REINFORCEMENT LEARNING FOR MARKET MAKING

CARLSSON, SIMON REGNELL, AUGUST

Degree Project in Financial Mathematics (30 ECTS credits)
Master's Programme in Industrial Engineering and Management
KTH Royal Institute of Technology year 2022
Supervisor at Skandinaviska Enskilda Banken: Hanna Hultin
Supervisor at KTH: Henrik Hult

A B S T R A C T

Market making – the process of simultaneously and continuously providing buy and sell prices in a financial asset – is rather complicated to optimize. Applying reinforcement learning (RL) to infer optimal market making strategies is a relatively uncharted and novel research area. Most published articles in the field are notably opaque concerning most aspects, including precise methods, parameters, and results. This thesis attempts to explore and shed some light on the techniques, problem formulations, algorithms, and hyperparameters used to construct RL-derived strategies for market making. First, a simple probabilistic model of a limit order book is used to compare analytical and RL-derived strategies. Second, a market making agent is trained on a more complex Markov chain model of a limit order book using tabular Q-learning and deep reinforcement learning with double deep Q-learning. Results and strategies are analyzed, compared, and discussed. Finally, we propose some exciting extensions and directions for future work in this research field.

KEYWORDS: *Reinforcement learning, Market making, Deep reinforcement learning, Limit order book, Algorithmic trading, High-frequency trading, Quantitative finance, Financial mathematics, Machine learning, Artificial intelligence, Q-learning, DDQN*

FÖRSTÄRKNINGSINLÄRNINGSBASERAD LIKVIDITETSGARANTERING

SAMMANFATTNING

Likviditetsgarantering (eng. ”market making”) – processen att simultant och kontinuerligt kvotera köp- och säljpriser i en finansiell tillgång – är förhållandevis komplicerat att optimera. Att använda förstärkningsinlärning (eng. ”reinforcement learning”) för att härleda optimala strategier för likviditetsgarantering är ett relativt outrett och nytt forskningsområde. De flesta publicerade artiklarna inom området är anmärkningsvärt återhållsamma gällande detaljer om de tekniker, problemformuleringar, algoritmer och hyperparametrar som används för att framställa förstärkningsinlärningsbaserade strategier. I detta examensarbete så gör vi ett försök på att utforska och bringa klarhet över dessa punkter. Först används en rudimentär probabilistisk modell av en limitorderbok som underlag för att jämföra analytiska och förstärkningsinlärda strategier. Därefter brukas en mer sofistikerad Markovkedjemodell av en limitorderbok för att jämföra tabulära och djupa inlärningsmetoder. Till sist presenteras även spännande utökningar och direktiv för framtida arbeten inom området.

ACKNOWLEDGMENTS

We would like to express our sincerest gratitude to our supervisor at SEB, Hanna Hultin, for her excellent guidance, insights, and codebase provided. Without her help and support during the past months, this thesis would not have been possible. It has truly been a pleasure to work with someone so knowledgeable on this topic.

We would also like to extend our gratitude to the FX quantitative trading (especially Simon Österberg) and the data science teams at SEB for sharing their industry expertise with us.

We also want to express our gratitude toward SEB for providing us with office space, computers, cloud computing services, and – most importantly – unlimited access to your coffee machines.

Lastly, we would like to thank Henrik Hult, our supervisor at KTH, for providing valuable guidance and support in terms of thesis-writing expertise and the MC model used in the thesis.

Simon Carlsson[1] & August Regnell[2]
Stockholm, May 24, 2022

[1] simoncarlsson@outlook.com
[2] august.regnell@gmail.com

DISCLAIMER

The content of this thesis is for informational purposes only. It is not intended as investment research or investment advice, or a recommendation, offer, or solicitation for the purchase or sale of any security, financial instrument, financial product, or service, or to be used in any way for evaluating the merits of participating in any transaction. We, the authors, do not accept any liability for any investment decisions based on statements of this thesis.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACRONYMS

DDQN	Double Deep Q-Network
DQN	Deep Q-Network
DRL	Deep Reinforcement Learning
FX	Foreign Exchange
GCP	Google Cloud Platform
HFT	High-Frequency Trading
LH	Long-Horizon
LH-DDQN	Long-Horizon for Double Deep Q-learning
LH-Q	Long-Horizon for Q-learning
LO	Limit Order
LOB	Limit Order Book
MC	Markov Chain
MDP	Markov Decision Process
MiFID	Markets in Financial Instruments Directive
MM	Market Maker
MO	Market Order
NYSE	New York Stock Exchange
PnL	Profit and Loss
RL	Reinforcement Learning
SEB	Skandinaviska Enskilda Banken
SH	Short-Horizon
SH-DDQN	Short-Horizon for Double Deep Q-learning
SH-Q	Short-Horizon for Q-learning
SPM	Simple Probabilistic Model

SUMMARY OF NOTATION

Here is a summary of the notation used in this thesis, sorted by appearance and categorized by the section in which the notation was first introduced. This summary is not entirely exhaustive as we have chosen to omit most standard mathematical notations and most constants.

THE LIMIT ORDER BOOK

x	An order
p_x	Posted price of order x
ω_x	Volume of order x
t_x	Submission time of order x
τ_x	Order type of order x
$\mathcal{L}(t)$	Limit order book at time t
$\mathcal{B}(t)$	Set of all buy orders in the LOB at time t
$\mathcal{A}(t)$	Set of all sell orders in the LOB at time t
δ	Tick size or pip
$b(t)$	Bid price at time t
$a(t)$	Ask price at time t
$s(t)$	Bid-ask spread at time t
$m(t)$	Mid price at time t
$q^b(t)$	Bid volume at time t
$q^a(t)$	Ask volume at time t
$q^p(t)$	Volume on price level p at time t
$\mu(t)$	Micro price at time t
$\rho_n(t)$	Order imbalance for n levels at time t
S_t	Mid price process at time t
W_t	Standard Brownian motion or Wiener process
$M_t^{\pm}$	Market order arrival processes
$\delta^{\pm}$	Bid $(-)$ and ask $(+)$ depths ($\delta_t^{\pm}$ is the same quantities at time t)
d	Number of price levels in vector representation of LOB (MC model)
$\boldsymbol{S}_t$	Markov chain state at time t (random variable)

$\boldsymbol{s}_t$	Markov chain state at time t (observation)
$\mathcal{S}$	State space in a Markov chain or Markov decision process

REINFORCEMENT LEARNING

$\mathcal{A}$	Action space in a Markov decision process
A_i	Action on step i in Markov decision process
$\mathcal{R}$	Reward space
R_i	Reward on step i in Markov decision process
$\pi(\cdot \mid \boldsymbol{s})$	Policy in state $\boldsymbol{s}$
$v_\pi(\cdot)$	State-value under policy π
$q_\pi(\cdot, \cdot)$	Action-value under policy π
γ	Discount factor
$Q(\cdot, \cdot)$	Action-value in Q-learning algorithm
$a_*(\cdot)$	Optimal action
$\hat{v}(\cdot; \cdot)$	Approximated state-value
$\hat{q}(\cdot, \cdot; \cdot)$	Approximated action-value

OPTIMAL MARKET MAKING

$N_t^{\pm}$	Counting process of filled limit orders
X_t	Cash process
Q_t	Inventory process
$\underline{q}$	Maximum short position allowed
$\overline{q}$	Maximum long position allowed
T	Time horizon in a market making episode
V_t	Value process

RESULTS

H_t	Holding value process
$\overline{r}$	The average reward received during testing of a strategy
σ_r	The standard deviation of the reward during testing of a strategy
dt	Size of time step
Δ	Order depth shortfall
$p^L(\cdot, \cdot)$	VWAP liquidation price in the limit order book $\mathcal{L}(t)$ for quantity Q_t, at time t
$\bar{d}$	Maximum order depth

1 INTRODUCTION

1.1 BACKGROUND

A market maker (MM) is a market participant that simultaneously and continuously provides buy and sell prices of a financial asset to provide liquidity to the market and generate a profit off the spread – a process called *market making*. The core objectives of a market maker are to (1) maximize profits and (2) limit the risk of holding too large of an inventory, which in the event of adverse market movements may inflict large losses. [48]

One can achieve optimal market making by maximizing a (risk-adjusted) return measure, which combines the two core objectives above. Typically, the optimal quotes are a function of the market maker's inventory levels and prevailing market conditions [25]. Traditionally, the problem of optimal market making has been modeled with stochastic differential equations based on strict (and often naive) assumptions and solved with classical methods for optimal control, such as dynamic programming [11]. Perhaps the most famous undertaking in this setting is the Avellaneda and Stoikov model from 2007 [6].

