differences between the environments, where a_t is what action, h_t how much history given, i.e., the last frames. Offset and slope is used for the demand curves. Each environment is seen as a way of changing the market structure for the agents.

Agent	a_t	σ	λ	Reward	Offset	Slope	h_t	Funds	Inventory
dmv1	4	0.2	10	Equation 5.7	10	–	10	10^6	1000
dmv2	6	1.5	5	Listing 5.1	8	0.5	10	10^6	1000
lobv1	10	2.0	150	Listing 5.2	8	5	10	$2 \cdot 10^5$	200

In this thesis new OpenAI environments were created, in order to simulate both dealer markets and limit orderbook markets, which are based on the papers and ideas presented in chapter 2.

The environments used are *DealerMarket-v1, DealerMarket-v2* and the *LOBMarket-v1*, where the main differences between them are shown above in Table 5.1. Nevertheless, the same types of experiments have been run for each individual environment:

1. *Firstly*, by running the models for a shorter period of time to find relevant hyper-parameters using random search, for some 100 iterations. Each run was simulated for 200-500 episodes of a maximum 10 000 intervals or time steps.

2. *Secondly*, training the models for some two million time steps for intervals of 10 000[1] to collect data, monitor and visualize learning of the agent. Here the author also used different random seeds in order to take into account randomness in the results.

3. *Thirdly*, testing the environment on a random agent/policy to use as benchmark against the trained agents, in order to see if the agent has actually learned anything.

4. *Finally*, performing statistical tests and price impact regressions on the collected data to see if phenomena from the literature can be found.

[1] Each time step is equivalent to 1/10th of a second. Meaning that each simulation last for at most $(1000 * 2000)/(3600 * 8.5) \approx 555$ hours or some 65 trading days.

5.2.1 DealerMarket-v1

This environment is inspired by the ideas underpinning Ho and Stoll [33], meaning that the equilibrium price is following a Brownian motion, and a changing demand curve controlled by the slope parameter in Table 5.1. Also note that the orders in the simulation arrive according to a Poisson distribution based on the demand curve. All these parameters are all changing after each episode in the OpenAI environment. DealerMarket-v1 only has a single agent, who is a dealer or market maker with four possible actions: Move bid up (0), Move bid down (1), Move ask up (2) and Move ask down (3). There are no hidden states in the environment, in order to make it fairly easy for the agent, to use this as a benchmark to other environments and agents.

As, input (only the last 10 frames), the agent has the following observed state variables: *volume imbalance, offset imbalance, inventory imbalance, spread, wealth* and *share value*. These are feed into the environment during training (when the agent samples possible actions) using the OpenAI class. Each called $[ibv, ibo, ibif, sp, w, v]$ hereafter and defined below:

$$ibv = (\text{last at bid - last at ask})/(\text{last at bid + last at ask}) \tag{5.1}$$

$$ibo = (\text{off set ask + offset bid})/(\text{spread}) \tag{5.2}$$

$$ibif = (\text{inventory - funds/ref price})/(\text{wealth}) \tag{5.3}$$

$$sp = (\text{offset ask - offset bid}) \tag{5.4}$$

$$w = (\text{funds/price ref + inventory}) \tag{5.5}$$

$$v = \text{share value} \tag{5.6}$$

The state variables in Equation 5.1 to Equation 5.6 are changing throughout the training of the agent, where the agents' goal is to optimize its reward, shown in Equation 5.7:

$$Reward = \Delta\text{inventory} + \Delta funds \tag{5.7}$$

Hence optimizing the change of the agents' reward depending on its cash and inventory at each time step.

5.2.2 DealerMarket-v2

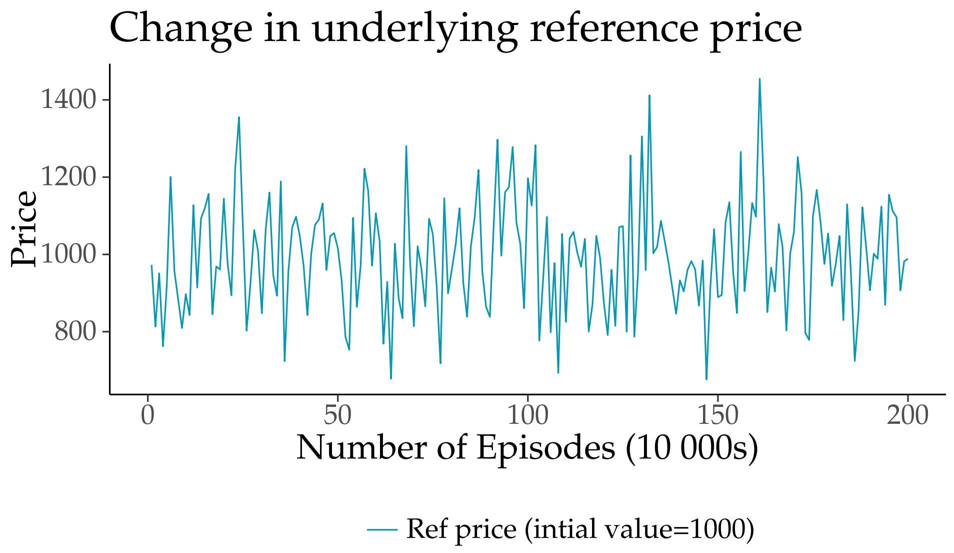


Figure 5.2: Example showing the change in reference price for the DealerMarket-v2 environment.

The *DealerMarket-v2* agent instead has these actions: Move bid up (0), Move bid down (1), Move ask up (2), Move ask down (3), Move ref price up (4) and Move ref price down (5). In Figure 5.2 the reference (ref) price is used to simulate price changes. Also, looking at Table 5.1 this environment is more volatile and complicated. The reward function is also different, shown in Listing 5.1. In practice the agent is given $(+1)$ for each share worth of wealth at the end of an episode, the agent is also penalized with $-100\times$ (% time left) when running out of cash, where tv-true price, ti-inventory, ii-initial inventory, iv-initial value, tf-funds, i_f-initial funds, sc-current step and sm-maximum step

```
if( not(self._is_episode_over())):
    return -1 * ((self.events.volume_bid==0)*0.005 +
        (self.events.volume_ask==0)*0.005)  *
        (self.state.offs_bid+self.state.offs_ask)/50
else:
  return (tv*ti-ii*iv)/iv + (tf-i_f)/iv
      -100*((sm-sc)/sm) #-0.01*10000
```

Listing 5.1: Reward function for DealerMarket-v2.

5.2.3 LOBMarket-v1

In order to, make the experiments more realistic a simplified version of a limit orderbook market was used with an orderbook and matching engine. Firstly, what the agent observes is a bit different from before. As input the agent gets the 10 last frames, however now with the following variables: *stance bid, stance ask, best bid, best ask, the agents best bid and ask, offset of bid and ask, trades and levels for bid and ask, the agents trades, imbalance volume, imbalance of wealth* and *relative wealth*. In total 25 scalar statistics that are what one could expected to be distributed out to participants on a real trading platform[2].

Secondly, levels here indicate the vision width in each direction from the reference price, where the agent can see $(+20/-20)$ directions of previous prices in the LOB. The reward function is slightly changed and varies a bit, as shown in Listing 5.2. As with DealerMarket-v2 (see Listing 5.1) the move from sparse rewards to more carefully shaped rewards is due to better performance when training the agents.

```
# if time, inventory or funds didn’t run out
if( not(self._is_episode_over())):
 return
    min(1,max(-1,atvb*((ampb-self.info[’price_true’])
 /self.info[’price_true’]))) \
  + min(1,max(-1,atva*((ampa-self.info[’price_true’])
  /self.info[’price_true’])))
 else:
  return ((tv*ti-ii*iv)/iv + (tf-i_f)/iv
    -5*self.events[’went_broke’] -25*((sm-sc-1)/sm))
```

Listing 5.2: Reward function for LOBMarket-v1

Finally, as outputs the agent has ten actions, the same six actions (0) to (5) as in DealerMarket-v2. However new for this environment is four other actions: move, submit or cancel orders at bid or ask quote prices. The environment flow simulation is the same as before with Poisson arrivals of orders et cetera. In terms of complexity, this environment is seen as the hardest for the agent to learn and navigate in, as seen in Table 5.1.

[2] At NASDAQ this type of information is sent out via the so called *Net Order Imbalance Indicator* (NOII).

5.3 Implementation

5.3.1 Neural Network Models

The following Neural networks models were used in the different environments, which were found after both hyper-parameter search, and what have been used in previous literature. All the models used are shown on the next page in Table 5.2 and figures Figure 5.3 to Figure 5.5. For *DealerMarket-v1* the author trained a 8-layer fully-connected neural network (FCNN) using the DQN agent and Boltzmann policy with LeakyReLU as activation layer.

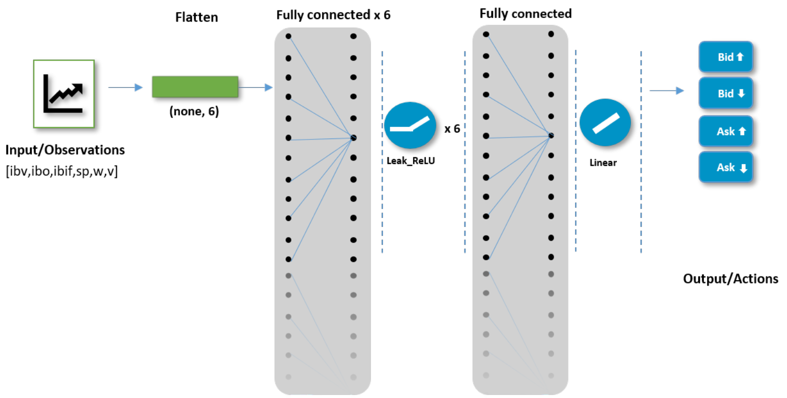


Figure 5.3: Neural Network for DealerMarket-V1 using 7 fully connected layers and leaky ReLU as activation function. Input is the observable state variables, which are flattend to be feed into the network.

For *DealerMarket-v2* the author also trained an 8-layer fully-connected neural network (FCNN) using the PPO agent and with ReLU as activation layer. Here the agent also has six possible actions as seen in Figure 5.4 on the next page.

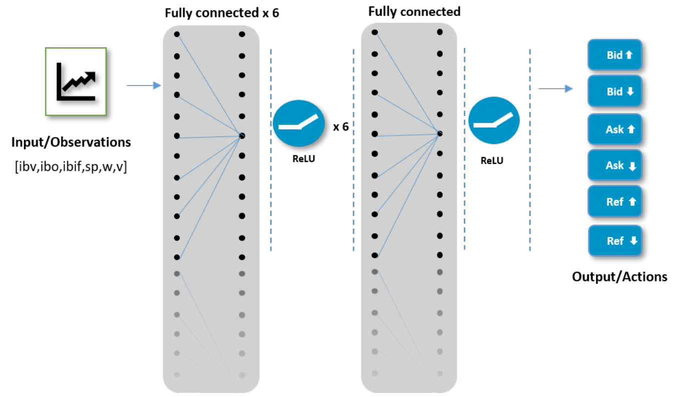


Figure 5.4: Neural Network for DealerMarket-v2 using 7 fully connected layers and leaky ReLU as activation function. Input is the observable state variables.

For *LOBMarket-v1* the author trained an 8-layer fully-connected neural network (FCNN) with two LSTM layers, using the PPO agent, with ReLU as activation layer. Here the agent instead has ten possible action as output, as seen in Figure 5.5.

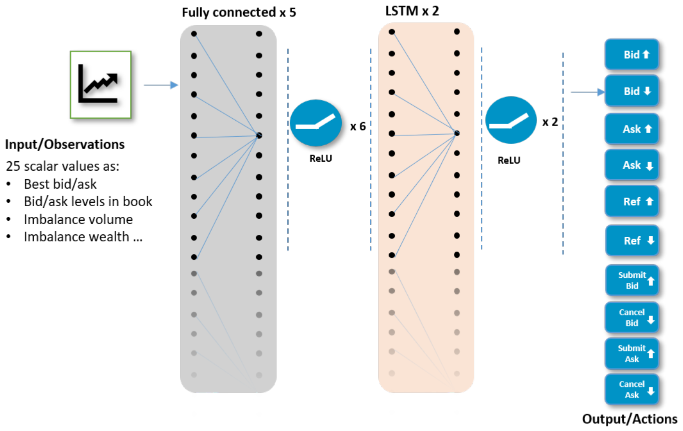


Figure 5.5: Neural Network for LOBMarket-v1 using 5 fully connected layers and 2 LSTM layer. With ReLU as activation function. Input is the observable state variables some 25 scalar values. Output is one of ten actions.

Table 5.2: The different network architectures used in the thesis. Type indicates what type of agent used and policy. The random agent is sampling via a uniform distribution from different actions.

Environment	Architecture	Type	Library
DealerMarket-v1	8-layer FCNN	DQN + Boltzmann	keras-RL
DealerMarket-v2	8-layer FCNN	PPO + ε - decay	tensorforce
DealerMarket-v2	Random model	Random policy	tensorforce
LOBMarket-v1	6-layer FCNN + 2 LSTM	PPO + ε - decay	tensorforce
LOBMarket-v1	Random model	Random policy	tensorforce

In Table 5.2 the main network architectures and models used are shown. DealerMarket-v1 is the only agent using keras-RL and DQN, which did not have any option of training a random policy. The two other environments used tensorforce and the PPO algorithm. As can be seen on the previous page, the network architectures are quite similar with the following number of hidden neurons (in each individual layer): $[1024, 512, 512, 256, 256, 128, 64]$. This was based on both trial and error, the hyper-parameter search and previous papers.

The main differences are that both DealerMarket-v1 and DealerMarket-v2 use only fully connected layers, whereas LOBMarket-v1 has added two recurrent LSTM layers. Looking at Table 5.3 the chosen hyper-parameters from the random search and previous papers are presented.

Table 5.3: Values of the different hyper-parameters used, that were chosen in the thesis. Learning rate (η), episodes (eps), memory (mem), policy parameters (τ, ε), clipping (ϵ), Generalized Advantage Estimate (GAE) lambda (λ).

Environment	Type	Hyper-parameters	Values
DealerMarket-v1	DQN	[η, eps, mem, τ]	$[1e^{-5}, 2 \cdot 10^6, 1 \cdot 10^5, 0.25\ ]$
DealerMarket-v2	PPO	[η, eps, mem, ϵ, λ]	$[3e^{-4}, 2 \cdot 10^6, 3.2 \cdot 10^5$, 0.2, 0.97]
LOBMarket-v1	PPO	[η, eps, mem, ϵ, λ]	$[1e^{-5}, 2 \cdot 10^6, 3.2 \cdot 10^5$, 0.2, 0.97]

5.3.2 Software & Hardware

OpenAI Gym & Baselines

OpenAI Baselines is a state-of-the-art library used for research and testing of different reinforcement learning algorithms created by Dhariwal et al. [21]. From the library this work has implemented the previous mentioned environments.

keras-RL

Keras-RL is a library for reinforcement learning, developed by Plappert [52] with some state-of-art reinforcement learning algorithms such as: *Deep Q Learning* and *SARSA*. It is an easy to use interface of keras modular API for building neural networks. This library also works seamless with OpenAI, which was why it was selected to be used as the primary library for this project. However, as the project progressed, another more up-to-date library that was more frequently updated was needed. Hence the choice fell on *Tensorforce*.