Advances in artificial intelligence (AI) and computer processing have enabled a new paradigm in optimal market making to gain attention and popularity in academia and the financial industry. Methods belonging to the specific branch within AI called machine learning (ML) – automated analytical model building – are used to teach an agent how to make markets optimally. These methods remedy many of the restrictions and downsides with analytically obtained strategies.

The specific class of machine learning methods used is reinforcement learning (RL), which involves an agent learning from experience interacting in a stochastic environment [68]. Applying RL methods for optimal market making is a relatively novel, albeit highly promising, endeavor. Indeed, there seems to exist a consensus in the academic community that RL approaches to optimal market making are superior to standard market making strategies. [25]

Since market making naturally can be framed as a Markov decision process, reinforcement learning lends itself particularly well for it [25]. While vanilla reinforcement learning methods typically cannot handle problems with large state and/or action spaces (like market making with extensive market data), recently developed deep reinforcement learning (DRL) techniques can circumvent these flaws by approximating e.g., the state-action values with neural networks [5].

Notably, the agent in reinforcement learning needs to interact with the environment for it to learn, which is not possible with historical data, which is why simulations of the market have been used, such as Markov and RNN (recurrent neural network) models [41]. Several authors have successfully used generative models to find optimal market making strategies with DRL methods such as *Deep Q-Learning* (DQN) and *Double Deep Q-Learning* (DDQN). However, one should also note that some success has been found with non-deep models. [25]

This thesis will explore Q-learning and DDQN in two ways of modeling the market: a simple probabilistic model from [11] and a Markov chain model from [39].

1.2 OBJECTIVE

This thesis aims to explore and shed some light on the relatively novel area of reinforcement learning for market making. Specifically, this involves (1) examining different models for the underlying asset and the limit order book, (2) framing the optimal market making problem in different ways, (3) assessing feasible and reasonable state and action spaces, (4) finding functioning reward signals, (5) comparing the performance of RL-methods and analytically derived strategies. Specifically, we are keen to investigate and answer:

- **Research Question 1 (RQ1):** Are reinforcement learning methods able to outperform analytically derived strategies?
- **Research Question 2 (RQ2):** Can deep reinforcement learning algorithms learn better performing market making strategies than tabular Q-learning?

1.3 DELIMITATIONS

This thesis does not aim to be an exhaustive account or examination of reinforcement learning for market making. Instead, only a small subset of possible combinations of order book models, reinforcement learning methods, and problem settings are considered. In our work, we primarily consider a static Markov chain limit order book model in which we let a single market maker agent interact. The Markov chain model jointly and implicitly models the actions of other market participants. Additionally, we employ only Q-learning and DDQN – one tabular and one deep technique – out of all existent (D)RL methods. For tabular learning methods, the state space is severely curtailed to increase the learning speed, thus forfeiting information that the agent could use to infer better strategies. Furthermore, a constant and simplistic action space is considered in all experiments, erasing the possibility for more sophisticated and granular strategies. Additionally, only risk-neutral reward formulations are considered, seriously dampening the learning of risk-averse (and in practice probably more desirable) strategies.

The static modeling of the market severely limits the performance of the learned strategies in a real market, which is why the findings of this thesis serve only to demonstrate and highlight frameworks that can be used in reinforcement learning applications for market making.

1.4 OUTLINE

The remaining part of this thesis is split into five sections structured in the following way:

- **Section 2**: Theory and Related Work
- **Section 3**: Experiments and Results
- **Section 4**: Discussion and Future Work
- **Section 5**: Conclusion
- **Appendices**: Derivations and Additional Results

Section 2 introduces the relevant theory and previous work on similar topics. The performed experiments are presented in a story-telling fashion in *Section 3*, with a summary of the results at the end of the section. Thitherto unmentioned observations and comparisons are discussed further

in *Section 4*, followed by a conclusion in *Section 5*. The interested reader can find derivations and additional results that are not essential to the thesis in the *Appendices*.

1.5 GITHUB

For the reader interested in exploring our work and material on GitHub, we refer to our repository `Reinforcement Learning for Optimal Market Making`.[1] The code is split into three main categories:

1. Code used to simulate the LOBs and environments.
2. Code used to train the reinforcement learning agents.
3. Code used to generate tables and graphs used to evaluate strategies.

[1] https://github.com/KodAgge/Reinforcement-Learning-for-Market-Making

2

THEORY AND RELATED WORK

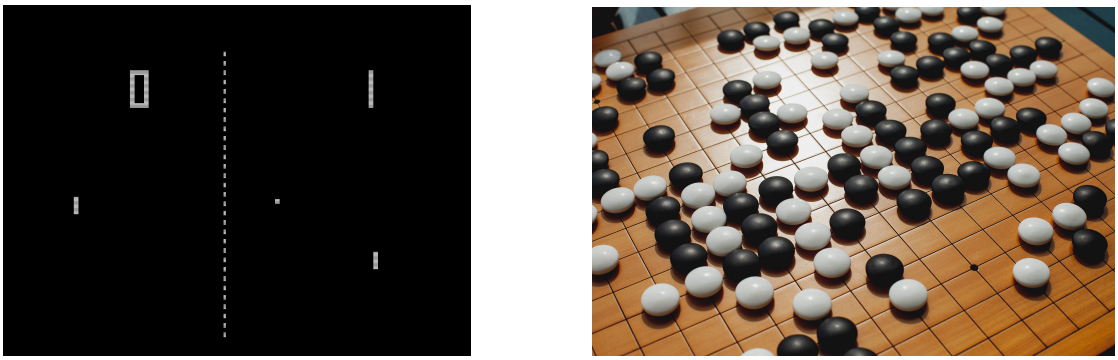
This chapter is divided into four sections and introduces the reader to theoretical frameworks, machine learning methods, financial settings, and previous works related to our thesis. The first section covers both tabular and deep reinforcement learning techniques. Next, the second section covers limit order books, including their description and how we model them in this thesis. The third section covers optimal market making, analytical solutions, and related work on reinforcement learning and market making. Finally, the fourth and last section covers some evaluation metrics used in this thesis.

2.1 REINFORCEMENT LEARNING

Reinforcement learning (RL) originates from the idea of how we as humans learn new things – by trial and error. By trying new actions, thus interacting with our environment, we observe the consequences and receive a reward, e.g., points in a game, the feeling of satisfaction, or surviving a dangerous situation. Subsequently, we learn what constitutes a desirable behavior to receive these rewards.

In the world of machine learning, one has taken a computational approach to this type of learning, idealizing the learning situations and finding different learning methods. Reinforcement learning is thus the process of learning what to do in different situations – mapping the optimal action (in terms of the expected reward) to different states [68].

RL has been used successfully in many tasks and has lately exploded in popularity due to its very promising synergies with deep learning (DL) – see Section 2.1.4 for a short description. The first uses have been for tasks such as assigning credit [54], but lately, RL has been used to outperform humans on very complex tasks such as Atari games [55] and the board game Go [63]. Illustrations of these games are shown in Figure 2.1.



(a) (b)

Figure 2.1: **(a)** the Atari game *Pong* and **(b)** the board game *Go*.

2.1.1 *Markov Decision Processes*

Markov decision processes (MDPs) are used to frame the problem of learning from the interaction with an environment to achieve a goal. The *agent* is the learner and the one making decisions. It interacts with the *environment*, which comprises everything outside the agent. The dynamics consist of the agent and environment interacting continually where the environment reacts to the agent's action and provides new situations. After every action, the agent also receives a *reward*, a numerical value, which it seeks to maximize. This whole section about MDPs is based on [68], except where other sources are cited.

More formally, the interaction occurs at discrete time steps $t \geq 0$, the number of which can be both finite and infinite. The agent receives a representation of the environment's *state* at every time step t, $\boldsymbol{s} = \boldsymbol{S}_t \in \mathcal{S}$.[1] Based on the state, the agent chooses an action $a = A_t \in \mathcal{A}(\boldsymbol{s})$. After the action, in the next time step, the agent receives a reward $r = R_{t+1} \in \mathcal{R}$, and the state is updated to $\boldsymbol{s}' = \boldsymbol{S}_{t+1}$. This dynamic thus gives rise to a sequence

$$\{\boldsymbol{S}_i, A_i, R_{i+1}\}_{i=0}^{n}. \tag{1}$$

The transition probability from state $\boldsymbol{s} = \boldsymbol{S}_t$ after the agent taking the action $a = A_t$ to state $\boldsymbol{s}' = \boldsymbol{S}_{t+1}$ and reward $r = R_{t+1}$ is given by

$$p(\boldsymbol{s}', r \mid \boldsymbol{s}, a) := \mathbb{P}(\boldsymbol{S}_{t+1} = \boldsymbol{s}', R_{t+1} = r \mid \boldsymbol{S}_t = \boldsymbol{s}, A_t = a). \tag{2}$$

A *policy*, $\pi(\cdot \mid \boldsymbol{s})$, is a mapping from all states to the probabilities of selecting each possible action. Based on a policy π, one can calculate the *value* of a state $\boldsymbol{s}$, that is the expected return when following the policy π after starting in state $\boldsymbol{s}$. This is defined as v_π, the *state-value*

$$v_\pi(\boldsymbol{s}) := \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid \boldsymbol{S}_t = \boldsymbol{s}\right], \forall \boldsymbol{s} \in \mathcal{S}, \tag{3}$$

where $0 < \gamma \leq 1$ is a discount factor. The discount factor has to be $\gamma < 1$ in order to ensure convergence for an infinitely long episode. However, it can be put to one when the episodes are finite.

On a similar note, the value of taking action a in state $\boldsymbol{s}$ and then following π, the *action-value* is defined in [68] as

$$q_\pi(\boldsymbol{s}, a) := \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid \boldsymbol{S}_t = \boldsymbol{s}, A_t = a\right]. \tag{4}$$

The state-value in Equation (3) can be re-written as

$$v_\pi(\boldsymbol{s}) := \mathbb{E}_\pi\Big[R_{t+1} + \gamma v_\pi(\boldsymbol{S}_{t+1}) \mid \boldsymbol{S}_t = \boldsymbol{s}\Big], \tag{5}$$

which is called the *Bellman equation*. The optimal state-value is $v_*(\boldsymbol{s}) = \sup_\pi v_\pi(\boldsymbol{s})$, and a policy π_* is called optimal if $v_*(\boldsymbol{s}) = v_{\pi_*}(\boldsymbol{s})$, $\forall \boldsymbol{s} \in \mathcal{S}$. The *Bellman optimality equation* is thus

[1] Note that we denote the state in bold, contrary to how for instance, Sutton and Barto do it in [68]. This is partly because we only use multi-dimensional states throughout this thesis, and the bold notation may thus be considered more appropriate, and partly to distinguish the state notation-wise from the mid price process introduced in the simple probabilistic model.

$$v_*(\boldsymbol{s}) := \sup_{a \in \mathcal{A}(\boldsymbol{s})} \mathbb{E}_\pi \Big[R_{t+1} + \gamma v_*(\boldsymbol{S}_{t+1}) \mid \boldsymbol{S}_t = \boldsymbol{s}, A_t = a \Big]. \tag{6}$$

Similarly, the Bellman optimality equation for the action-value is defined in [38] as

$$q_*(\boldsymbol{s}, a) := \mathbb{E} \Big[R_{t+1} + \gamma \sup_{a'} q_*(\boldsymbol{S}_{t+1}, a') \mid \boldsymbol{S}_t = \boldsymbol{s}, A_t = a \Big]. \tag{7}$$

2.1.2 *Tabular Learning and Action-Value Methods*

One of the most central ideas in reinforcement learning is *temporal-difference* (TD) learning. This technique combines ideas from Monte Carlo methods and dynamic programming and can thus learn directly from experience without knowing the environment's underlying dynamics.

Two of the most popular TD learning methods are *SARSA* and *Q-learning*, which share the same goal of approximating the Q-matrix for all state-action pairs $(\boldsymbol{s}, a) \in (\mathcal{S}, \mathcal{A}(\boldsymbol{s}))$. This is why these methods are called *tabular learning methods*; they estimate the "*Q-table*." The two methods differ slightly; SARSA needs to simulate five values per estimation and Q-learning only four. Q-learning also directly approximates the optimal action-value function q_*. [68]

The largest problems for these methods occur when the action and/or the state space grows too large. It becomes infeasible to visit every state-action pair enough times to warrant an accurate estimate of their values. Also, the learned values cannot be generalized in any way; knowing the values for 99 of the 100 state-action pairs does not tell anything about the 100th value. Another big problem is the *exploration-exploitation tradeoff* (not exclusive to tabular methods) – should the agent exploit the currently best action or explore worse actions that may lead to better results in the long run? One way of including a balance is to use an ϵ-greedy policy which gives the probability $1 - \epsilon + \epsilon/|\mathcal{A}(\boldsymbol{s})|$ to the greedy action and probability $\epsilon/|\mathcal{A}(\boldsymbol{s})|$ to all other actions [38]. Here $|\mathcal{A}(\boldsymbol{s})|$ denotes the cardinality of $\mathcal{A}(\boldsymbol{s})$.

2.1.2.1 *Q-Learning*

Q-learning is an off-policy TD learning algorithm created by Watkins in 1989. The update equation is given by

$$Q(\boldsymbol{S}_t, A_t) \leftarrow Q(\boldsymbol{S}_t, A_t) + \alpha \Big[R_{t+1} + \gamma \max_a Q(\boldsymbol{S}_{t+1}, a) - Q(\boldsymbol{S}_t, A_t) \Big], \tag{8}$$

where α is the learning rate. Equation (8) is a version of the Robbins-Monro algorithm. It's called *off-policy* due to the *max* in Equation (8) – instead of following the policy and using the generated A_{t+1} (as in SARSA) it chooses the optimal action. This, combined with the following requirements on the sequence of α – $\alpha_t > 0$, $\alpha_t \to 0$, $\sum \alpha_t = \infty$, and $\sum \alpha_t^2 < \infty$ – i.e., the usual conditions required by the Robbins-Monro algorithm (see for instance [47]), means that the algorithm will converge to the optimal action-value function q_* with probability 1 given that the environment is an MDP [52, 68].

It is common to use ϵ-greedy policy for Q-learning, ϵ is the exploration rate, that is the probability that a random action is chosen. In this case, $A_t \sim \pi_t(\cdot|\boldsymbol{s})$ in Equation (8) is chosen from $\boldsymbol{s} = \boldsymbol{S}_t$ based on the ϵ-greedy policy defined in [38] as follows,

$$\pi_t(a|\boldsymbol{s}) = \begin{cases} 1 - \epsilon + \epsilon/|A(\boldsymbol{s})|, & \text{if } a = a_*(\boldsymbol{s}), \\ \epsilon/|A(\boldsymbol{s})|, & \text{if } a \neq a_*(\boldsymbol{s}), \end{cases} \tag{9}$$

where

$$a_*(\boldsymbol{s}) = \arg\sup_{a \in \mathcal{A}(\boldsymbol{s})} q_t(\boldsymbol{s}, a). \tag{10}$$

2.1.3 *Deep Learning*

Deep learning (DL) is a type of machine learning that uses *artificial neural networks* (ANNs) as a nonlinear function approximation. They consist of networks of interconnected units that replicate neurons found in the nervous systems of animals. While there are many sophisticated and complex architectural structures of ANNs, the most known is the generic feedforward network which has no loops. An illustration of such a network can be seen in Figure 2.2. Training of such a network consists of updating the weights and biases of the neurons to better predict the wanted output. The updating is usually done with stochastic gradient descent (SGD) with mini-batches. A mini-batch of size n consists of n randomly drawn data points over which the gradient is calculated. [68]

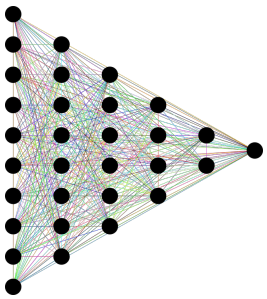


Figure 2.2: An illustration of an artificial neural network.