Tensorforce

Tensorforce is developed by Schaarschmidt, Kuhnle, and Fricke [56], and is another python based reinforcement learning library. Built on top of Tensorflow with a modular API passing parameters using python dictionaries. Some implemented agents in the library that are of interest for this project are *AC3, PPO, DQN* and both a *random* and *constant* agent for sanity checks [56].

Platform specification

The majority of the experiments and simulations were performed on a virtual machine on AWS. The author used a p2.xlarge EC2 instances on AWS, with the following specifications: 1 Tesla K80 GPU, 4 vCPUs and 61 GiB RAM. For initial testing and debugging local experiments were also run on a Intel i5 dual-core CPU with 2,40 GHz and 8 GB RAM with a Windows 10 Pro operating system.

5.4 Visualization

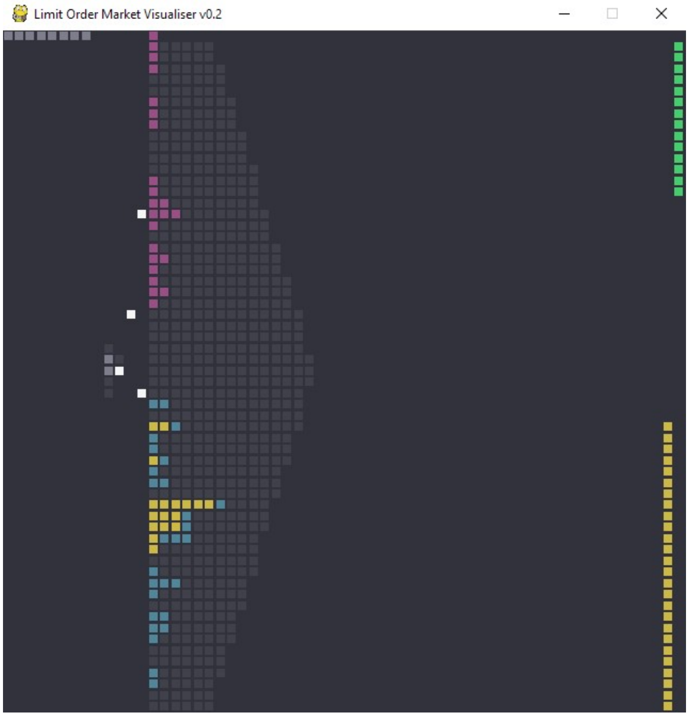


Figure 5.6: Example of visualization during training. Showing how an untrained agent posting bid and ask quotes in a LOB. Purple bricks are ask levels, blue are bid levels, whilst yellow are the agents asks, green the agents bids. Finally, pink and blue are the agents ask and bid trades.

In order to understand what the agents are doing, some visualization have been employed. Firstly, during training using the pygame[3] python library, the visualization shows what bid and ask quotes that the agent thinks are the best to quote. An example of the visualization is shown in Figure 5.6. The gray bricks in the background are the current demand and supply curves, where purple bricks are the current ask levels in the LOB. Conversely blue bricks are the bid levels, yellow is the agent's ask qoute, and green the agent's bid quotes. Secondly, after training visualization of relevant metrics from each simulation is done.

[3] https://www.pygame.org

These visualizations are all done in R reading the training history or callbacks saved in JSON files, where both dplyr [67] and ggplot2 [66] are used for this purpose.

5.5 Evaluation

A common approach in machine learning to evaluate models is to use k-fold cross-validation. However, in this thesis all data generated is from each completed simulation. Therefore, other different metrics are gathered after each simulation in order to evaluate the agents which are discussed below.

- *Price Impact Regression*. An estimate of Kyle's lambda (λ), i.e., the price impact of the agent's orders is done via regression. Using order imbalance (q_n), inventories (∇I_n) and initial inventories ($\nabla Init_n$), where a larger λ implies that volumes have a larger price impact on prices.
- *Visualizing learning & Strategies*. Looking at how the agents act together with the price data stored from each run, to see what strategies are used by the agents.
- *Spreads & Inventory*. Analyzing how the spreads & inventory are changing during training. For instance, if the spreads decline over time, how the correlation between spreads and inventory changes, and the order imbalance.
- *Net Profit and Loss (Net PnL)*. Calculating the Net PnL of the agent to see if the market maker in fact has learned to make a profit or not, thus optimizing its inventory levels, which is defined in Equation 5.8:

$$PnL_t = Cash_t - Init\_Cash_t + \Delta Inv_t \cdot S_t \tag{5.8}$$

- *Compare agent to zero intelligence or random policy*. The random agent is used as benchmarks to see if the agent on average is better or not during each simulation.
- *Stylized facts of simulated data*. Looking for some of the stylized facts mentioned in chapter 2 to see how close or far away from reality the simulations are.

Chapter 6

Results

In this section the results from the different experiments and environments are presented, where DealerMarket-v1 is not as extensively analyzed as the other environments because it serves as an initial baseline. This section starts with some stylized facts about the simulated data, continuing with a breakdown of different statistic gathered after each simulation.

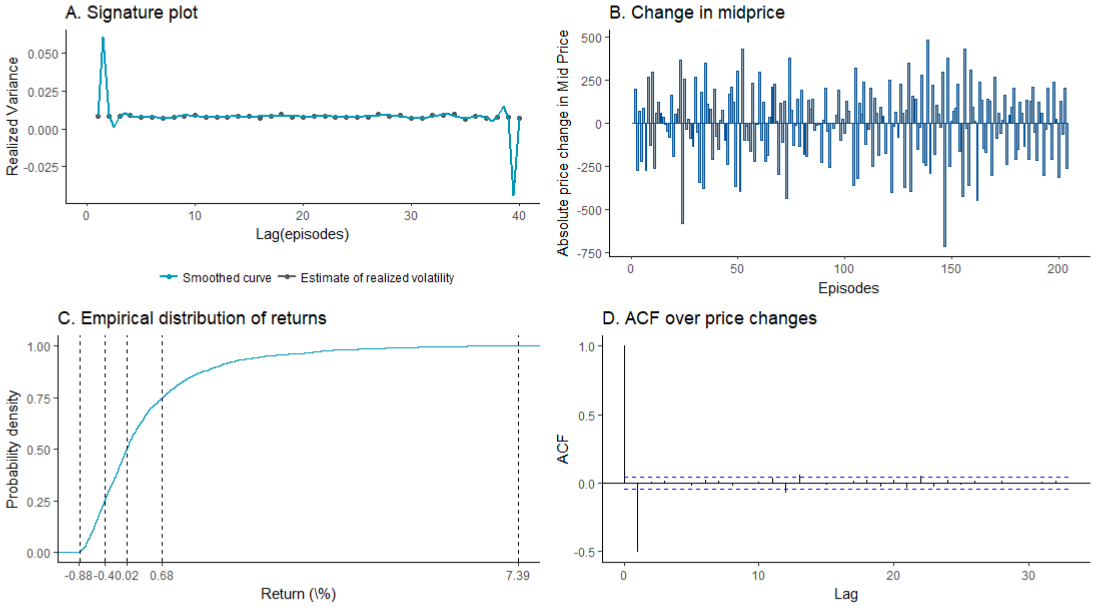


Figure 6.1: Stylized facts for DealerMarket-v2. A signature plot over the realized variance (A) and plot over periodicity in mid prices (B). Empirical distribution of returns (C) and an acf over price changes (D).

6.1 Stylized Facts of Data

In Figure 6.1 one sees that the simulated prices exhibit certain behaviour, where plot (A) shows a signature plot over the realized variance or volatility for DealerMarket-v2. Looking at the plot it is clear that the simulated stock prices are exhibiting weak mean reversion, which is in line with what is mentioned in [8]. Looking at plot (B) the midprice changes are exhibiting activity clustering, as in [8], certain periods of repeating similar changes are displayed in the midprices.

Finally, looking at plot (D) one can also see an absence of linear correlations clearly with near zero autocorrelations after lag 1. This is also in line with what is stated in previous literature. In Figure 6.2 one can see similar patterns for LOBMarket-v1 as in Figure 6.1. However, note that both the signature plot (A) and midprice changes plot (B) are different. The signature plot is still mean reverting with a small trend, whilst the changes in mid prices are more frequent.

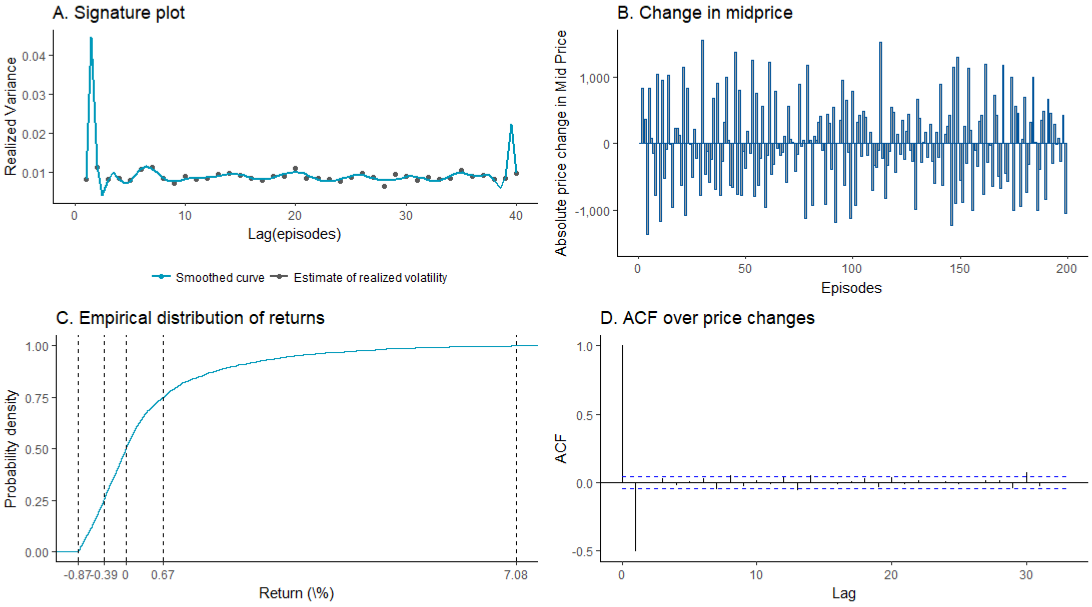


Figure 6.2: Some stylized facts for LOBMarket-v1. A signature plot (A), periodicity in mid prices (B), empirical distribution of returns (C) and an acf over price changes (D).

6.2 Agent's Rewards

Summary of rewards

Table 6.1: Table over rewards for DealerMarket-v1, DealerMarket-v2 and LOBMarket-v1 with mean, max, std and 95% confidence interval (CI) after training for $2 \cdot 10^6$ steps.

Model	Mean	Std	Max	Min	CI
DealerMarket-v1	391.88	150.47	478.04	-9.43	16.18
DealerMarket-v2	22.47	78.46	347.40	-816.11	2.85
LOBMarket-v1	-5.02	23.88	145.46	-526.77	0.88

Regarding Table 6.1 above one can see that the accumulated reward is fairly consistent for DealerMarket-v1 compared to the other environments presented later. Also note that the reward schemes are somewhat different between the different environments as discussed previously, which will obviously affect the distribution of the accumulated rewards. This environment is easier to navigate in for the agent with less volatility and less arrivals of orders, compared to the other environments.

As can be seen from Table 6.1 the standard deviation of the reward is quite large, mainly due to the randomness associated with training reinforcement learning agents. Nevertheless, calculating confidence intervals for the reward results in an average reward close to 400 (391.81 ± 16.18). In Table 6.1 for DealerMarket-v2 the agent makes on average a reward close to 22 (22.47 ± 2.85).

Note also that the reward is much smaller for this environment compared to the DealerMarket-v1 environment, most likely due to higher complexity in the DealerMarket-v2 environment with higher volatility in the underlying prices of the asset and changed reward function, explaining the quite wide fluctuation between min and max values of the reward. Finally, looking at the reward for the LOBMarket-v1 in Table 6.1, the mean reward is the smallest -5 (-5.02 ± 0.88). As, this is the hardest environment for the agent, it is also making it harder for the agent to find optimal behavior.

Q-value function, episode reward & loss

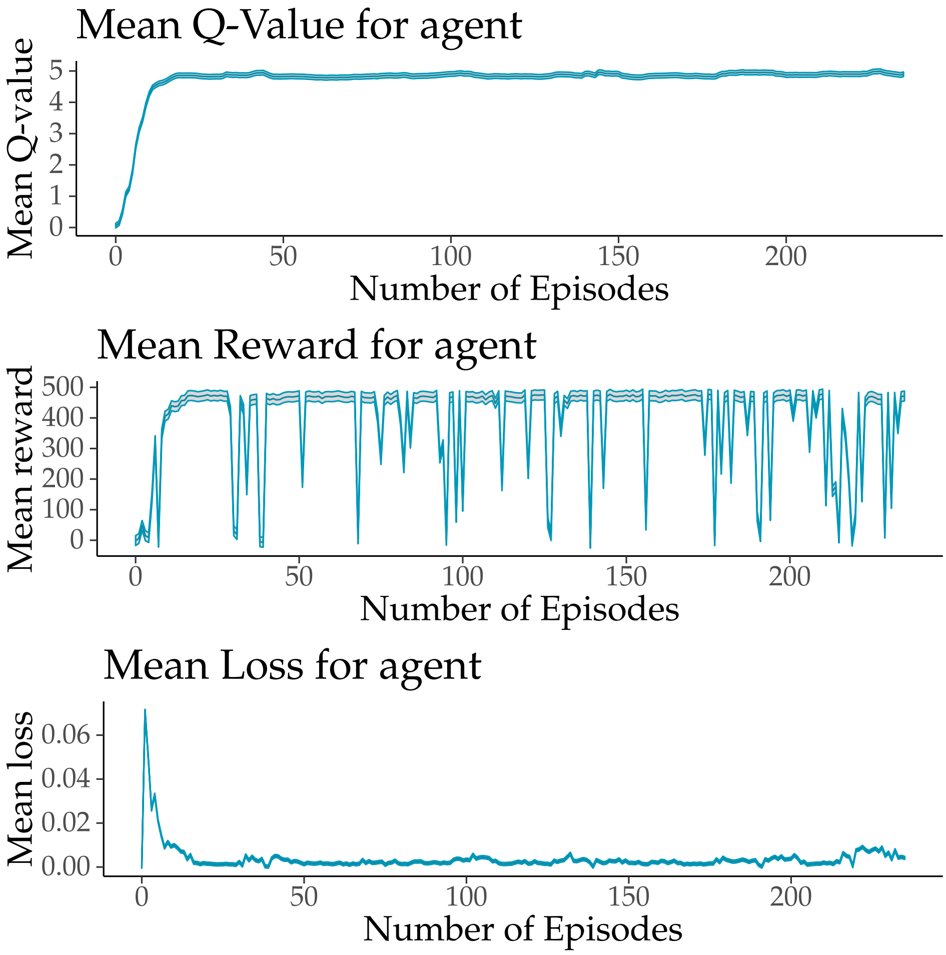


Figure 6.3: Plots from training the DealerMarket-v1 agent. With best weights after training for $2 \cdot 10^6$ time-steps. Plotted with 95 % confidence interval

In Figure 6.3 the complete history from training the DealerMarket-v1 agent is shown, which was the only environment using the DQN. Notice that the optimal Q-value (action-value function) stagnates quite quickly at a value of five, which seems to be the optimal Q-value. Nevertheless, the reward seems to be quite high, and is still fluctuating quite heavily. At, the same time the mean loss for the agent is decreasing.

Episode reward vs. random policy

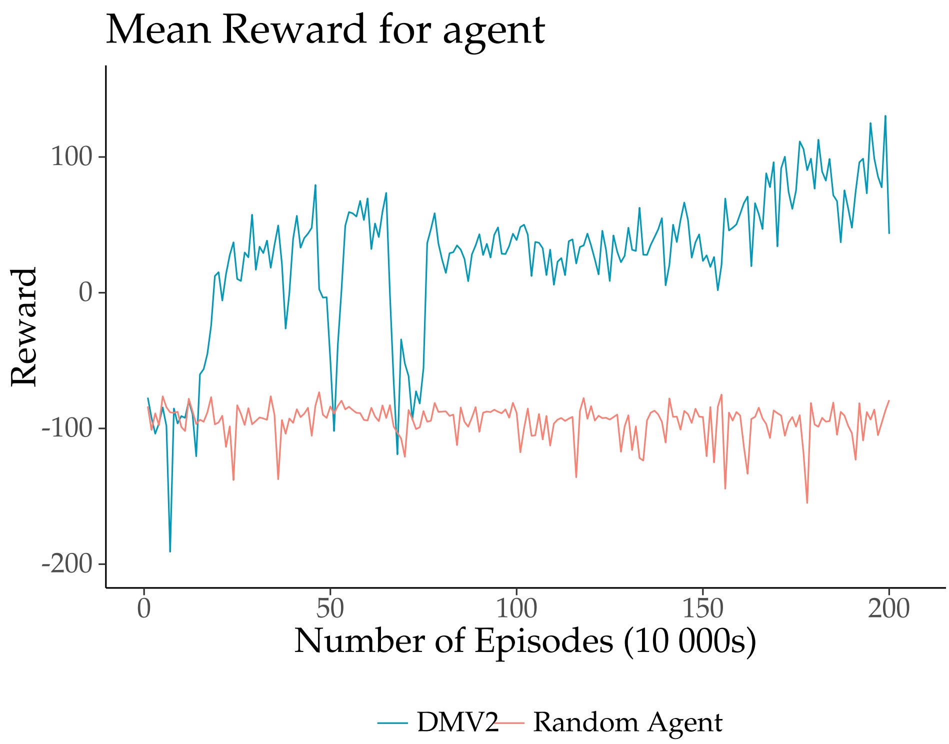


Figure 6.4: Plot showing the average reward after training DealerMarket-v2 for $2 \cdot 10^6$ time steps. Average is taken after 10 000 time steps, compared to a random agent on the same environment.

In Figure 6.4 the mean reward for both the DealerMarket-v2 agent and a random policy on the environment is shown, both this and the LOBMarket-v1 agent used the PPO algorithm. The mean is taken after each episode, i.e., after some 10 000 time steps, which clearly shows that the agents is worse than a random policy in the beginning of the training. The agent is slowly learning a more optimal behaviour after some 75 000 time steps when it starts to outperform the random policy quite consistently. Compared to DealerMarket-v1, the mean reward seems to converge more smoothly using the PPO algorithm, as the same type of fluctuations previously seen are not present here. This is expected as policy gradient methods tends to converge more easily than the DQN.

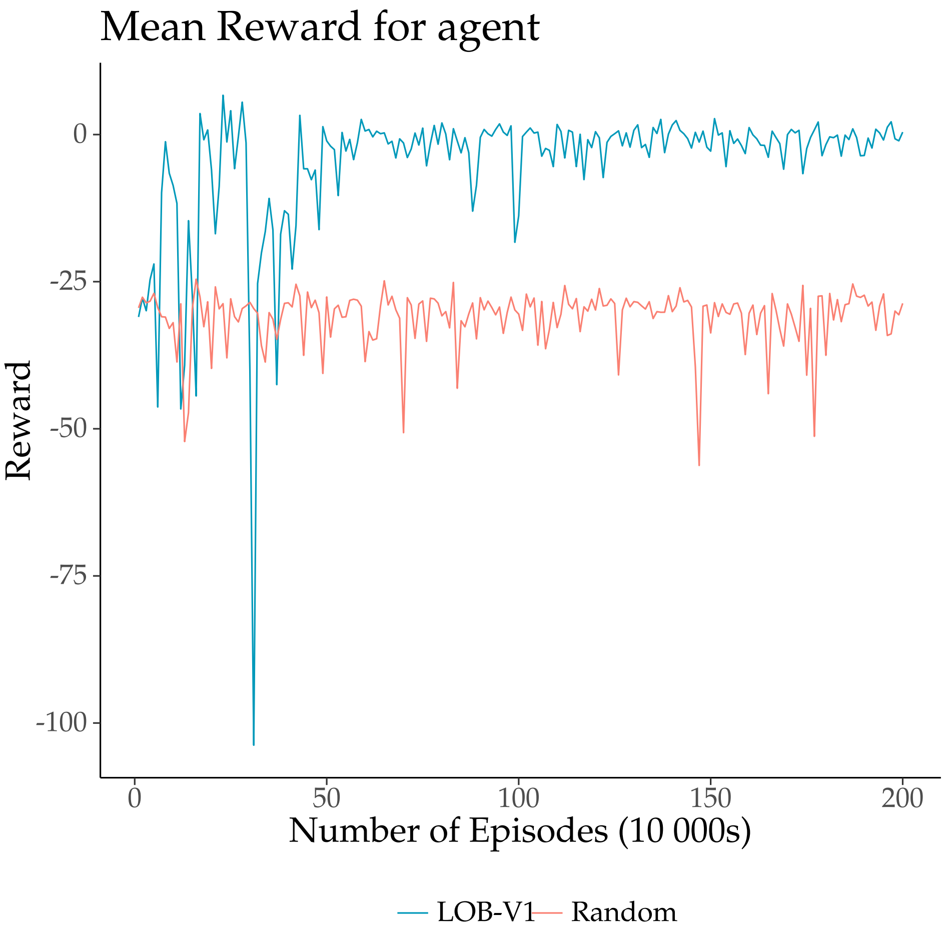


Figure 6.5: Plot showing the average reward after training LOBMarket-v1 for $2 \cdot 10^6$ time-steps. Average is taken after 10 000 time steps, compared to a random agent on the same environment.

The mean reward for LOBMarket-v1 environment is shown in Figure 6.5. Clearly the agent is better than a random policy, which has a negative reward throughout the full simulation. Note that a random approach seems to work better in the limit order book environment compared to the dealer market environment. As can be seen from the DealerMarket-v2 the agent is performing worse than the random policy in the beginning of training. However, after roughly 25 000 time steps the agent outperforms the random policy. The reward is dropping a bit at the end of training (as for DealerMarket-v2), which can be due to a too high learning rate because the author did not use any annealing of the learning rate during training.

6.3 Price Impact Regressions

Table 6.2: Results from running price impact regression on the change in mid prices during the simulations. ** indicates significant estimates.

	DealerMarket-v1			
	Estimate	Std. Error	t value	Pr(>|t|)
β_0	1.0648	0.5943	1.79	0.1711
β_1	0.0112	0.0084	1.32	0.2771
β_2	-0.0757^{**}	0.0237	-3.19	0.0497
β_3	-0.0058	0.0116	-0.50	0.6542
	DealerMarket-v2			
β_0	-56.8926^{**}	25.199	-2.26	0.0242
β_1	3.4324^{**}	1.1661	2.94	0.0033
β_2	-0.2601^{**}	0.0149	17.44	0.0000
β_3	0.0817^{**}	0.0242	3.38	0.0008
	LOBMarket-v1			
β_0	37.6559	187.0559	0.20	0.8408
β_1	43.9947^{**}	11.6959	3.76	0.0003
β_2	-12.8060	12.3870	-1.03	0.3034
β_3	-32.0142	18.0639	-1.77	0.0791

Looking at Table 6.2, *DealerMarket-v1* has a Kyle's $\lambda \approx 0.011$ with $R^2 = 0.741$, *DealerMarket-v2* has a Kyle's $\lambda \approx 3.4324$, with $R^2 = 0.3723$, and *LOBMarket-v1* has a Kyle's $\lambda \approx 43.9948$, with $R^2 = 0.1457$. The form of the equation for all the regressions is shown in Equation 6.1:

$$\Delta S_n = \beta_0 \pm \beta_1 \sqrt{q_n} \pm \beta_2 \sqrt{\Delta I_n} \pm \beta_3 \sqrt{\Delta Init_n} + \epsilon_n \tag{6.1}$$

DealerMarket-v1 is where the agent has the smallest price impact, which is expected due to lower volatility and less frequency of orders. However, with higher volatility in the DealerMarket-v2 environment the price impact seems to increase. Similar this can be seen for LOBMarket-v1, with the largest price impact. Furthermore, other reasons can be that the agent is posting more orders, thus effecting prices more often compared to the other environments. Finally, transforming the regressors with a square root yielded higher R^2, and lower AIC values when performing the regressions indicating a better model.

6.4 Bid-Ask Spread & Inventory

Table 6.3: Table over inventories and spreads for DealerMarket-v1, DealerMarket-v2 and LOBMarket-v1. With mean, std, max, min and 95 % confidence interval for the rewards, after training for $2 \cdot 10^6$ steps

	Inventories				
Model	Mean	Std	Max	Min	CI
DealerMarket-v1	939.41	629.04	3024.19	2.67	120.17
DealerMarket-v2	1206.18	831.95	5994.08	0.00	30.26
LOBMarket-v1	214.96	133.56	1430.24	-1.00	4.91
	Spreads				
DealerMarket-v1	10.28	2.20	22.73	2.38	0.42
DealerMarket-v2	39.49	6.81	47.45	6.50	0.25
LOBMarket-v1	22.95	4.36	32.00	4.40	0.16

Table 6.3 shows a summary over the different agents inventories and spreads. In Figure 6.6 for DealerMarket-v1, looking at plot (A), the gap between the bid, ask and mid prices is quite big. One would expect that the price difference would be narrower, this might be due to the simplistic nature of the environment. However, in plot (C) one sees that the spread is decreasing and looking at the inventory in plot (B) and the volumes in plot (D), the inventory seems to be increasing. While the bid and ask volumes are similar and close to 5, which interestingly is similar to what the Q-value stagnates at. Maybe, due to the fact that the agent learns that 5 in this environment seems to be the optimal volume to post.

For DealerMarket-v2 in Figure 6.7 in figure (A) the prices have tightened likely as it gets harder for the agent to make a profit, hence needing to post more bid and ask quotes. Also, the inventory seems to be increasing (B), whilst the spread is decreasing (C), however the posted bid, and ask volumes seems to be decreasing. In LOBMarket-v1, Figure 6.8 the difference between the prices are also very tight, likely due to that agent post much more bid and ask quotes. The inventory in figure (B) seems to be increasing at the end of training, and at the same time the spread in figure (C) is also decreasing. Note that the prices, posted volumes and inventory have all decreased.

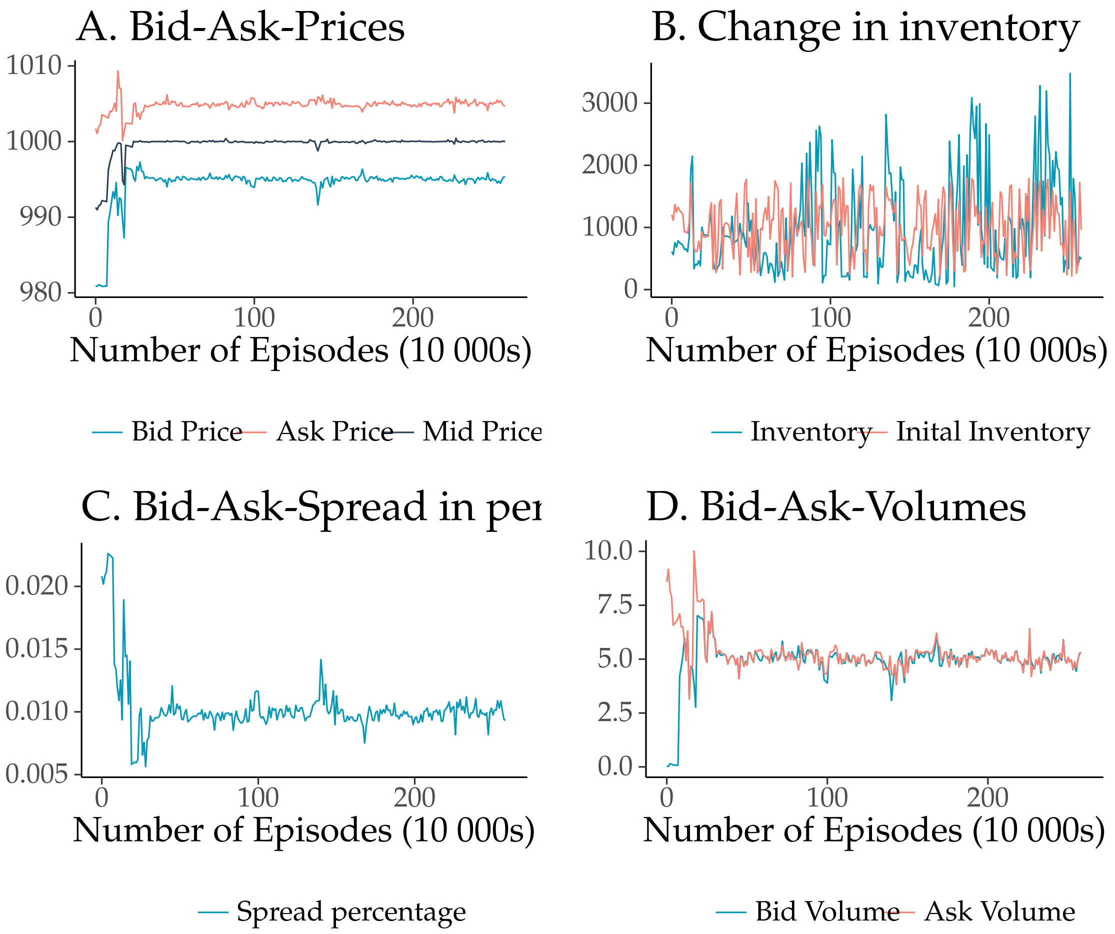


Figure 6.6: Plots for DealerMarket-v1, showing the change in bid-ask spread(A), inventory (B), percentage change in spread (c) as well as bid-ask volumes (D).