In a feedforward network like the one depicted above, the *layers* (i.e., the vertical alignments of nodes in the figure) are commonly *linear* and coupled with an activation function. This means that the input value at a node is a sum of affine transformations (with no non-linearity) of the previous layer's output values. Every layer is usually represented by a function $f^{(i)}(h^{(i-1)}) = g^{(i)}(W^{(i)}h^{(i-1)} + b^{(i)})$ where $h^{(i-1)}$ is the value at the $(i-1)$th layer, $W^{(i)}$ is the weight matrix of the ith layer, $b^{(i)}$ is the bias vector of the ith layer and $g^{(i)}$ is the activation function of the ith layer. A full network can thus be represented as a composition of all n layers, $f(x) = f^{(n)} \circ \cdots \circ f^{(1)}(x)$. [38]

In this thesis the ReLU activation function is used, i.e., the function $g(x) = x_+$. The Adam algorithm (*adaptive moment estimation*), a stochastic gradient descent algorithm developed by [46], is used to train the neural networks in this thesis. Adam has gained immense popularity due to its effective handling of non-convex optimization problems. In large neural networks, a common problem is exploding and/or vanishing gradients. Using the *batch normalization transform* between layers, this can be combated. It first normalizes the input with its mean and variance, and then it scales the output with a factor γ and a shift β – two learnable parameters. [42] Specifically, at each hidden layer, the signal is transformed as below:

$$\widetilde{\boldsymbol{Z}} = \boldsymbol{Z} - \frac{1}{n}\sum_{i=1}^{n} \boldsymbol{Z}_i, \qquad \widehat{\boldsymbol{Z}} = \frac{\widetilde{\boldsymbol{Z}}}{\sqrt{\epsilon + \frac{1}{n}\sum_{i=1}^{n} \widetilde{\boldsymbol{Z}}_i^2}}, \qquad \boldsymbol{H} = \max\{0, \gamma\widehat{\boldsymbol{Z}} + \beta\}, \tag{11}$$

where $\boldsymbol{Z} = \{\boldsymbol{Z}_i\}_{i=1}^n$ are the values of the mini-batch, ϵ is a constant added to the mini-batch variance for numerical stability and $\boldsymbol{H}$ is the output of the batch-normalization layer.

2.1.4 *Deep Reinforcement Learning*

So far, we have focused on methods that aim to find the actual state-values and action-values; a different method is to approximate these with an appropriate function instead. We may instead assume that these functions are represented by parameterized functional forms instead of tables. That is $\hat{v}(\boldsymbol{s};\boldsymbol{\theta}) \approx v_\pi(\boldsymbol{s})$ and $Q(\boldsymbol{s}, a; \boldsymbol{\omega}) \approx q_*(\boldsymbol{s}, a)$ where $\boldsymbol{\theta}, \boldsymbol{\omega} \in \mathbb{R}^d$ are the parameters that parameterize these functions and d is the number of parameters. [68] For parameterized Q-learning with the value function $Q(\boldsymbol{s}, a; \boldsymbol{\theta})$ where $\boldsymbol{\theta}$ are the parameters, the update equation becomes, by [34],

$$\boldsymbol{\theta}_{i+1} = \boldsymbol{\theta}_i + \alpha(Y_i^Q - Q(\boldsymbol{S}_i, A_i; \boldsymbol{\theta}_i))\nabla_{\boldsymbol{\theta}_i} Q(\boldsymbol{S}_i, A_i; \boldsymbol{\theta}_i), \tag{12}$$

where Y_i^Q is the target and defined as

$$Y_i^Q = R_{i+1} + \gamma \max_a Q(\boldsymbol{S}_{i+1}, a, \boldsymbol{\theta}_i). \tag{13}$$

By adopting this approach, one overcomes the underlying issue of the generalization of tabular methods. When the state and/or action space is large enough, it is infeasible to visit every possible state, and the agent cannot generalize from previous similar encounters. Using a function approximation, one effectively reduces the number of values that has to be learned from $|\mathcal{A}| \cdot |\mathcal{S}|$ to $\dim(\boldsymbol{\theta})$ ($\boldsymbol{\theta}$ is the parameters parameterizing the approximating function), which is a significant reduction of the time and data complexity of the learning algorithm.

Many feasible function-approximation include *linear methods* (e.g., *polynomials*, *Fourier basis*, *coarse coding*, *tile coding*, and *radial basis functions*), *non-linear methods* (predominantly ANNs), and *kernel methods*. These can be split into *on/off-policy* and *prediction/control* (state- or action-value) methods [68]. We will focus on off-policy control approximation methods based on deep learning functional forms, e.g., *Deep Q-learning* (DQN) and *Double Deep Q-learning* (DDQN).

2.1.4.1 *Deep Q-Network*

In the famous paper *Playing Atari with Deep Reinforcement Learning*, Mnih et al. introduced a new variant of the Q-learning algorithm using convolutional neural networks.

They point out the impractical approach of tabular learning based on the Bellman equation, Equation (4) – the action-value function is estimated separately for every state, and there is no generalization at all. They instead define a neural network function approximator $Q(\boldsymbol{s}, a; \boldsymbol{\theta}^-) \approx Q^*(\boldsymbol{s}, a)$, parameterized by $\boldsymbol{\theta}^-$, called a Deep Q-Network (DQN) or the *target network*. The target network is a periodic copy of the *online network* parameterized by $\boldsymbol{\theta}$ that is updated every step – that is, the target network is updated to equal the online network such that $\boldsymbol{\theta}_i^- = \boldsymbol{\theta}_i$ every τ steps. The online network is trained by minimizing the loss function $L_i(\boldsymbol{\theta}_i)$ defined as

$$L_i(\boldsymbol{\theta}_i) = \mathbb{E}_{\boldsymbol{s},a\sim\rho(\cdot)}\Big[(y_i - Q(\boldsymbol{s},a;\boldsymbol{\theta}_i))^2\Big], \tag{14}$$

where $y_i = \mathbb{E}_{s'\sim\mathcal{E}}\big[r + \gamma \max_{a'} Q(\boldsymbol{s},a';\boldsymbol{\theta}_{i-1})|\boldsymbol{s},a\big]$, $\rho(\boldsymbol{s},a)$ is the probability distribution over sequences $\boldsymbol{s}$ and actions a called the behavior distribution and $\mathcal{E}$ is the emulated environment. The update equation changes slightly from the regular parameterized Q-learning (Equation (12)) to

$$\boldsymbol{\theta}_{i+1} = \boldsymbol{\theta}_i + \alpha(Y_i^{DQN} - Q(\boldsymbol{S}_i, A_i;\boldsymbol{\theta}_i))\nabla_{\boldsymbol{\theta}_i} Q(\boldsymbol{S}_i, A_i;\boldsymbol{\theta}_i), \tag{15}$$

where α is the step size and the target is $Y_i^{DQN} = R_{i+1} + \gamma \max_a Q(\boldsymbol{S}_{i+1}, a;\boldsymbol{\theta}_i^-)$. The authors also used a technique called experience replay, storing the agent's past experiences (states, actions, and rewards) in a dataset $\mathcal{D}$. During training experiences $e_t = (\boldsymbol{s}_t, a_t, r_t, \boldsymbol{s}_{t+1})$ are sampled from the dataset, $e \sim \mathcal{D}$. This technique helps in many ways, it increases data efficiency, lowers the chances of strong correlations between samples, and decreases the chances of unwanted feedback loops leading to convergence to poor local minimum or divergence. [55]

2.1.4.2 *Double Deep Q-Network*

In a follow-up paper to [55], Van Hasselt et al. show that DQN significantly overestimates action-values and proposes an adaptation to the algorithm that addresses this [72].

By rewriting the target, Y_i^{DQN}, of the update equation for the DQN algorithm, Equation (15), the overestimation is easily spotted,

$$Y_i^{DQN} = R_{i+1} + \gamma Q(\boldsymbol{S}_{i+1}, \arg\max_a Q(\boldsymbol{S}_{i+1}, a;\boldsymbol{\theta}_i^-);\boldsymbol{\theta}_i^-). \tag{16}$$

In Equation (16), one sees that the same weights are used to both select and evaluate an action. By using two sets of weights, $\boldsymbol{\theta}$, and $\boldsymbol{\theta}'$, for the networks, one to determine the greedy policy and the other to determine its value, one arrives at the Double Q-learning algorithm. In this algorithm, the two value functions are learned by assigning experience randomly to update one of the two value functions [34]. The target now takes the following form,

$$Y_i^{DoubleQ} = R_{i+1} + \gamma Q(\boldsymbol{S}_{i+1}, \arg\max_a Q(\boldsymbol{S}_{i+1}, a;\boldsymbol{\theta}_i);\boldsymbol{\theta}_i'). \tag{17}$$

By combining Double Q-learning and DQN, Van Hasselt et al. arrive at the algorithm Double DQN (DDQN). They use the online network to evaluate the greedy policy and the target network to estimate its value. Compared to regular Double Q-learning, the second network's weights $\boldsymbol{\theta}_i'$ are replaced with the weights of the target network $\boldsymbol{\theta}_i^-$ for the evaluation of the current greedy policy. [72] The target network is updated in the same way using experience replay as in DQN but replacing

$$Y_i^{DQN} = R_{i+1} + \gamma \max_a Q(\boldsymbol{S}_{i+1}, a;\boldsymbol{\theta}_i^-) \tag{18}$$

with

$$Y_i^{DoubleDQN} = R_{i+1} + \gamma Q(\boldsymbol{S}_{i+1}, \arg\max_a Q(\boldsymbol{S}_{i+1}, a;\boldsymbol{\theta}_i);\boldsymbol{\theta}_i^-). \tag{19}$$

2.1.5 *Other Reinforcement Learning Topics*

The following section will discuss some of the common characteristics and problems of reinforcement learning methods.