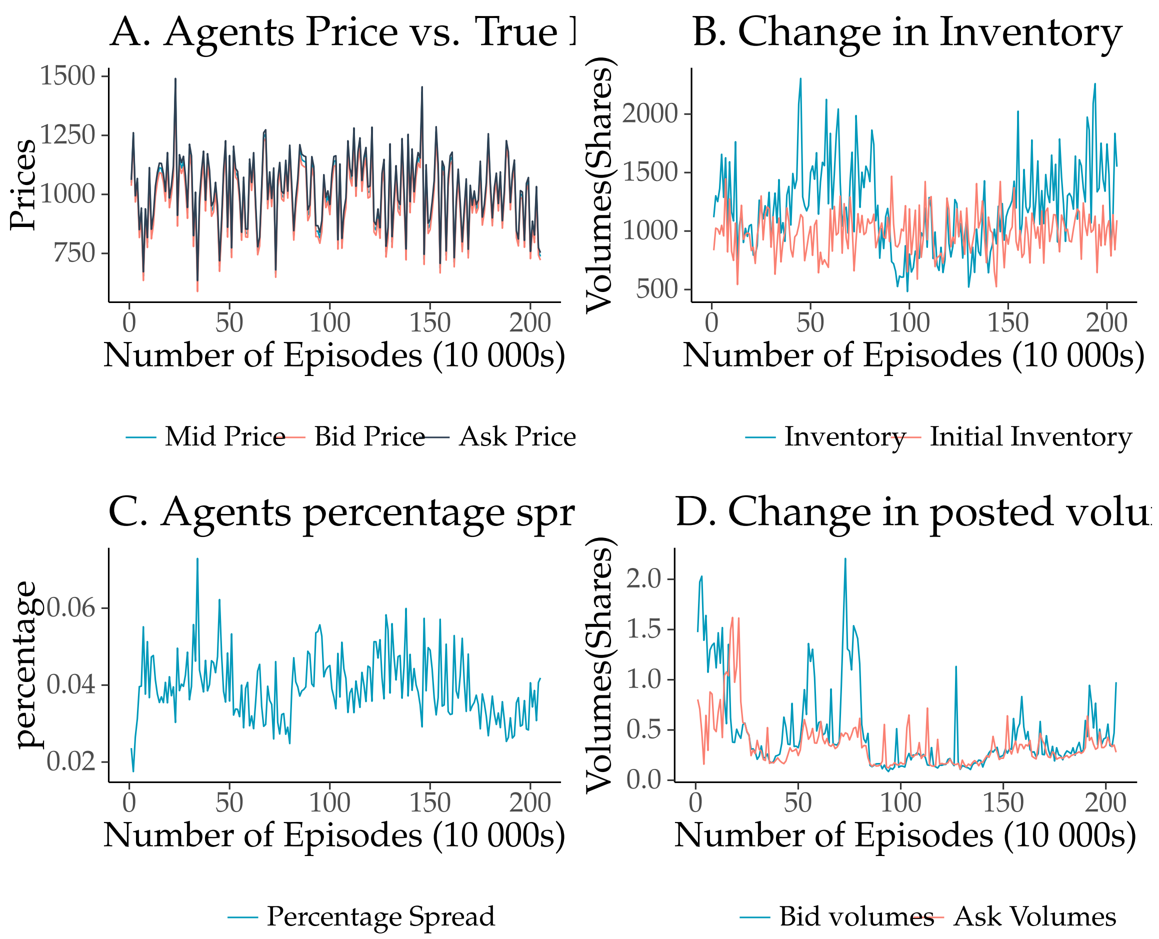


Figure 6.7: Plots for DealerMarket-v2, showing the change in bid-ask spread(A), inventory (B), percentage change in spread (c) as well as bid-ask volumes (D).

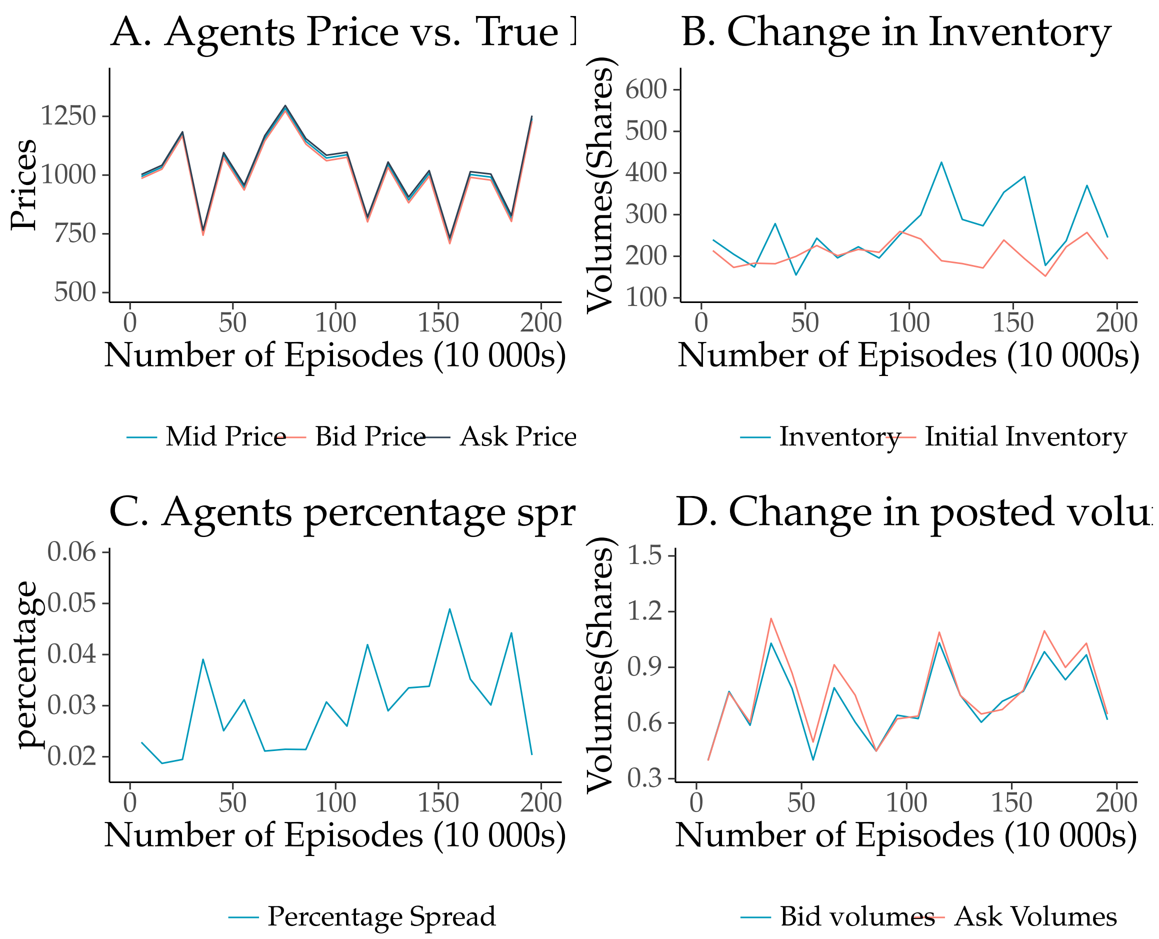


Figure 6.8: Plots for LOBMarket-v1, showing the change in bid-ask spread(A), inventory (B), percentage change in spread (c) as well as bid-ask volumes (D).

Correlation between inventory and prices

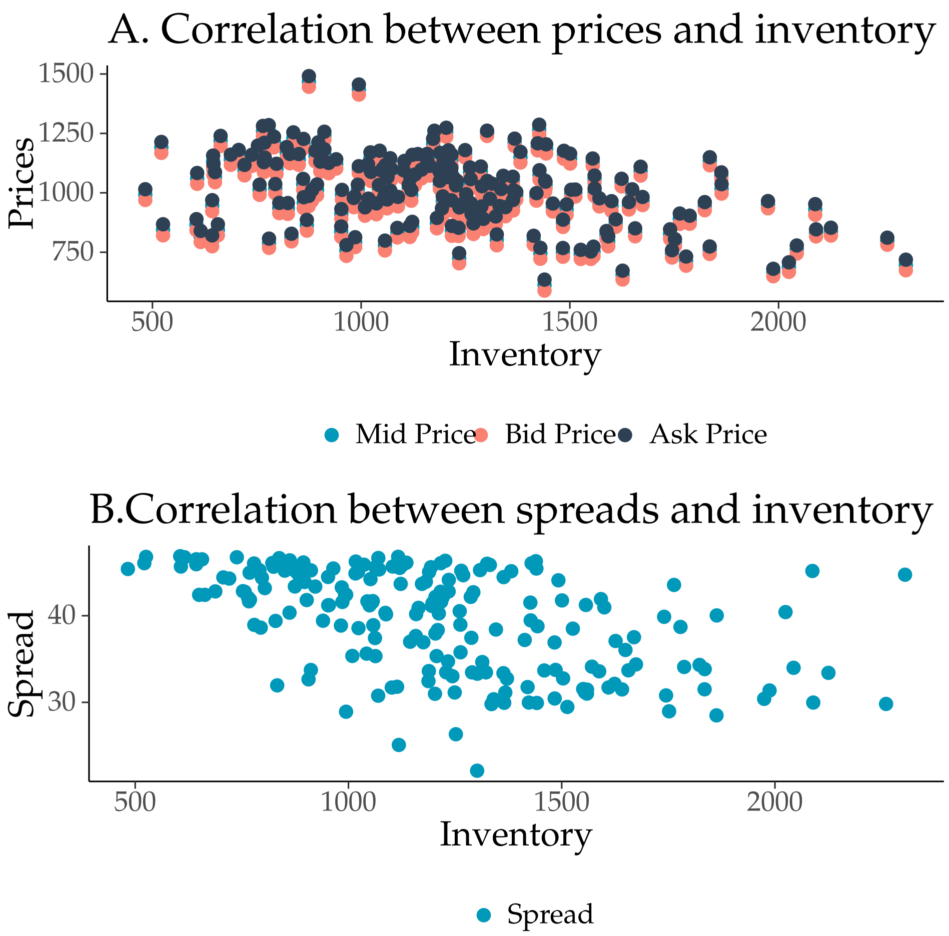


Figure 6.9: Plot for DealerMarket-v2, over correlation between mid, bid and ask price (A) and spread (B) against the inventory, which is the average over each each time step.

In graphs (A) and (B) in Figure 6.9 some correlation between prices and inventory is visible. In fact all prices are negatively correlated to the inventory with $\rho \approx -0.41$. The spreads are also negatively correlated to the inventory with $\rho \approx -0.16$. However, to establish whether spreads are independent of inventory as stated by Ho and Stoll [33], a *Chi-squared Test of Independence* is needed. This test is performed on the full dataset with significance level $\alpha = 0.05$, resulting in a p-value of 0.2403, thus failing to reject the null hypothesis (H_0) of independence.

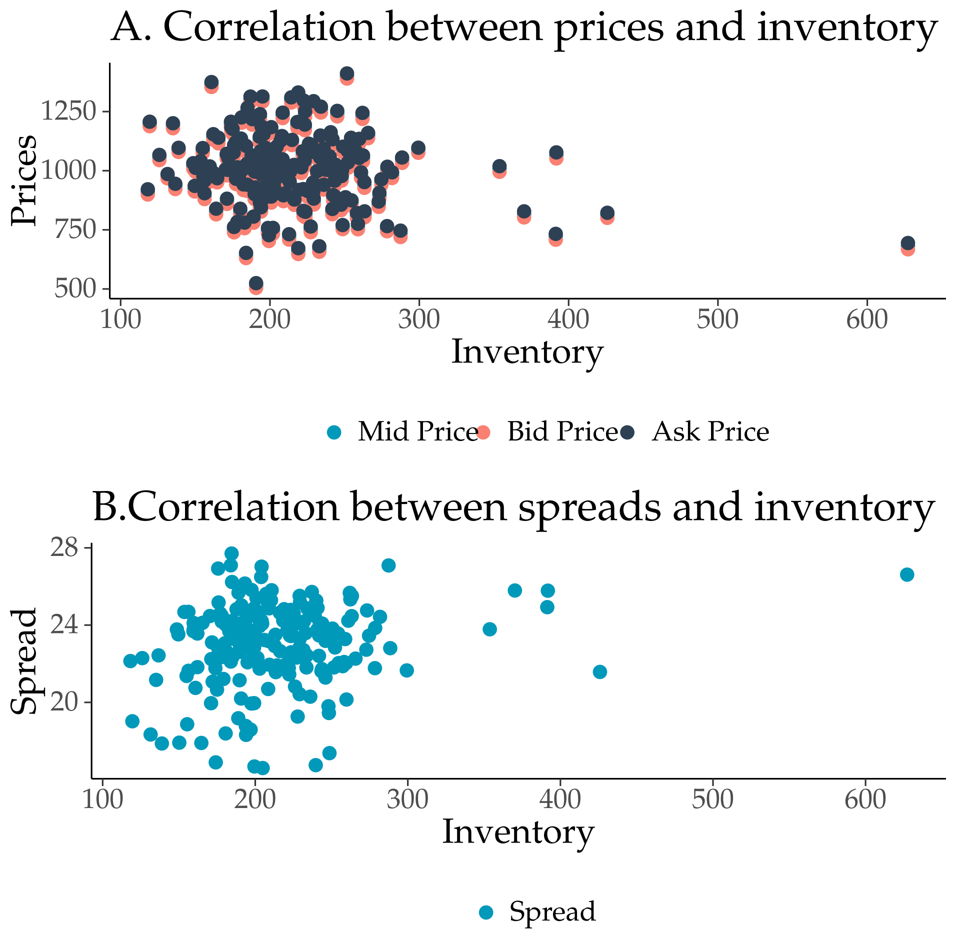
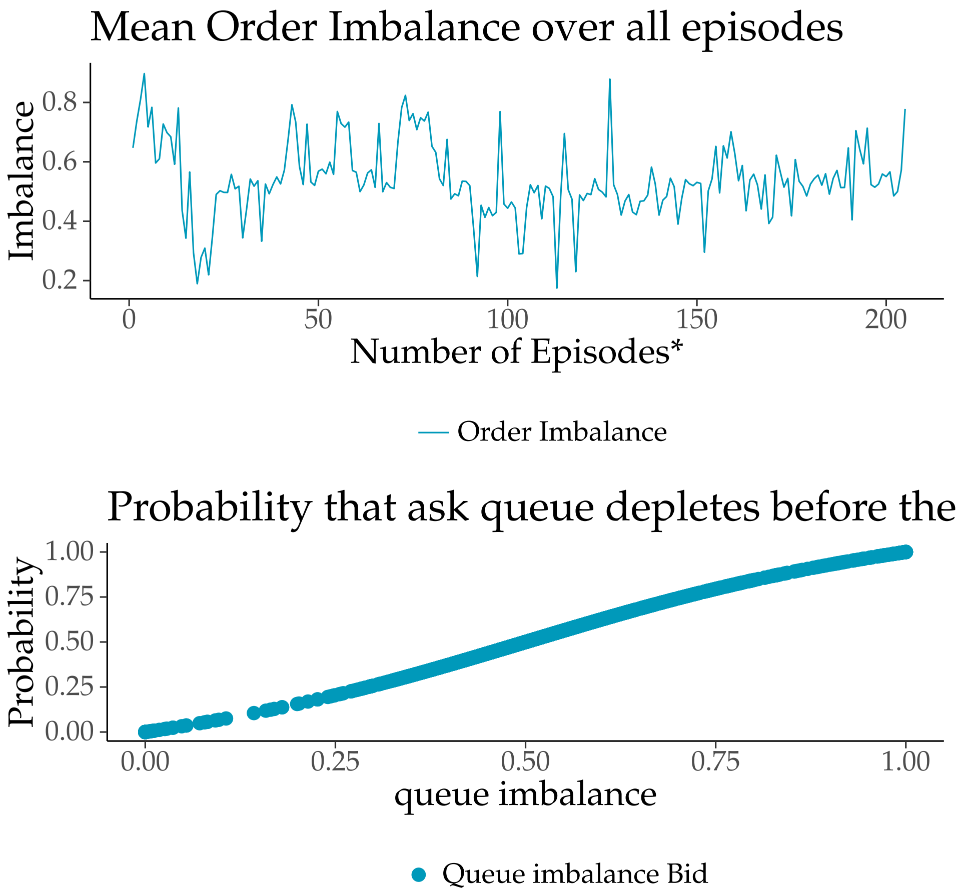


Figure 6.10: Plot for LOBMarket-v1, over correlation between mid, bid and ask price (A) and spread (B) against the inventory. Average over each each time-step.

In graphs (A) and (B) in Figure 6.10 it seems to be some correlation between prices and inventory, in fact all prices are negatively correlated to the inventory $\rho \approx -0.25$. The spreads are positively correlated with the inventory with $\rho \approx 0.13$, this explains the different shapes of the graphs compared to DealerMarket-v2 [1]. However, to establish whether spreads are independent of inventory one can perform a *Chi-squared Test of Independence*. This test is performed on the full dataset with significance level $\alpha = 0.05$, resulting in a p-value of 0.2510, thus failing to reject the null hypothesis (H_0) of independence.

[1] Notice the more narrow interval of preferred inventory levels.

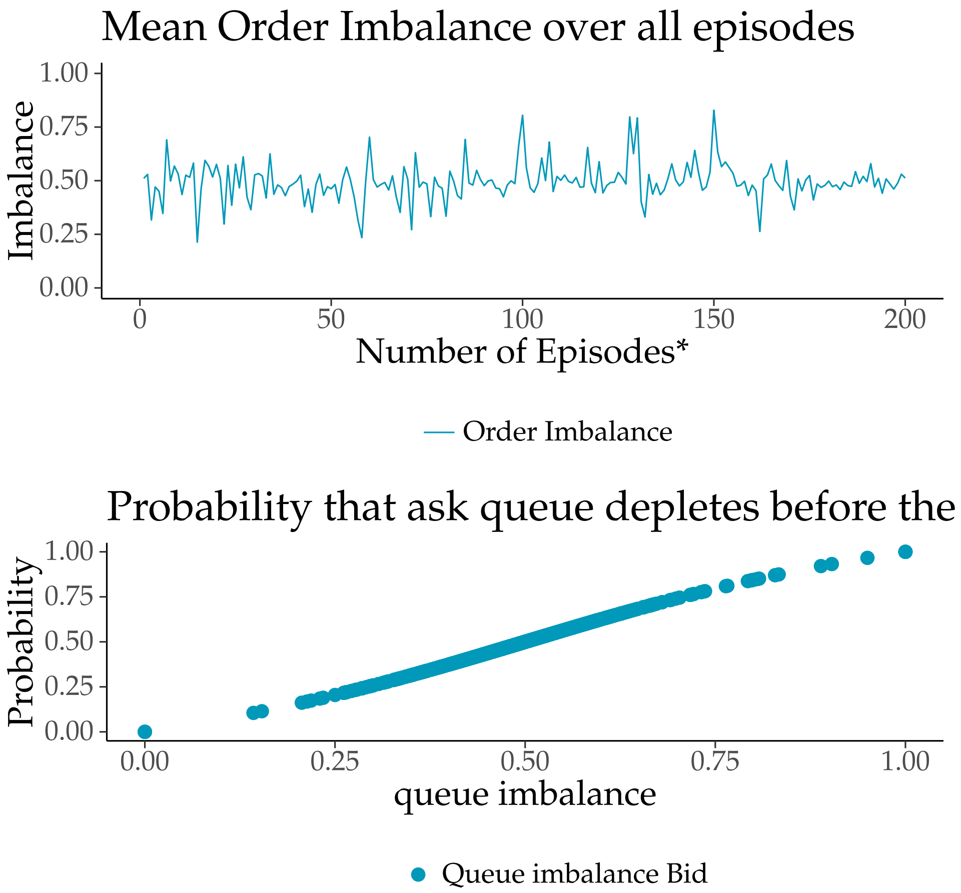
Order Imbalance



*Average over 10 000 timesteps, each timestep is 1/10th of a second.

Figure 6.11: Plot for DealerMarket-v2, over order imbalance between bid an ask volumes (A) and plot over the probability of the ask queue depleting before the bid queue. (B).

In Figure 6.11 the queue dynamics for DealerMarket-v2 between the bid and ask volumes are presented, and in plot (A) one can see the order imbalance, i.e., the difference between the bid and ask volumes to the total volumes. As can be seen from the plot the imbalance is either on the bid side (positive) or the ask side (negative), turning our attention to plot (B) one can instead see the probability that the ask queue depletes before the bid queue. Finally, the shape of this graph is similar to what is shown in Bouchaud et al. [8].



*Average over 10 000 timesteps, each timestep is 1/10th of a second.

Figure 6.12: Plot for LOBMarket-v1, over order imbalance between bid an ask volumes (A) and plot over the probability of the ask queue depleting before the bid queue (B).

In Figure 6.12 the queue dynamics for LOBMarket-v2 between the bid and ask volumes are presented, as can be seen from the plot the imbalance is either on the bid side (positive) or the ask side (negative). The shape of this graph is similar to what is shown in [8]. Note that the order imbalance is on average closer to 0.5 compared to plot Figure 6.11 meaning that both the ask and bid queues are fairly balanced in the LOB, maybe due to the matching engine used in the environment, compared to the DealerMarket-v2 environment.

6.5 Profit & Loss

Table 6.4: Table over P&L for DealerMarket-v2 and LOBMarket-v1. With mean, max, std and 95% confidence interval (CI), after training for $2 \cdot 10^6$ steps

Model	Mean	Std	Max	Min	CI
DealerMarket-v2	−18861.06	540266.01	2220487.97	−1436406.28	19650.46
LOBMarket-v1	−3366.20	60769.67	313634.96	−334806.89	2236.15

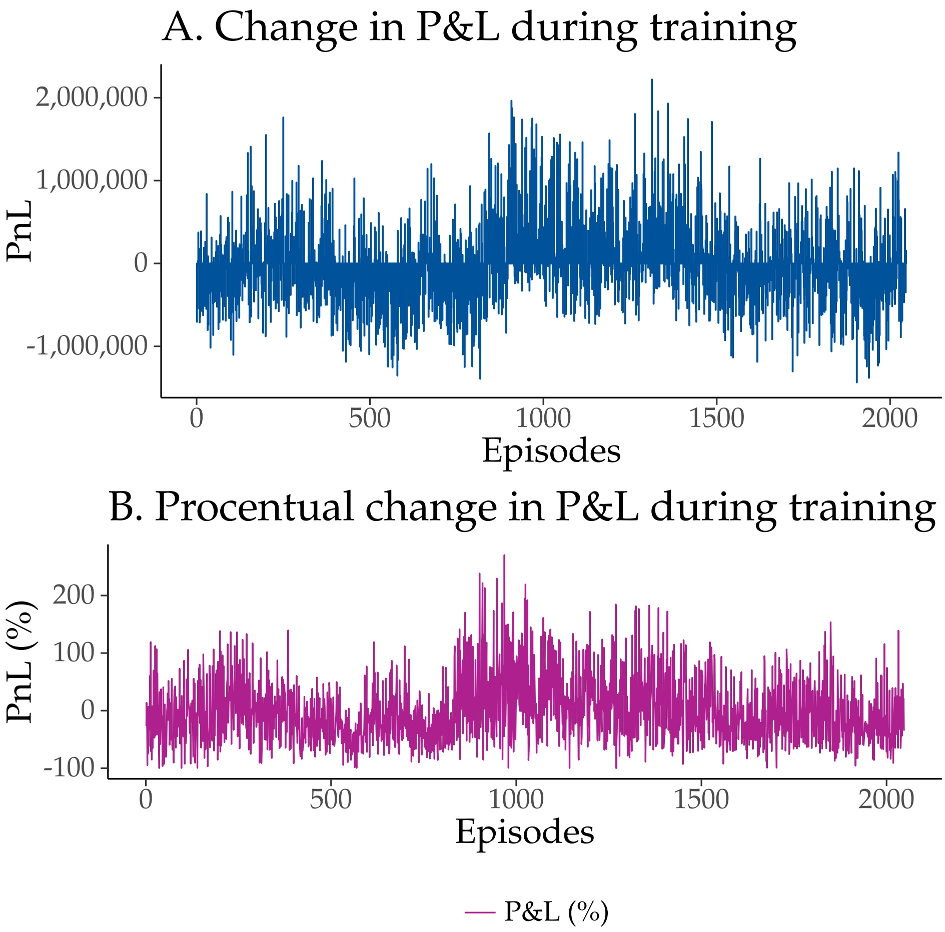


Figure 6.13: Net Profit & Loss for DealerMarket-v2 (A) accumulated P&L (B) and (C) percentage P& for the agent during training, with average after 1000 time steps.

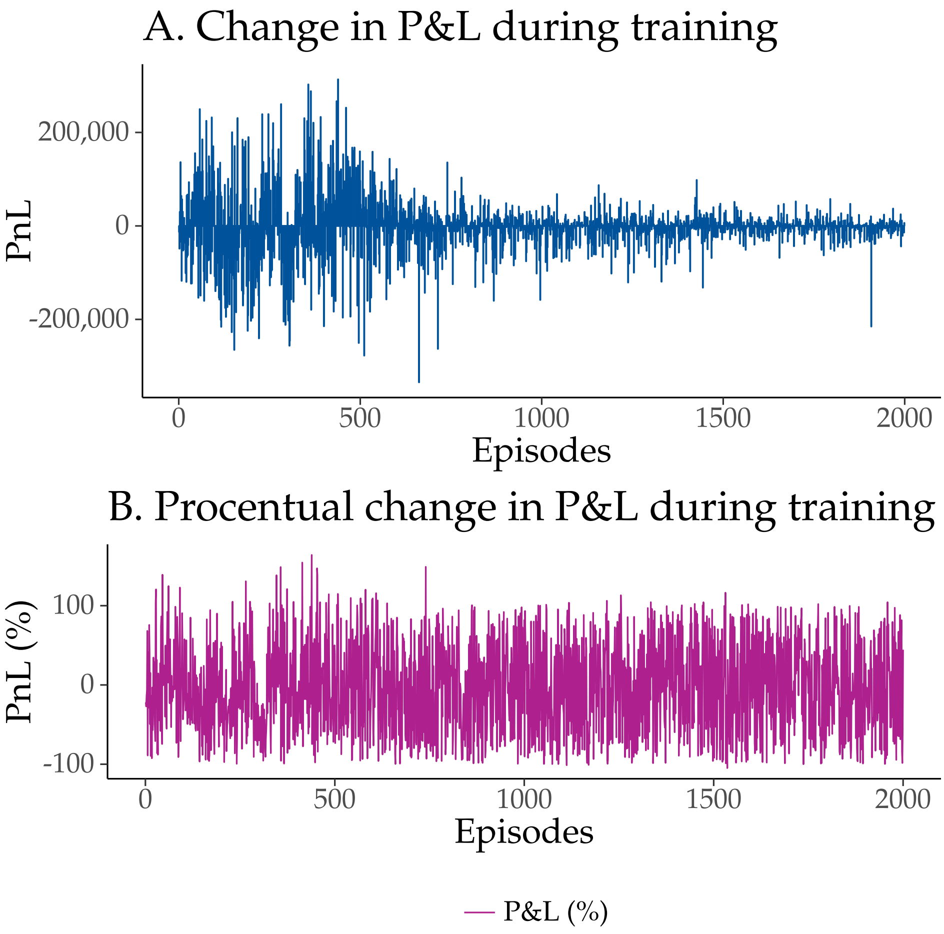
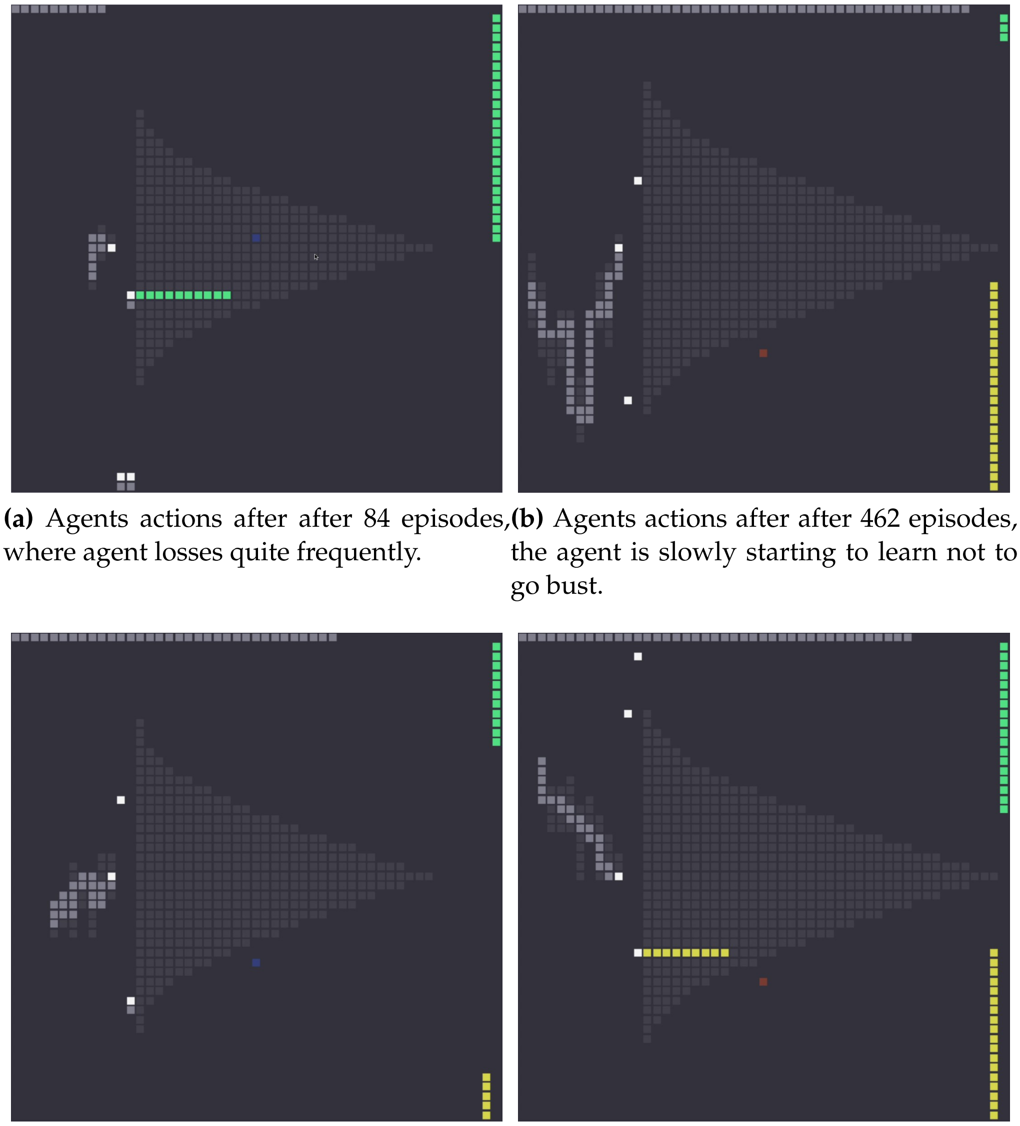


Figure 6.14: Net Profit & Loss for LOBMarket-v1 (A) and accumulated P&L (B) for agent during training, with average after 1000 time steps.