2.1.5.1 *Exploring Starts*

One way of maintaining exploration during the training of a reinforcement learning agent is to use the assumption of *exploring starts*. Under this assumption, during the start of each episode, each state-action pair is given a non-zero probability, guaranteeing that every state-action pair (reachable at the first time step) will be visited an infinite amount of times in the limit of an infinite number of episodes [68].

While this method is useful for exploring in many cases, other more effective methods are often used. One such example is the ϵ-greedy policy discussed in Section 2.1.2.1.

2.1.5.2 *Sparse Rewards*

One of the essential parts of setting up a reinforcement learning problem is to design the reward signal. While one may not have to know what the correct actions should be during the design, it is important for the success of the reinforcement learning that the reward signal frames the problem and that it assesses progress in solving it.

Granting a reward in proportion to how well the agent succeeds in solving the problem is a common and often effective way. However, for some problems, especially when the tasks are long and complex, this design falls apart – the problem of sparse rewards arises. Thus, one should instead deliver frequent non-zero rewards over large and sparse rewards at, say, the end of the episode, which would make it more likely that the agent will achieve the goal at least once. One should also note that this design can lead the agent to discover unexpected ways to receive rewards, which can be both undesirable and dangerous. [68]

2.1.5.3 *Initial Q-Matrix*

Q-learning, among other RL methods, is dependent (to a varying degree) on the initial Q-matrix. These methods are, to quote Sutton and Barto, *"biased by their initial estimates."* [68] Sometimes, assigning a high action-value may facilitate and quicken the learning by encouraging exploration.

2.1.5.4 *Average of Q-Matrices*

In this thesis, ensemble averaging is employed, i.e., performing multiple RL runs and combining them into one by averaging them. The idea is that this procedure should decrease the variance of the trained models and increase the performance. There exist various variance-reducing approaches in the field of reinforcement learning, see, e.g., [4, 59, 76] which all are based on the idea of ensemble averaging via various schemes.

Our approach used in this thesis is very straightforward. After training k RL agents, we average the action values in a given state for these k agents to find the optimal action in this state. Strategies derived in this way will be called *mean strategies* for the remainder of this thesis.

2.2 THE LIMIT ORDER BOOK

Today's modern financial markets are order-driven, where the trading typically occurs via a continuous double-auction mechanism called a *limit order book* (LOB) [29, 48]. A double-auction is a type of auction in which buyers and sellers can simultaneously submit bids (offers to buy) and asks (offers to sell) for standardized lots of commodities or securities [22]. The origins of double-auctions are not clear, but such mechanisms have governed the trading on the New York Stock Exchange and the major Chicago exchanges for at least 100 years [22].

The bids and asks, as mentioned above, are in financial markets referred to as buy and sell *limit orders*. A limit order is an offer to buy/sell a specified quantity at a maximum/minimum price (the "limit"). Whenever a market participant is willing to trade at the best available price, the participant can submit a *market order* that is immediately executed against outstanding limit orders. A market participant can also cancel a limit order that it has posted but has not yet been matched against an incoming market order. All orders are aggregated and displayed in the limit order book for all participants to observe in order-driven markets. An LOB can thus be described as a collection of all active limit orders. [29] Graphically, this collection of active orders can be illustrated as in Figure 2.7.

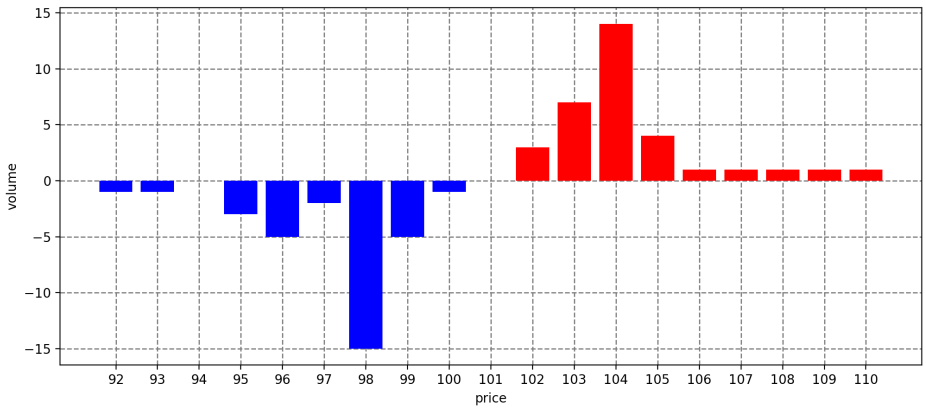


Figure 2.3: A graphical illustration of an LOB. The negative volumes in blue are buy orders, and the positive volumes in red are sell orders. More on this will be explained in later sections. Note that this figure's granularity is low, and no individual orders are observable.

There are many order-driven markets, all with custom rules and features, but in its simplest form, the three types of actions mentioned above (i.e., limit orders, market orders, and cancellations) constitute the foundation. Other more exotic order types exist in today's order-driven markets, including *fill-or-kill*, *hidden*, *iceberg*, and *good-till-time* orders [11]. However, this thesis will not further discuss these types of orders.

A noteworthy property of order-driven markets is that although the trading is centralized to a specific trading venue, the trading is decentralized in that any market participant can submit bids and asks and trade with any other market participant who is willing to accept the offered price. This is in stark contrast to quote-driven markets, where the trading occurs via one or a few market makers who are the sole providers of prices. If a market participant wants to trade in a quote-driven market, this must be done through a centralized market maker; this is typically how

bonds, foreign exchange (FX), and spot commodities are traded. Equities, futures, and other standardized derivatives are typically traded in order-driven trading venues.

2.2.1 *Mathematical Description*

Mathematically speaking, one may consider an order x to be a tuple $(p_x, \omega_x, t_x, \tau_x)$, where p_x is the price, $\omega_x \in \mathbb{Z}$ is the volume posted (if $\omega_x > 0$ then x is a sell order and if $\omega_x < 0$ then x is a buy order), t_x is the submission time of the order, and τ_x is the order type. In the stylized limit order book in this thesis, the allowed order types are *buy limit order* (LB), *sell limit order* (LS), *cancel buy order* (CB), *cancel sell order* (CS), *buy market order* (MB), and *sell market order* (MS). As such, $\tau_x \in \{\text{LB}, \text{LS}, \text{CB}, \text{CS}, \text{MB}, \text{MS}\}$. Note that for all orders resting in the limit order book, (p_x, ω_x, t_x) *typically* provides sufficient information with regards to the priority (see Primer 1) as all orders are active limit orders and the sign of ω_x indicates whether it is a sell or a buy order.

Returning to the initial description of an LOB as a collection of non-executed limit orders, we may denote the LOB at time t as $\mathcal{L}(t)$, which is the set of all *active* limit orders. A limit order is active if it has not been executed or canceled yet. We denote the set of active buy LOs at time t as $\mathcal{B}(t)$ and the set of active sell LOs at time t as $\mathcal{A}(t)$.

An important component of an LOB is its *tick size*. The tick size δ constitutes the smallest permissible interval between two price levels, and the LOB is thus defined on a discrete grid of prices (i.e., the *price levels*) [11]. In FX markets, the tick equivalent is called *pip* and constitutes the smallest quoted unit of the spot price. Whenever applicable in this thesis, δ will also denote the pip. As two examples, the tick size for `AAPL.O`[2] is \$0.01, and for the currency pair `EUR/USD`, the pip is 0.0001.

Next, we describe and define some critical quantities of an LOB. We refer to Table 2.1 for mathematical definitions of these quantities. The *bid price*, $b(t)$, is the highest price among all buy orders in the LOB. The *ask price*, $a(t)$, is the lowest price among all sell orders in the LOB. We assume that it always holds that the ask price is greater than the bid price. The difference between the ask and bid prices is called the *bid-ask spread*, denoted by $s(t)$. The *mid price*, $m(t)$, is the (non-weighted) average of the bid and ask prices. The *bid volume*, $q^b(t)$, and *ask volume*, $q^a(t)$, are the volumes posted on the bid and ask prices. Similarly, we have the posted volume on price level p, $q^p(t)$. The *micro price*, $\mu(t)$, is a weighted average of the bid and ask prices and is adjusted for order imbalances in the bid and ask. The observant reader may notice that (perhaps somewhat unintuitive at first) the weight of the bid price is the ask volume divided by the total bid and ask volume, and vice versa for the ask price (see Table 2.1). The idea is that this weighting incorporates the order imbalance into the mid price and better captures the "true" price [11]. The *order imbalance* for the top n levels, $\rho_n(t)$, measures the imbalance in posted volumes on the bid and ask sides for the n first price levels. It is a number in $[-1, 1]$, where ρ close to -1 indicates an ask-heavy imbalance and ρ close to 1 indicates a bid-heavy imbalance.

2.2.2 *Illustration of Dynamics*

This section aims to illustrate the dynamics of the limit order book graphically. First and foremost, a snapshot of an LOB (with a limited number of price levels displayed) is presented in

[2] Apple Inc.'s share listed on Nasdaq in New York. Its RIC (Reuters Instrument Code) is `AAPL.O`.

Quantity at time t	**Mathematical definition**
Bid price	$b(t) = \max_{x \in \mathcal{B}(t)} p_x$
Ask price	$a(t) = \min_{x \in \mathcal{A}(t)} p_x$
Bid-ask spread	$s(t) = a(t) - b(t)$
Mid price	$m(t) = \frac{b(t)+a(t)}{2}$
Bid volume	$q^b(t) = \sum_{x \in \mathcal{B}(t)} \omega_x \mathbb{1}_{p_x = b(t)}$
Ask volume	$q^a(t) = \sum_{x \in \mathcal{A}(t)} \omega_x \mathbb{1}_{p_x = a(t)}$
Volume at price level p	$q^p(t) = \sum_{x \in \mathcal{L}(t)} \omega_x \mathbb{1}_{p_x = p}$
Micro price	$\mu(t) = \frac{\lvert q^b(t)\rvert}{\lvert q^b(t)\rvert + q^a(t)} a(t) + \frac{q^a(t)}{\lvert q^b(t)\rvert + q^a(t)} b(t)$
Order imbalance for n levels	$\rho_n(t) = \frac{\lvert \sum_{i=0}^{n-1} q^{b(t)-i\delta}(t)\rvert - \sum_{i=0}^{n-1} q^{a(t)+i\delta}(t)}{\lvert \sum_{i=0}^{n-1} q^{b(t)-i\delta}(t)\rvert + \sum_{i=0}^{n-1} q^{a(t)+i\delta}(t)}$

Table 2.1: Mathematical definitions of important quantities related to the limit order book.

Order Priority and Matching Algorithms
Different exchanges have distinct order matching algorithms and rules that rank and match buy and sell orders. All active orders are ranked according to an order precedence hierarchy. The primary ranking criterion is price, e.g., an incoming buy market order will execute first against the sell limit order on the best ask before potentially moving to the next price level(s). If there is more than one limit order on a price level, secondary criteria are needed to rank the orders. A widespread secondary criterion is time, i.e., the first order to arrive at a price level usually has priority over later-arriving orders. Some trading systems prioritize displayed volume over hidden volume (in the case of iceberg orders) before the order submission time is considered. The New York Stock Exchange (NYSE) employs a price-parity hierarchy where (designated) market makers' orders are prioritized as a secondary rule, after which time follows as a third rule. [8]

Primer 1: Order priority and matching algorithms.

the upper-left of Figure 2.4. Multiple characteristics of an LOB described in previous sections can be observed, including the various price levels (separated by the tick size), the aggregated quantity posted on each level, the bid and ask price, and the bid-ask spread. Next, we display what happens to the LOB for arriving market orders, limit orders, and cancellations.

In the upper-right of Figure 2.4, we see an arriving buy market order with a volume of three in orange. Notice how it first is matched against the entire volume posted on $10,001$ (i.e., a volume of two), and the remaining volume of one is bought at $10,002$. In the lower-left in Figure 2.4, a buy limit order of size two at price level $9,999$ arrives, depicted in light blue. Lastly, the graph in the lower-right of Figure 2.4 shows in light pink the cancellation of a sell limit order at price level $10,003$ resulting in the removal of one share from the outstanding volume.

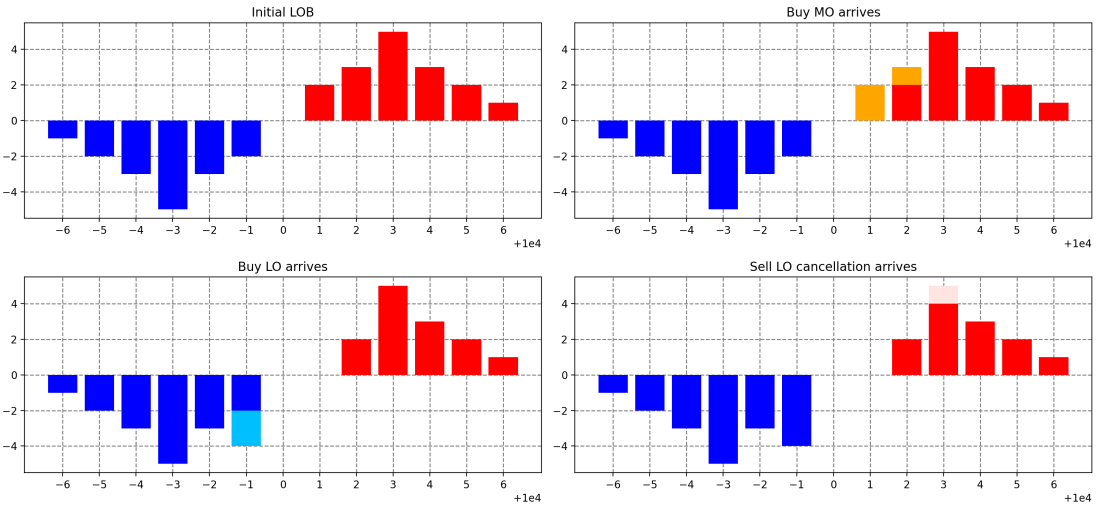


Figure 2.4: **upper left:** a snapshot on an LOB, **upper right:** the LOB as a buy market order with volume three arrives, **lower left:** the LOB as a buy limit order with volume two arrives on price level $9,999$, and **lower right:** the LOB as a cancellation of a sell limit order on price level $10,003$ arrives.

2.2.3 *Modeling the Limit Order Book*

It is valuable to be able to model a limit order book realistically for various reasons. In this thesis, we employ reinforcement learning to teach an agent how to make markets optimally. Saving the details of what this entails for later sections, employing reinforcement learning requires a huge number of interactions in the environment (i.e., limit order book) before a (nearly) optimal strategy can be found. One approach to finding this strategy is to simply let the agent learn from interacting and experimenting on its own in real financial markets. Once the agent has learned an optimal strategy, the potential reward from this can be large, albeit the losses that can accrue during learning are immense. Therefore, before letting an AI agent "loose" in the markets, it is advantageous if it can train in an as realistic environment as possible. Even if the aim is not to put the trained AI into production to trade in the markets but rather merely to study the optimal behavior, it is still desirable to have realistic depictions of the reality. The primary model used in this thesis is a Markov chain LOB model, which is presented in Section 2.2.3.2, before which we briefly cover a simpler model.

It should be noted that these models only model the regular *intraday* evolution of the limit order book. However, there are events such as the opening and closing auctions and, e.g., circuit breakers that are not modeled with these models. Specific auction rules are used to match orders during these occasions rather than the regular LOB rules.

Many assets (equities, futures, commodities, and FX) are typically traded across multiple trading venues. One usually refers to this as *market fragmentation*. It should be noted that the modeling in Section 2.2.3.2 may also be generalized and considered to work in this setting. For example, the Markov chain modeling of the LOB may be viewed as the modeling of an aggregate of bids and asks in fragmented markets (instead of an actual limit order book). In addition to regular trading venues, *dark pools* (also known as Alternative Trading Systems) have since 2007 gained popularity as an alternative type of trading venue. Dark pools are trading venues where the limit order book is hidden from all market participants. [1] According to Nasdaq, dark pools

capture approximately 40% of the US equity trading volume [44]. The modeling in this thesis does not consider dark pools, but there exist some works that support simultaneous trading in lit ("regular") and dark financial markets, see, e.g., [11, 13, 19]. It should be noted that these works all consider the problem of optimal execution of a larger block of shares or similar.