In Table 6.4 a summary of the P&L for the DealerMarket-v2 & the LOBMarket-v1 environments is presented. In Figure 6.13 one can see the DealerMarket-v2 agent's average profit and loss (P&L) per episode, whilst in Figure 6.14 the P&L for the LOBMarket-v1 is shown. Notice that it takes some time until the agents actually learns not to go bankrupt, around some 1 million time steps seems to be needed. With, the chance of making between $100\% - 200\%$ of its initial investment for each agent. However, as the difficulty of the environment increases (LOBMarket-v1), the agent has a harder time of reaching higher rewards. Nonetheless, this is still above zero and better than the random strategies tested on the same environments.

6.6 Visualization of agents states & actions



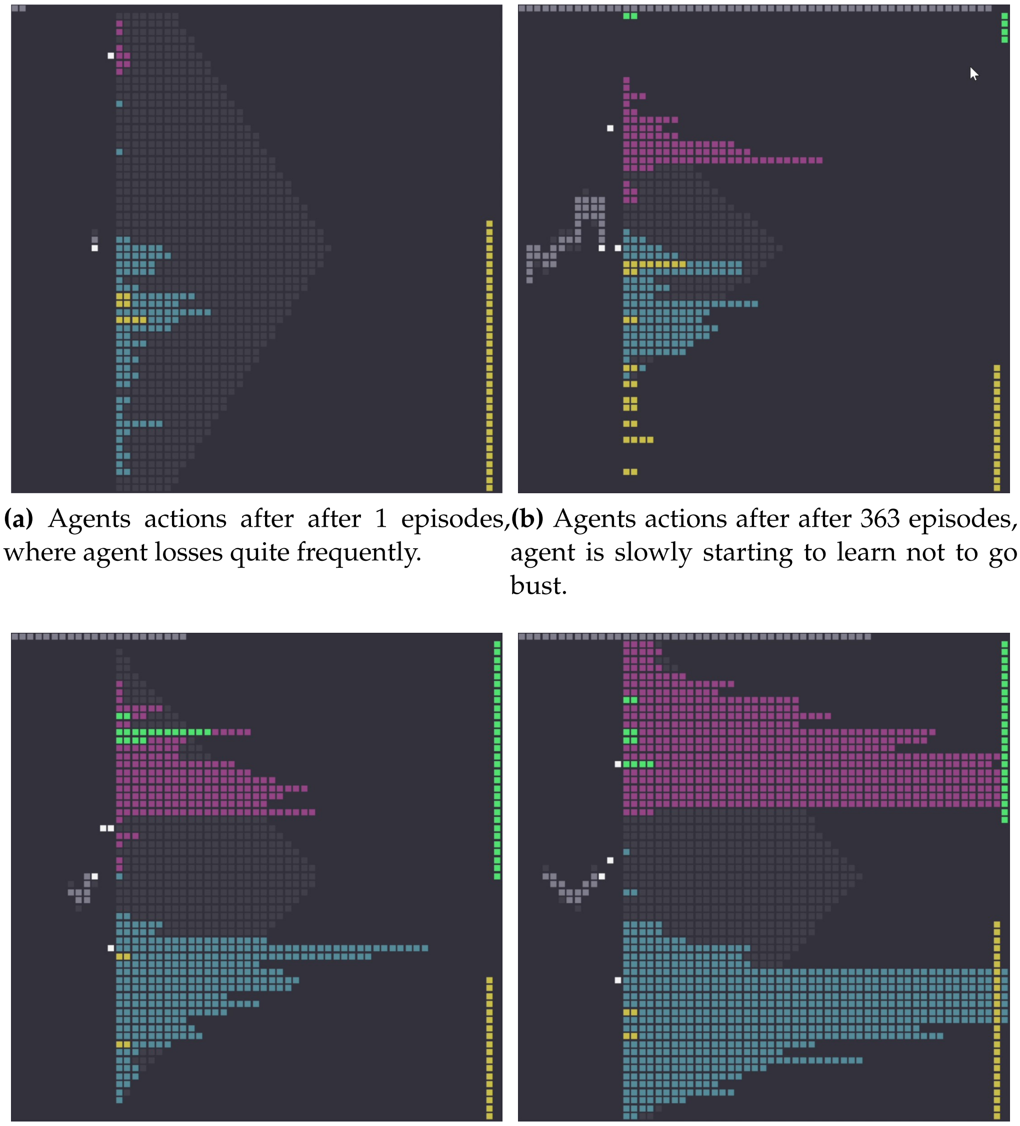
(a) Agents actions after after 84 episodes, where agent losses quite frequently.

(b) Agents actions after after 462 episodes, the agent is slowly starting to learn not to go bust.

(c) Agents actions after after 1169 episodes, the agent has learned to not go bust and makes a small profit.

(d) Agents actions after after 2006 episodes, the agent is balancing quotes and makes profits frequently.

Figure 6.15: Change in agents behaviour during training for DealerMarket-v2.



(a) Agents actions after after 1 episodes, where agent losses quite frequently.

(b) Agents actions after after 363 episodes, agent is slowly starting to learn not to go bust.

(c) Agents actions after after 1069 episodes, the agent has learned not to go bust andmakes a small profit.

(d) Agents actions after after 1896 episodes, the agent is balancing quotes and make profits frequently.

Figure 6.16: Change in agents behaviour during training for LOBMarket-v1.

In Figure 6.15 the DealerMarket-v1 and Figure 6.16 and LOBMarket-v1 agents states are visualized, during different parts of the training. After some 10^6 time steps the agents are learning not to go bankrupt, and starts to exhibit wanted behaviour.

Chapter 7

Discussion & Conclusions

7.1 Discussions

Stylized Facts

In general, when analyzing agent based markets, one usually analyses the statistical properties of simulated data to validate the experimental setup. *Firstly*, looking at the signature plots for DealerMarket-v2 and LOBMarket-v1 one clearly sees *mean-reverting* behaviour of the sampled volatility from the changes in the midprices. These graphs do exhibit some flatness, indicating weak mean reversion as mentioned in [8]. *Secondly*, looking at the changes in the midprices during the simulations, one sees as indicated by Bouchaud et al. [8] some *activity clustering*.

This clustering behaviour shows that the midprice is followed by different periods of a lot of changes in the midprice, both positive and negative as in a real-world market. *Finally*, as mentioned in [16] looking at the ACF for the prices changes, there seems to be an absence of autocorrelations. Where, after lag 1 autocorrelations are close to 0 for the remaining lags. However, if the results would have been presented per time step[1], this would of course effect the shape of the ACF. Nonetheless, according to Cont [16], one would still expect an absence of linear autocorrelations. In summary the simulated data used in the different experiments, exhibit real-world *stylized facts* concluding that the experimental setup is realistic.

[1] Would correspond to intraday periods.

The market maker agents' behaviour

During the simulations for the environments: *DealerMarket-v1 (1), DealerMarket-v2 (2)* and *LOBMarket-v1 (3)* some interesting behaviour has emerged from the agents interaction with the environments. *Firstly*, starting with how the rewards has evolved, the (1) agent had the highest average reward (391.88 ± 16.18), followed by the (2) (22.47 ± 2.45), and (3) (-5.02 ± 0.88). A first remark is that the (1) environment had the lowest σ or volatility compared to the other environments, which had 7.5 to 10 times greater σ. Both the (2) and (3) had different arrival rates of incoming orders compared to (1), this is obviously affecting the difficulty of learning in the environment, thus making it more realistic, and effecting the agent's mean reward. Nevertheless, both (2) and (3) approach higher rewards occasionally with maximum rewards of 347.40 and 145.6 respectively. Clearly the initial values of the inventory and cash also have an effect, as well as the arrival rates of the orders, which is most clearly seen for the (3) environment.

Benchmarking the agent's performance against a random agent or zero intelligence agent was only done for the (2) and (3) environments, as keras-RL did not have this capability. In Figure 6.4 and Figure 6.5 both the (2) and (3) environments outperform the zero intelligence agents employed indicating that the agent's behaviours are more optimal than random strategies, meaning that the choices the agents are doing are not by mere chance, instead some strategic behaviour is employed in order to not go bankrupt. Obviously, this is something that one wants to see when simulating a participant's behaviour on the stock market in order to make it more realistic. Another noteworthy observation is that the zero- intelligence agent is doing better in the (3) environment.

A reason for this is that sometimes acting randomly may not be a dumb choice. However, in a dynamic environment such as the stock market this is clearly not enough to make profits in the long run. Next, looking at the price impact, all agents have different impacts, (1) clearly had the lowest impact with a $\lambda \approx 0.011$. Followed by a significant[2] $\lambda \approx 3.43$ for (2), whereas (3) had a significant $\lambda \approx 43.99$.

[2] $\alpha = 0.05$ or with 95% confidence.

Regardless, of possible estimation errors and the need for more granular data to increase the R^2, what does this really mean? According to the literature [13, 24] the price impact is effected by the level of competition and volatility. One clearly sees the effect of the underlying assets volatility as this thesis is only looking at single agent interactions. (1) with the lowest σ parameter has the lowest price impact, whilst both using higher σ in the (2) and (3) increases the price impact by several tenfolds or centuples meaning that given volumes from (2) and (3) have a larger effect on the observed prices. Naturally, this will effect the market makers profits, as clearly seen in Table 6.1.

Furthermore, looking at the price dynamics for the agent's inventory and spreads, in Figure 6.6 to Figure 6.8 the prices behaves a bit differently for (1) the gap between the prices are wider. Conversely, for the other environments, the price differences are narrower. Note that the absolute differences in the prices are larger for the (2) and (3) environments compared to (1), possibly due to more price fluctuations in those environments due to a higher σ. Interestingly the trend for the spreads in figure (C) is declining, which is in line with what is stated in the literature [8, 30, 44], that the the spread declines over time. Looking at the agents inventories, recall that there is no intrinsic need to hold inventory for a market maker [13]. This as the market maker will buy (sell) in anticipation of a following sale. However, in figure (B) in Figure 6.6 to Figure 6.8 for the different agents, the inventory seems to be increasing compared to its initial inventory.

As a larger inventory could yield higher returns when buying and selling more frequently, the agents might see it fit to balance its future profits with a higher inventory, when taking optimal actions. Noticeable, the average inventory (see Table 6.3) is still close to each agents initial inventory after 65 days of trading. However, an interesting question to pose is how would a real life market maker act? Looking at Figure 6.6, at around 100 episodes the agent's inventory is decreasing (figure D), contemporaneously the spread (plot C) and the volumes are increasing. Comparing this to the tactics in Table C.1, this behaviour coincides with what a dealer or market maker would do with low inventory or that the dealer wants to encourage his clients to sell. Similar examples can be found for the other two environments as well.

Indicating that the different agents are in fact behaving realistically. Moreover, how profitable are the agents market making strategies? The general tendency is that it takes some 10^6 time steps before the agents make positive P&L consistently, which can be seen for both the (2) and (3) environments in Figure 6.13 and Figure 6.14. Interestingly a positive P&L is not always equivalent to getting a positive reward, showcasing the difficulty of shaping rewards for wanted behavior in an agent. On average in the (2) environment the agent has a $P\&L = -18860$, whilst for the (3) environment the agent instead has a $P\&L = -3322$, which is negative as it takes longer times for the agent to learn optimal behaviour.

Looking instead on the rolling mean for the last 100 episodes is of more interest, there (2) has a $P\&L = 17977$ and LOBMarket-v1 has a $P\&L = -1010$. Not surprisingly as it takes time even for experience traders to make a profit in the market. Nevertheless, this is far better than a random strategy. Note that each simulation is ran for approximately 65 trading days, where the agents starts *tabula rasa*. Training for a even longer time or using pre-trained models, would have affected the P&L positively. However, also increasing the chance of overfitting. Looking at Q_3 or the third quantile of the rewards, the agent instead makes a $P\&L = 300200$ and $P\&L = 16330$ for the same two environments.

Advancing with how the spreads and inventories are related and looking at Figure 6.9 and Figure 6.10, clearly the prices and inventories tend to cluster around certain inventory levels, suggesting that some inventory levels are more favourable than others. Ho and Stoll [33] states that the spreads are independent of the inventory levels. This is tested by performing a Chi-squared Test of Independence (H_0) on the full dataset for both environments. The null hypothesis of independence cannot be rejected with 95% confidence, focusing on queue dynamics, and order imbalances for both the (2) and (3) in Figure 6.11 and Figure 6.12. The average order imbalance for (2) is 0.53 and for (3) 0.49, meaning that on average both the ask and bid queues are of equal length, i.e., a balanced buy and sell pressure in the LOB. From the literature [8, 13] higher order imbalance means that the agents should post more buy orders and less sell orders, vice versa holds, when the imbalance is low, that is also seen from the simulations.

Also note that the shape of figure B in Figure 6.11 and Figure 6.12 indicates a monotonic correlation as in [8] between the queue imbalance and the direction of the next price movement. *Finally*, looking at visualizations of the agents behavior in Figure 6.15 and Figure 6.16, some similar patterns emerges. The agent goes bankrupt very quickly early during training, this is also where the agent is displaying weird behaviour. As posting orders with the same price for a long time, however after some 350-450 episodes the agent is not going bankrupt as often. Here the agent also learns that by doing nothing occasionally is optimal. At around 1000 episodes the agent is rarely going bankrupt, and at the end of training the agent learns how to make the market by posting quotes more aggressively in response to changed prices, lower inventory and higher volatility contemporaneously, making a profit and balancing the trade-off between its cash and inventory. To summarize the following have been observed during the different experiments and simulations:

- The simulated environments are realistic as several know stylized facts found in [16] are successfully reproduced.
- Both DQN and PPO agents with adequate reward schemes, i.e., employing sparse or shaped rewards outperforms random or zero intelligence agents.
- A higher price impact from the agents choices is observed when increasing volatility as stated by the theory.
- The agents spreads are declining at the end of each simulation, whereas the agent's inventories are somewhat increasing but close to their initial inventory levels.
- Some of the agent's inventory management corresponds to what a real-world dealer would do as shown in Table C.1.
- It takes at least 1 million time steps before the agent is making a positive P&L.
- Bid-ask spreads for the agents tend to cluster at certain inventory levels.
- Bid and ask queue price dynamics to have been replicated as mentioned in [8].

7.2 Conclusions

The research question examined in this thesis was the following:

> *"Will trading dynamics such as the bid-ask spread clustering, optimal trade execution and optimal inventory costs, be exhibited and learned by reinforcement learning agents on a simulated market."*

After completing this work, the author believes that the research questions has been answered. Where, the main conclusions are: (1) *the simulated environments are realistic* and (2) *DQN & PPO agents can successfully replicate trading dynamics as bid-ask spread clustering*. The author concludes that: *reinforcement learning is a suitable choice in modelling market participants behaviour, such as market makers and HFT traders* when using DQN or PPO agents.

Compared to previous research this thesis shows that both DQN & PPO based reinforcement learning agents are realistic choices when simulating behavior in a dealer market and a limit order book, in order to understand market microstructure. Nonetheless, more research is needed to further validate this work, on real-world data, and applying it to other market structures.