2.2.3.1 *Simple Probabilistic Model*

The model presented in this section is the one presented by Cartea et al. in Chapter 10.2 in [11], which does not model the LOB *per se* but rather the mid price dynamics, the arrival of market orders, and the conditional probability of an arriving market order to fill a limit order. The model is time-continuous and consists of the following variables:

- The mid price $S = \{S_t\}_{0 \leq t \leq T}$, where $S_t = S_0 + \sigma W_t$, $\sigma > 0$, $S_0 = \mu$, and W_t is a standard Brownian motion. Here, σ and μ are non-negative and real-valued constants.
- The arrival of buy $(+)$ and sell $(-)$ market orders is modeled by $M^{\pm} = \{M_t^{\pm}\}_{0 \leq t \leq T}$, which are Poisson random processes with intensities $\lambda^{\pm}$. Here, $\lambda^{\pm}$ are non-negative and real-valued constants.
- A market participant (our agent, i.e., a market maker) posts buy and sell limit orders on the depths $\delta^{\pm} = \{\delta_t^{\pm}\}_{0 \leq t \leq T}$, i.e., buy limit orders are posted on $S_t - \delta_t^-$ and sell limit orders are posted on $S_t + \delta_t^+$.
- Conditioned on the arrival of a market order, a market participant's posted limit order is filled with probability $\exp\{-\kappa^{\pm}\delta_t^{\pm}\}$, where $\kappa^{\pm} > 0$ are non-negative and real-valued constants and are measures of the competitiveness in the market.

One should note that this model does not include any explicit modeling of the LOB; it only assumes that there is one market maker (the agent) who can only put up a single sell and a single buy LO, with which market participants can trade. However, this model allows for an analytical solution to be found, thus allowing for a comparison between analytical and RL-generated solutions, which is why it was chosen. Henceforth, this model will be referred to as the simple probabilistic model (SPM).

2.2.3.2 *Markov Chain Model*

The Markov chain model presented in this section is predominately based on the one presented by Hult and Kiessling in [39], which was influenced by [16]. It is a time-continuous Markov chain where the state is the posted volume on each price level. In this model, there are six event types: *buy limit order* (LB), *sell limit order* (LS), *cancel buy order* (CB), *cancel sell order* (CS), *buy market order* (MB), and *sell market order* (MS). The arrival of an event results in a state transition in the Markov chain.

It is possible to represent a simplified and consolidated LOB as two arrays rather than as a set of tuples. We follow Hultin et al. (who use the model by Hult and Kiessling in [41]) and construct the two vectors $(a(t), q^{b(t)+\delta}(t), \ldots, q^{b(t)+d\delta}(t))$ and $(b(t) - a(t), q^{a(t)-\delta}(t), \ldots, q^{a(t)-d\delta}(t))$, $d \in \mathbb{N}$. The first vector represents ask side – the ask price and the available volume at the different price levels. $q^{b(t)+i\delta}(t)$ is thus the volume available at ith price level above the current bid price. Note that only d price levels of both sides are considered. Similarly, the second vector represents the bid volumes. The motivation behind this representation is that it facilitates the modeling of the

LOB. For each order x, the information (ω_x, t_x), and thus the information about individual limit orders, is lost in this representation.

Above, d denotes the number of price levels we consider on the bidding and asking sides. Throughout this thesis, we let $d = 10$. In theory, there are infinitely many price levels, so choosing a finite d truncates the LOB. A remark is that ten levels are reasonably often the depth included with so-called *Level 2* market data,[3] which in our opinion justifies $d = 10$. Additionally, including additional price levels grows the state space exponentially, which impedes the learning for RL algorithms (when one includes the entire LOB in the state space).

Before presenting the transition rates in the Markov chain, we introduce some additional notations. First, it should be noted that we have done some minor notational alterations compared to the notation used by Hult and Kiessling.

First, let $n \in \mathbb{N}$, $n \geq 2$, be the number of price levels in the limit order book. We let the Markov chain consist of the current volumes posted on each price level at time t, $\boldsymbol{s}_t = (s_{t,1}, \ldots, s_{t,n})^\top$, where $s_{t,i}$ is the posted volume on price level i at time t. We denote the state of feasible states as $\mathcal{S}$. For a given state $\boldsymbol{s} \in \mathcal{S}$, the functions $\text{bid}(\boldsymbol{s})$ and $\text{ask}(\boldsymbol{s})$ are the bid and ask prices, respectively. It is assumed that $s_{t,n} > 0$ and $s_{t,1} < 0$ and that $\text{bid}(\boldsymbol{s}) < \text{ask}(\boldsymbol{s})$, $\forall \boldsymbol{s} \in \mathcal{S}$ and $\forall t$.

Next, we show the state transitions and their rates. As previously, let $x = (p_x, \omega_x, t_x, \tau_x)$ be an order. We denote $\boldsymbol{e}_{p_x/\delta}$ as the n-dimensional unit vector with 1 in the (p_x/δ)th coordinate. (Note that the set of price levels is $\{\delta, 2\delta, \ldots, n\delta\}$, why for any admissible price p it will hold that $p/\delta \in \{1, \ldots, n\}$.) Then,

$$
\begin{aligned}
\text{LS:}\quad & \boldsymbol{s} \to \boldsymbol{s} + \omega_x \boldsymbol{e}_{p_x/\delta},\ p_x > \text{bid}(\boldsymbol{s}),\ \omega_x \geq 1, \\
& \text{with rate } p_L(\omega_x)\lambda_L^S(p_x - \text{bid}(\boldsymbol{s})), \\
\text{LB:}\quad & \boldsymbol{s} \to \boldsymbol{s} + \omega_x \boldsymbol{e}_{p_x/\delta},\ p_x < \text{ask}(\boldsymbol{s}),\ \omega_x \leq -1, \\
& \text{with rate } p_L(|\omega_x|)\lambda_L^B(\text{ask}(\boldsymbol{s}) - p_x), \\
\text{CS:}\quad & \boldsymbol{s} \to \boldsymbol{s} - \boldsymbol{e}_{p_x/\delta},\ p_x \geq \text{ask}(\boldsymbol{s}), \\
& \text{with rate } \lambda_C^S(p_x - \text{bid}(\boldsymbol{s}))|q^{p_x}|, \\
\text{CB:}\quad & \boldsymbol{s} \to \boldsymbol{s} + \boldsymbol{e}_{p_x/\delta},\ p_x \leq \text{bid}(\boldsymbol{s}), \\
& \text{with rate } \lambda_C^B(\text{ask}(\boldsymbol{s}) - p_x)|q_x^p|, \\
\text{MS:}\quad & \boldsymbol{s} \to \boldsymbol{s} + \omega_x \boldsymbol{e}_{p_x/\delta},\ p_x = \text{bid}(\boldsymbol{s}),\ 1 \leq \omega_x \leq |q^{p_x}|, \\
& \text{with rate } p_M(\omega_x)\lambda_M^S, \\
\text{MB:}\quad & \boldsymbol{s} \to \boldsymbol{s} + \omega_x \boldsymbol{e}_{p_x/\delta},\ p_x = \text{ask}(\boldsymbol{s}),\ -q^{p_x} \leq \omega \leq -1, \\
& \text{with rate } p_M(|\omega_x|)\lambda_M^B.
\end{aligned}
\tag{20}
$$

Above, ω denotes the order size, and as previously, we use the convention that $\omega < 0$ is a buy order and $\omega > 0$ is a sell order. $p_L(\cdot)$ and $p_M(\cdot)$ denote the probability mass functions of the absolute order sizes of LOs and MOs. We follow Hult and Kiessling and let the probability mass functions of the absolute size be

$$
p_L(|\omega|) \propto (\exp(\gamma_L) - 1)\exp(-\gamma_L|\omega|), \text{ and} \tag{21}
$$

$$
p_M(|\omega|) \propto (\exp(\gamma_M) - 1)\exp(-\gamma_M|\omega|). \tag{22}
$$

[3] There exists no standard number of price levels included in Level 2 market data, but it is usually in the range of 5-10.

Note that the probability mass functions are merely proportional to the expressions on the right-hand side above, as the expressions are without a normalizing constant.

The observant reader will also notice that due to the transition definitions in Equation (20), the size ω of market orders is limited by the best bid and ask volumes which implies that a single market order cannot *walk the book* (in one "event" at least).[4] Also, cancellations are always of size 1.