7.3 Future work

After conducting this research, the author sees the following areas for possible future research:

- Extend this works models and environments to a multiagent framework. By doing so one could for instance use competitive self-play between several agents to get and study more complex interactions.

- Apply reinforcement learning with DQN and PPO to derivatives market, or another type of market then a stock market as well as using other algorithms for learning such as actor-critic (AC3).

- Use the limit orderbook provided by Parity together with real historical stock data and use the same type of single agent framework.

Bibliography

[1] Frédéric Abergel. *Market microstructure: confronting many viewpoints*. John Wiley & Sons, 2012.

[2] Samir Abrol, Benjamin Chesir, and Nikhil Mehta. “High Frequency Trading and US Stock Market Microstructure: A Study of Interactions between Complexities, Risks and Strategies Residing in US Equity Market Microstructure”. In: *Financial Markets, Institutions & Instruments* 25.2 (2016), pp. 107–165.

[3] A Agarwal. *High-frequency trading: evolution and the future–how the emergence of high frequency trading is altering the financial landscape as firms look to make money on the millisecond*. Tech. rep. Technical report, Capital Markets, Cap Gemini, Paris, France Google Scholar, 2012.

[4] Kai Arulkumaran et al. “A brief survey of deep reinforcement learning”. In: *arXiv preprint arXiv:1708.05866* (2017).

[5] Trapit Bansal et al. “Emergent complexity via multi-agent competition”. In: *arXiv preprint arXiv:1710.03748* (2017).

[6] Francesco Bertoluzzo and Marco Corazza. “Testing different Reinforcement Learning configurations for financial trading: Introduction and applications”. In: *Procedia Economics and Finance* 3 (2012), pp. 68–77.

[7] Katalin Boer-Sorban. *Agent-based simulation of financial markets: a modular, continuous-time approach*. EPS-2008-119-LIS. 2008.

[8] Jean-Philippe Bouchaud et al. *Trades, Quotes and Prices: Financial Markets Under the Microscope*. Cambridge University Press, 2018.

[9] Olivier Brandouy, Philippe Mathieu, and Iryna Veryzhenko. “On the design of agent-based artificial stock markets”. In: *International Conference on Agents and Artificial Intelligence*. Springer. 2011, pp. 350–364.

[10] Danny Busch. "MiFID II: regulating high frequency trading, other forms of algorithmic trading and direct electronic market access". In: *Law and Financial Markets Review* 10.2 (2016), pp. 72–82.

[11] Lucian Busoniu, Robert Babuska, and Bart De Schutter. "A comprehensive survey of multiagent reinforcement learning". In: *IEEE Transactions on Systems, Man, And Cybernetics-Part C: Applications and Reviews, 38 (2), 2008* (2008).

[12] Lucian Busoniu, Robert Babuska, and Bart De Schutter. "Multi-agent reinforcement learning: An overview". In: *Innovations in multi-agent systems and applications-1*. Springer, 2010, pp. 183–221.

[13] Álvaro Cartea, Sebastian Jaimungal, and José Penalva. *Algorithmic and high-frequency trading*. Cambridge University Press, 2015.

[14] Patrıcia Xufre Casqueiro and António JL Rodrigues. "Neuro-dynamic trading methods". In: *European journal of operational research* 175.3 (2006), pp. 1400–1412.

[15] Nicholas Tung Chan and Christian Shelton. "An electronic market-maker". In: (2001).

[16] Rama Cont. "Empirical properties of asset returns: stylized facts and statistical issues". In: (2001).

[17] Vincent Darley. *A NASDAQ market simulation: insights on a major market from the science of complex adaptive systems*. Vol. 1. World Scientific, 2007.

[18] Sanmay Das*. "A learning market-maker in the Glosten–Milgrom model". In: *Quantitative Finance* 5.2 (2005), pp. 169–180.

[19] Sanmay Das. "Intelligent market-making in artificial financial markets". In: (2003).

[20] Michael AH Dempster and Vasco Leemans. "An automated FX trading system using adaptive reinforcement learning". In: *Expert Systems with Applications* 30.3 (2006), pp. 543–552.

[21] Prafulla Dhariwal et al. *OpenAI Baselines*. https://github.com/openai/baselines. 2017.

[22] Xin Du, Jinjian Zhai, and Koupin Lv. "Algorithm Trading using Q-Learning and Recurrent Reinforcement Learning". In: *positions* 1 (2016), p. 1.

[23] Joaquin Fernandez-Tapia. "High-Frequency Trading Meets Reinforcement Learning: Exploiting the Iterative Nature of Trading Algorithms". In: (2015).

[24] Thierry Foucault, Marco Pagano, and Ailsa Röell. *Market liquidity: theory, evidence, and policy*. Oxford University Press, 2013.

[25] Lawrence R Glosten and Paul R Milgrom. "Bid, ask and transaction prices in a specialist market with heterogeneously informed traders". In: *Journal of financial economics* 14.1 (1985), pp. 71–100.

[26] Ian Goodfellow et al. *Deep learning*. Vol. 1. MIT press Cambridge, 2016.

[27] Martin Haferkorn. "High-frequency trading and its role in fragmented markets". In: *Journal of Information Technology* 32.3 (2017), pp. 283–296.

[28] Anne Håkansson. "Portal of research methods and methodologies for research projects and degree projects". In: *The 2013 World Congress in Computer Science, Computer Engineering, and Applied Computing WORLDCOMP 2013; Las Vegas, Nevada, USA, 22-25 July*. CSREA Press USA. 2013, pp. 67–73.

[29] Larry Harris. *Trading and exchanges: Market microstructure for practitioners*. Oxford University Press, USA, 2003.

[30] Joel Hasbrouck. *Empirical market microstructure: The institutions, economics, and econometrics of securities trading*. Oxford University Press, 2007.

[31] Nicolas Heess et al. "Emergence of locomotion behaviours in rich environments". In: *arXiv preprint arXiv:1707.02286* (2017).

[32] Dieter Hendricks and Diane Wilcox. "A reinforcement learning extension to the Almgren-Chriss framework for optimal trade execution". In: *Computational Intelligence for Financial Engineering & Economics (CIFEr), 2104 IEEE Conference on*. IEEE. 2014, pp. 457–464.

[33] Thomas Ho and Hans R Stoll. "Optimal dealer pricing under transactions and return uncertainty". In: *Journal of Financial economics* 9.1 (1981), pp. 47–73.

[34] Alexander Irpan. *Deep Reinforcement Learning Doesn't Work Yet*. 2018. URL: https://www.alexirpan.com/2018/02/14/rl-hard.html.

[35] Janyl Jumadinova and Prithviraj Dasgupta. "A comparison of different automated market-maker strategies". In: *12th Workshop on Agent-Mediated Electronic Commerce*. 2010, pp. 141–154.

[36] Michael Kearns and Yuriy Nevmyvaka. "Machine learning for market microstructure and high frequency trading". In: *High Frequency Trading: New Realities for Traders, Markets, and Regulators* (2013).

[37] Andrei Kirilenko et al. "The Flash Crash: High-frequency trading in an electronic market". In: *The Journal of Finance* 72.3 (2017), pp. 967–998.

[38] Andrei Kirilenko et al. "The flash crash: The impact of high frequency trading on an electronic market". In: *Available at SSRN* 1686004 (2011).

[39] Chakravanti Rajagopalachari Kothari. *Research methodology: Methods and techniques*. New Age International, 2004.

[40] Blake LeBaron. "Agent-based computational finance". In: *Handbook of computational economics* 2 (2006), pp. 1187–1233.

[41] Yuxi Li. "Deep reinforcement learning: An overview". In: *arXiv preprint arXiv:1701.07274* (2017).

[42] Johann Lussange et al. "A bright future for financial agent-based models". In: *arXiv preprint arXiv:1801.08222* (2018).

[43] Ananth Madhavan. "Market microstructure: A practitioner's guide". In: *Financial Analysts Journal* 58.5 (2002), pp. 28–42.

[44] Ananth Madhavan. "Market microstructure: A survey". In: *Journal of financial markets* 3.3 (2000), pp. 205–258.

[45] Serafin Martinez-Jaramillo and Edward PK Tsang. "Evolutionary computation and artificial financial markets". In: *Natural computing in computational finance*. Springer, 2009, pp. 137–179.

[46] Volodymyr Mnih et al. "Human-level control through deep reinforcement learning". In: *Nature* 518.7540 (2015), p. 529.

[47] John E Moody and Matthew Saffell. "Reinforcement learning for trading". In: *Advances in Neural Information Processing Systems*. 1999, pp. 917–923.

[48] Yuriy Nevmyvaka, Yi Feng, and Michael Kearns. "Reinforcement learning for optimized trade execution". In: *Proceedings of the 23rd international conference on Machine learning*. ACM. 2006, pp. 673–680.

[49] Maureen O'Hara. "High frequency market microstructure". In: *Journal of Financial Economics* 116.2 (2015), pp. 257–270.

[50] Maureen O'hara. *Market microstructure theory*. Vol. 108. Blackwell Publishers Cambridge, MA, 1995.

[51] Alvin Pastore, Umberto Esposito, and Eleni Vasilaki. "Modelling stock-market investors as Reinforcement Learning agents". In: *Evolving and Adaptive Intelligent Systems (EAIS), 2015 IEEE International Conference on*. IEEE. 2015, pp. 1–6.

[52] Matthias Plappert. *keras-rl*. `https://github.com/keras-rl/keras-rl`. 2016.

[53] Donovan Platt and Tim Gebbie. "The Problem of Calibrating a Simple Agent-Based Model of High-Frequency Trading". In: *arXiv preprint arXiv:1606.01495* (2016).

[54] Marco Raberto et al. "Agent-based simulation of a financial market". In: *Physica A: Statistical Mechanics and its Applications* 299.1-2 (2001), pp. 319–327.

[55] Aleksandras Vytautas Rutkauskas and Tomas Ramanauskas. "Building an artificial stock market populated by reinforcement-learning agents". In: *Journal of Business Economics and Management* 10.4 (2009), pp. 329–341.

[56] Michael Schaarschmidt, Alexander Kuhnle, and Kai Fricke. *TensorForce: A TensorFlow library for applied reinforcement learning*. Web page. 2017. URL: `https://github.com/reinforceio/tensorforce`.

[57] John Schulman et al. "Proximal policy optimization algorithms". In: *arXiv preprint arXiv:1707.06347* (2017).

[58] Alexander A Sherstov and Peter Stone. "Three automated stock-trading agents: A comparative study". In: *International Workshop on Agent-Mediated Electronic Commerce*. Springer. 2004, pp. 173–187.

[59] David Silver et al. “Mastering the game of Go with deep neural networks and tree search”. In: *nature* 529.7587 (2016), pp. 484–489.

[60] David Silver et al. “Mastering the game of go without human knowledge”. In: *Nature* 550.7676 (2017), p. 354.

[61] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. Vol. 1. 1. MIT press Cambridge, 2017.

[62] Csaba Szepesvári. “Algorithms for reinforcement learning”. In: (2009).

[63] William M.K. Trochim. *Conclusion Validity*. 2006. URL: https://socialresearchmethods.net/kb/concval.php.

[64] Oriol Vinyals et al. “StarCraft II: a new challenge for reinforcement learning”. In: *arXiv preprint arXiv:1708.04782* (2017).

[65] Huiwei Wang et al. “Reinforcement learning in energy trading game among smart microgrids”. In: *IEEE Transactions on Industrial Electronics* 63.8 (2016), pp. 5109–5119.

[66] Hadley Wickham. *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York, 2009. ISBN: 978-0-387-98140-6. URL: http://ggplot2.org.

[67] Hadley Wickham and Romain Francois. *dplyr: A Grammar of Data Manipulation*. R package version 0.5.0. 2016. URL: https://CRAN.R-project.org/package=dplyr.

[68] Steve Y Yang et al. “Algorithmic trading behavior identification using reward learning method”. In: *Neural Networks (IJCNN), 2014 International Joint Conference on*. IEEE. 2014, pp. 3807–3414.

[69] Steve Yang et al. “Behavior based learning in identifying high frequency trading strategies”. In: *Computational Intelligence for Financial Engineering & Economics (CIFEr), 2012 IEEE Conference on*. IEEE. 2012, pp. 1–8.

Appendix A

Correlation between variables

On the next two pages scatter matrices are shown for Figure A.1 and Figure A.2. The scatter matrices show the correlation between some of the most important variables. These were used when analyzing the data. Also, what the abbreviations mean are the following: av - Ask volume, bv - Bid volume, bp - Bid price, ap - Ask price, mp - Mid price, ara - Ask arrival rate, arb - Bid arrival rate, d_inv - Change in inventory, d_cash - Change in funds, val - Underlying asset price, inv - Inventory $cash$ - Funds.

Looking at the scatter matrix one can see for instance that cash is negatively correlated with both the bid volume and the ask volume. Conversely the agents' cash is positively correlated with both the bid and ask prices.

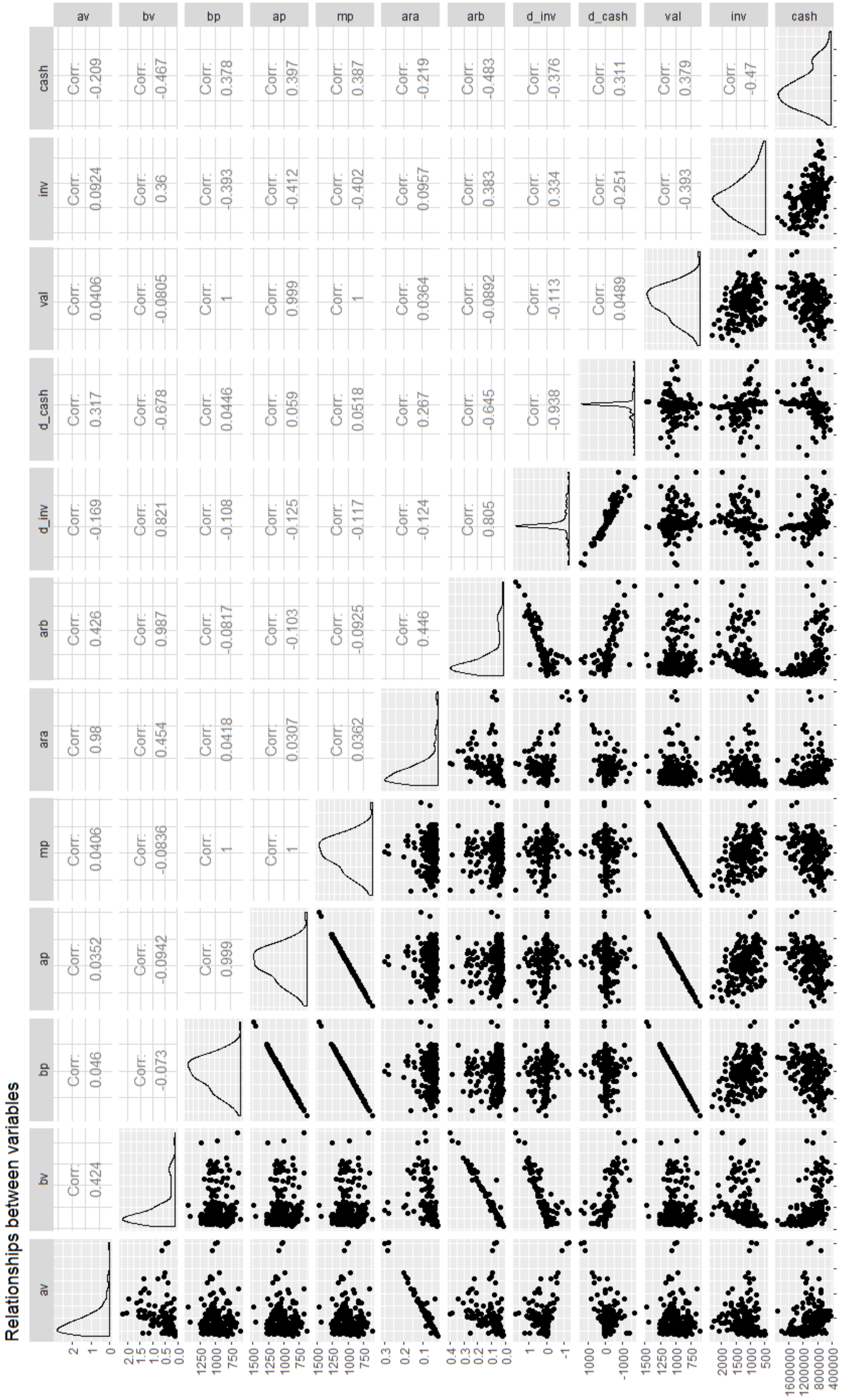


Figure A.1: Scatter matrix between some of the most relevant variables, to find correlations for DealerMarket-v2.

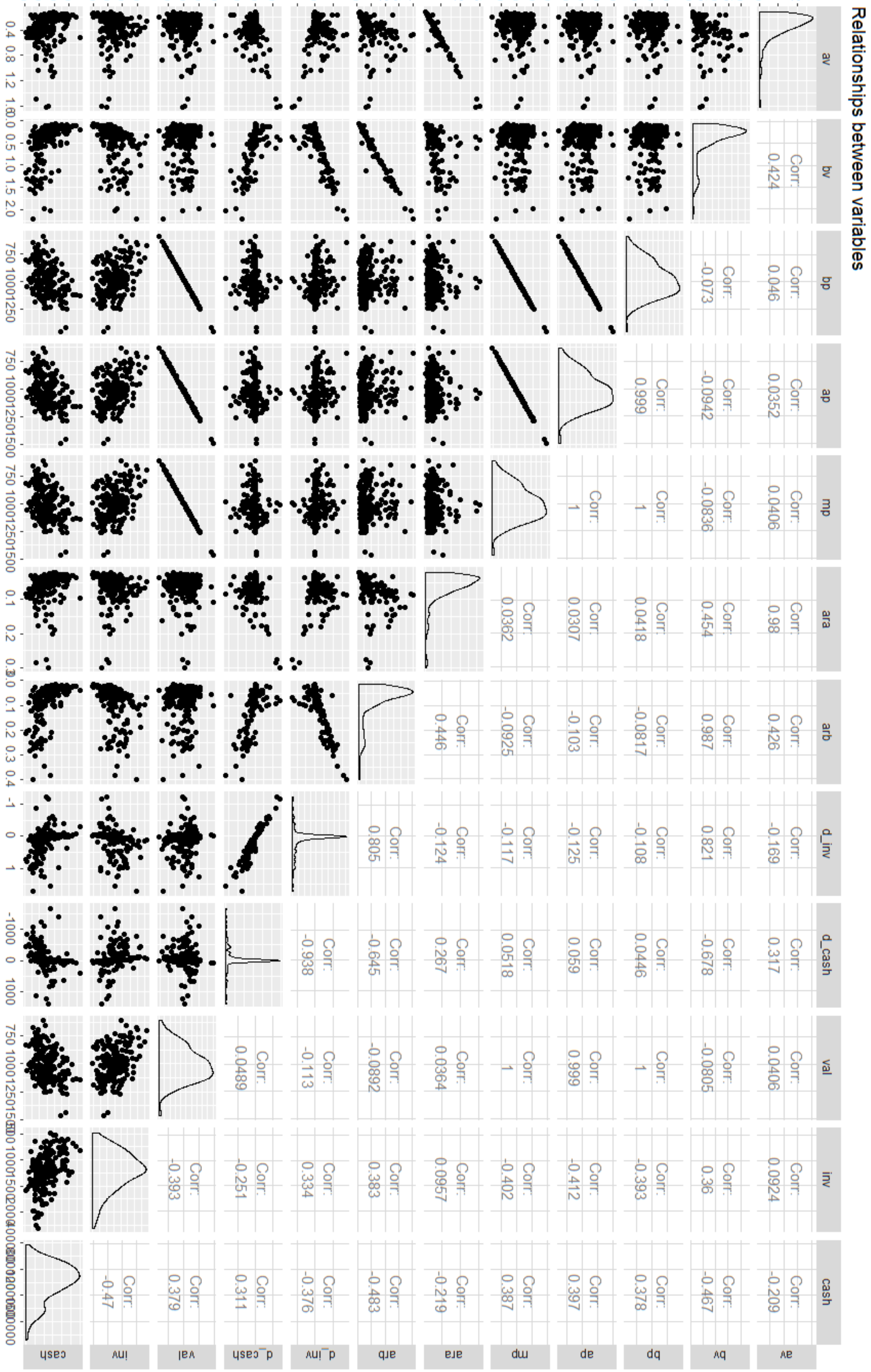


Figure A.2: Scatter matrix between some of the most relevant variables, to find correlations for LOBMarket-v1.

Appendix B

Policy and Value Iteration

In this chapter algorithms using dynamic programming for finding optimal value and optimal policy are shown, based on the material in [61].

B.1 Value Iteration

Algorithm 1: Value iteration

```
Initialize $V$ arbitrarily e.g. $V(s) = 0$ ;
$\Delta \leftarrow 0$ ;
while $\Delta < \theta$ (small positive number) do
    foreach $s \in \mathcal{S}$ do
        $v \leftarrow V(s)$ ;
        $V(s) \leftarrow \max\limits_a \sum_{s',r} p(s', r|s, \pi(s))[r + \gamma V(s')]$ ;
        $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
    end
end
Output: deterministic policy , $\pi \approx \pi_*$ s.t.
        $\pi(s) = \operatorname*{argmax}\limits_a \sum_{s',r} p(s', r|s, a)[r + \gamma V(s')]$
```

B.2 Policy Iteration

Algorithm 2: Policy iteration

```
1. Initialize ;
$V(s) \in \mathbb{R}$ and $\pi(s) \forall s \in \mathcal{S}$ ;
$\Delta \leftarrow 0$ ;
2. Policy Evaluation;
while $\Delta < \theta$ (small positive number) do
    foreach $s \in \mathcal{S}$ do
        $v \leftarrow V(s)$ ;
        $V(s) \leftarrow \sum_{s',r} p(s', r|s, \pi(s))[r + \gamma V(s')]$ ;
        $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
    end
end
3. Policy Improvement;
$policy\_stable \leftarrow true$ ;
while not policy_stable) do
    foreach $s \in \mathcal{S}$ do
        $old\_action \leftarrow \pi(s)$ ;
        $\pi(s) \leftarrow \underset{a}{\operatorname{argmax}} \sum_{s',r} p(s', r|s, a)[r + \gamma V(s')]$ ;
        if $old\_action \neq \pi(s)$ then $policy\_stable \leftarrow false$;
    end
    if $policy\_stable$ then stop;
    else go to 2;
end
return $V \approx v_*$ and $\pi \approx \pi_*$ ;
```

Appendix C

Derivation of Ho & Stoll Model

In this chapter, some explanations to the derivation of the Ho & Stoll model are presented, using dynamic programming, stochastic calculus and the famous Ito's lemma.

C.1 Ito's Lemma

In order to calculate the derivative of a function that depends on time and a stochastic process, *Ito's Lemma* can be used [50]. Suppose that Y is a smooth function:

$$Y = f(x, t) \tag{C.1}$$

where t is time and x is some well-defined Ito processes [50] in Equation C.1 given in Equation C.2:

$$dx = \mu dt + \sigma dz \tag{C.2}$$

then if one wish to maximize Y choosing x, the partial derivatives of Y is needed, which is given by:

$$dY = \frac{\partial f}{\partial t} dt + \frac{\partial f}{\partial x} + \frac{1}{2} \frac{\partial^2 f}{\partial x^2} (dx)^2 = \tag{C.3}$$

$$= \frac{\partial f}{\partial t} dt + \frac{\partial f}{\partial x} [\mu dt + \sigma dz] + \frac{1}{2} \frac{\partial^2}{\partial x^2} \sigma^2 dt \tag{C.4}$$

rewriting Equation C.4 gives Ito's Lemma in Equation C.5:

$$dY = \left[\frac{\partial f}{\partial t} + \frac{\partial f}{\partial x} \mu + \frac{1}{2} \frac{\partial^2 f}{x^2 \sigma^2} \right] dt + \frac{\partial f}{\partial x} \sigma dz \tag{C.5}$$

C.2 Optimal inventory for a Market Maker

In Ho and Stoll [33] a model that handles the risk the market maker faces when providing his service is presented. In the model the following assumptions are made:

- Transactions follow a stationary continuous time stochastic jump process, i.e., a Poisson process.
- The arrival rate of buy orders (λ_a) and sell orders (λ_b) will depend on the dealer's ask and bid prices.
- The dealer face uncertainty over the future value of his portfolio X.

In the the absence of any transactions the portfolio growth dX is given below:

$$dX = r_x X dt + X dZ_x \tag{C.6}$$

where r_x is the mean return, dZ_x is the Wiener process with mean zero, variance σ_X^2. The dealers wealth is divided into three components: *cash, inventory* and *base wealth*. The value of the cash account (F) is:

$$dF = rFdt - (p - b)dq_b + (p + a)dq_a \tag{C.7}$$

which changes with buys and sells of securities earning the risk-free rate r. The dealers' inventory (I) is given by:

$$dI = r_I I dt + pdq_b - pdq_a + IdZ_I \tag{C.8}$$

The inventory consists of shares in the stock the market maker makes or provides liquidity to. Finally, base wealth (Y) is given by:

$$dY = r_Y Y dt + Y dZ_Y \tag{C.9}$$

The objective of the dealer is now to maximize the expected utility of his total wealth $E[U(W_T)]$ at time horizon T, where

$$W_T = F_T + I_T + Y_T \tag{C.10}$$

Equation 2.9 is what is termed *the dealers pricing problem*. This is in fact an optimization problem with the goal to maximize the value function $J(\cdot)$ using dynamic programming. Thus, yielding the optimization problem below in Equation C.11:

$$J(t, F, I, Y) = \max_{a,b}[E[U(W_T)]|t, F, I, Y] \tag{C.11}$$

where U is the utility function, a and b are the ask and bid adjustments and t, F, I, Y are the states variables time, cash, inventory and base wealth [50]. The function $J(\cdot)$ gives the level of utility given that the dealer's decisions are made optimally [50]. As, there are no intermediate consumption before time T in this model, the recurrence relation found by using the principle of optimality is:

$$\max_{a,b} dJ(t, F, I, Y) = 0 \text{ and } J(T, F, I, Y) = U(W_T) \tag{C.12}$$

solving Equation C.12 finds a solution to the dealer's problem, where one have to find the ask and bid prices for each state. To solve Equation C.12 one requires to use stochastic calculus, and by writing out the partial differential equations that Equation C.12 implies and applying Ito's Lemma in Equation C.5:

$$\begin{aligned} \underset{a,b}{max}(dJ/dt) =& J_t + LJ \\ &+ max\{\lambda_a[J(F + pQ + aQ, I - pQ, Y) - J(F, I, Y)] \\ &+ \lambda_b[J(F - pQ + bQ, I + pQ, Y) - J(F, I, Y)]\} = 0 \end{aligned} \tag{C.13}$$

where J_t is the time derivative and LJ is the operator defined as

$$LJ = J_F rF + J_I r_I I + J_Y r_y Y + \frac{1}{2}J_{II}\sigma_I^2 I^2 + \frac{1}{2}J_{YY}\sigma_Y^2 Y^2 + J_{IY}\sigma_{IY} IY \tag{C.14}$$

$J_t + LJ$ is the total time derivative of derived utility when there are no transactions. Equation C.13 determines the solution, which is hard to solve explicitly, and Ho and Stoll [33] do not solve the general problem but introduces some transformations and simplifications in order to solve it. Firstly, by looking at the problem only at the endpoint (τ) where it is is equal to zero. Secondly, by taken the first-order approximation of the Taylor series expansion of Equation C.11 [50]. Ho and

Stoll [33] then, also define two new operators the sell (SJ) and buy (BJ) operators:

$$SJ = S[J(F,I,Y)] = J(F+Q, I-Q, Y) \tag{C.15}$$
$$BJ = B[J(F,I,Y)] = J(F-Q, I+Q, Y) \tag{C.16}$$

By using the new operators the problem in Equation C.13 can be rewritten as:

$$\begin{aligned} J_t =& LJ + \max_{a,b}\{[\lambda(a)aQSJ_F - \lambda(a)(J-SJ)]+ \\ & [\lambda(b)bQBJ_F - \lambda(b)(J-BJ)]\} \end{aligned} \tag{C.17}$$

where $\lambda(a) = \alpha - \beta_A$ and $\lambda(b) = \alpha + \beta_B$ are symmetric linear supply and demand functions to the dealer. There is no closed form solution for this problem, nonetheless via approximations the bid and ask quotes can be found below:

$$b^* = \alpha/2\beta + (J-BJ)/2BJ_FQ \tag{C.18}$$
$$a^* = \alpha/2\beta + (J-SJ)/2SJ_FQ \tag{C.19}$$

Finally, from Equation 2.12 and Equation 2.13 the the bid-ask spread is:

$$s = \alpha/\beta + (J-SJ)/2SJ_FQ + (J-BJ)/2BJ_FQ \tag{C.20}$$

The first term of Equation C.20 is the spread which maximizes the expected returns from selling and buying stocks. Whilst the rest of the terms are seen as *risk premiums* for sale and purchase transactions. This as the dealer or market maker sets the spread without knowing what side the transaction will have, i.e., bid or ask [33].

Ho and Stoll [33] demonstrates three important properties of the dealer's optimal pricing behavior:

1. The spreads depends on the time horizon of the dealer
2. The spread can be decomposed into a risk neutral spread plus an adjustment for uncertainty
3. The spread is independent of inventory level.

C.3 What human dealers would do

Dealer or market makers are profit-motivated traders who allow other traders to trade when they want to trade [29]. As, mentioned before one way of finding optimal bid-and ask quotes for inventory control is to use the Ho and Stoll [33] model. However, what would a human trader do reacting to different conditions on the market? In Harris [29] this is presented, shown below in Table C.1:

Condition	Tactics	Purpose
Inventories are too low or clients are net buyers	Raise bid price Increase bid size	Encourage clients to sell
	Raise ask price decrease ask size	Discourage clients from buying
Inventories are too high or clients are net sellers	Lower ask price Increase ask size	Encourages clients to buy
	Lower ask price Increase ask size	Discourage clients from selling

Table C.1: Tactics Dealers or Market Makers Use to Manage Their Inventories and Orders Flow. Adopted from [29]

TRITA -EECS-EX