In [39], the aforementioned model parameters are fitted on high-frequency `EUR/USD` data. For a detailed description of the model calibration procedure, we refer to the article by Hult and Kiessling [39]. The parameter values are presented in Table 2.2. As can be seen in the table, these parameter values are not perfectly symmetrical with regard to buy and sell actions. To avoid this influencing the optimal market making strategy, we let $\lambda_L^B(j) = \lambda_L^S(j)$ and $\lambda_C^B(j) = \lambda_C^S(j)$, $\forall j$, in our simulations. Specifically, they are the average of the buy and sell parameters in the table.

j	1	2	3	4	5		
$\lambda_L^B(j)$	0.1330	0.1811	0.2085	0.1477	0.0541	λ_M^S	0.0467
$\lambda_L^S(j)$	0.1442	0.1734	0.2404	0.1391	0.0584	λ_M^B	0.0467
$\lambda_C^B(j)$	0.1287	0.1057	0.0541	0.0493	0.0408	γ_M	0.4955
$\lambda_C^S(j)$	0.1308	0.1154	0.0531	0.0492	0.0437	γ_L	0.5667

Table 2.2: Model parameters in the Markov chain LOB model, fitted on high-frequency `EUR/USD` data in [39].

[4] Cartea et al. define walking the book as *"the process whereby a large entering market order is executed against standing LOs at increasingly worse prices"* [11].

2.3 OPTIMAL MARKET MAKING

As with so many other fields in applied mathematics, optimal market making strategies can be found through either analytical or numerical methods. In this section, we cover some results from these two paradigms in market making, and in the end, we discuss some benchmark models. Before proceeding to mathematical results, we invite the reader to read a brief description of market making at SEB in Primer 2.

Market Making in FX at Skandinaviska Enskilda Banken (SEB)
SEB is one of the world's largest dealer in Scandinavian currencies by market share. Previously, currency pairs were only traded in a primary marketplace, but market fragmentation in FX has risen over the past years with a surge in trading OTC through different dealers. Primary markets still exist and play an important role, but their role has gradually decreased. Many clients use reference prices at primary marketplaces for various processes; e.g., mark-to-market valuations. Additionally, stop losses are typically triggered by such market reference prices.

SEB's leading role in Scandinavian currencies has some consequences for the business. First and foremost, they can offer significantly better prices for their clients than through primary markets and other available venues. However, their massive trading volumes also mean that they can have a significant market impact that impedes continuous hedging activities via primary markets or other dealers. Since SEB generally provides considerably better prices than the primary markets or other dealers, it would be prohibitively expensive to hedge all deals externally, as this would almost always result in a guaranteed loss for SEB. Nevertheless, the large scale of SEB's operations means that they often can net positions internally from different transactions, notably through automatic quotation mechanisms that may skew prices, increasing the positive or negative net flow from clients. *Skewing prices* refers to when the price mark-up is higher on one of the bid or ask sides.

SEB's risk management and price quotation have not always been automatic. Over the past decade, a large shift has taken place from manual handling toward mostly automatic processes. The quotation mechanics are mathematically sophisticated; merely providing the clients with prices that are a fixed number of pips better than the reference prices is not satisfactory. The price discovery process is relatively heuristic, but adjustments to these prices are made based on sound and advanced statistical models.

Primer 2: A description of market making in FX at SEB.

2.3.1 *Analytical Results*

Analytical approaches to optimal market making are plentiful. Typically, the mathematical toolbox used to solve optimal market making problems consists of classical methods for optimal control, such as dynamic programming, as the problem frameworks generally are modeled with stochastic differential equations. Usually, these frameworks are based on rather strict and naive assumptions.

The two prevailing classes of approaches are inventory-based and information-based methods. Broadly speaking, inventory-based models are focused on actively managing the inventory risk, whereas information-based models are focused on actively managing the risk of adverse selection.

The two model frameworks are not mutually exclusive and may be used in combination. One such example is presented in Chapter 10.4 in [11] by Cartea et al.

2.3.1.1 *Inventory-Based Market Making*

Ho and Stoll provided an early attempt at deriving optimal market making strategies. Their paper from 1981 is considered the seminal paper in the inventory-based market-making framework by some. The authors examined the optimal behavior for a monopolistic dealer in a single stock (i.e., essentially a quote-driven market) where the demand to trade and the return on the dealer's inventory are stochastic. The authors used stochastic dynamic programming to derive utility-maximizing bid and ask prices (utility is further discussed in Primer 3). [36]

Avellaneda and Stoikov provided another work that advanced the research of Ho and Stoll. They combined the utility-maximizing setting of Ho and Stoll with the microstructure theory of limit order books. They used asymptotic expansion to derive explicit formulas for the optimal bid and ask quotes for a risk-averse market maker. [6]

Stoikov and Sağlam framed the problem of optimal market making in a mean-variance framework and thus avoided the concern of utility functions (discussed in Primer 3) while maintaining a direct connection to risk management in terms of inventory risk. Their work is further interesting due to the very setting they consider: market making in options. They deal with three different cases, (1) where the market maker can continuously delta hedge all risk through trading in the underlying asset, (2) where the market maker may not continuously delta hedge but may post quotes in an illiquid underlying asset, and (3) where there the volatility is stochastic, and the resulting Vega and Gamma exposures affect the optimal quotes. [66]

Roughly speaking, all three aforementioned preeminent works follow the same approach as the problem is framed as an optimal control problem and solved using dynamic programming principles. Next, we illustrate this procedure by recounting a derivation of an optimal market making strategy presented by Cartea et al. in [11].

Example: Cartea et al.

We here present a derivation of optimal bid and ask depths for a simple market making model with the SPM of the LOB in Section 2.2.3. This section is a brief account of the one provided by Cartea et al. in [11] (Chapter 10.2) and is an inventory-based market making model. The strategy derived here makes no use of utility, but the authors provide an example of a utility-maximizing strategy in Chapter 10.3.

In addition to the variables specified in Section 2.2.3, we next introduce the following variables:

- $N^{\pm} = \left\{N_t^{\delta,\pm}\right\}_{0\leq t\leq T}$ is the controlled counting process of the agent's filled sell $(+)$ and buy $(-)$ LOs.
- $X = \left\{X_t\right\}_{0\leq t\leq T}$ is the market maker's cash process and satisfies

$$dX_t = (S_{t-} + \delta_t^+)dN_t^{\delta,+} - (S_{t-} - \delta_t^-)dN_t^{\delta,-}. \tag{25}$$

- $Q = \left\{Q_t\right\}_{0\leq t\leq T}$ is the market maker's inventory process, $Q_t = N_t^{\delta,-} - N_t^{\delta,+}$.[5]

[5] As the reader may have noticed by now, Q and q are used to denote several different quantities. We here try to clarify the different notation used. $Q(\cdot,\cdot)$, $Q(\cdot,\cdot;\cdot)$, $q(\cdot,\cdot)$, $q_*(\cdot,\cdot)$, and $q_t(\cdot,\cdot)$ all denote the action-value in some

Expected Utility
The notion of *expected utility* was first formulated in 1738 by Swiss mathematician Daniel Bernoulli and has in economics been used to explain various phenomena, such as the purchase of insurance [69]. Utility functions are used to measure subjective utility, which typically is not linear in the payoff (rather increasing and concave), e.g., given a loss and gain of the same size, the loss is more hurtful than the gain is beneficial ($u(-X) < -u(X)$). We thus see that utility functions can be used to capture risk aversion [40]. Furthermore, these utility functions can be used to incorporate the characteristics of prospect theory, which have been shown to hold for humans in general in terms of framing and loss aversion [70, 71].

While there are infinitely many parametric families of utility functions, the parametric family hyperbolic absolute risk aversion (HARA) is popular and flexible. Usually, this family is parametrized by the two parameters γ and τ. With the only assumption that $A(x) = -\frac{u''(x)}{u'(x)} = \frac{1}{\tau+\gamma x}$ for $\tau + \gamma x > 0$ one can derive the utility function as (see e.g., [40]):

$$u(x) = \begin{cases} \frac{1}{\gamma-1}(\tau+\gamma x)^{1-1/\gamma}, & \text{if } \gamma \neq -1, 0, 1 \\ \log(\tau + x), & \text{if } \gamma = 1 \\ -\tau e^{-x/\tau}, & \text{if } \gamma = 0 \\ -\frac{1}{2}(\tau - x)^2, & \text{if } \gamma = -1 \end{cases} \tag{23}$$

Another popular family is the constant absolute risk aversion (CARA), usually denoted by exponential utility. By assuming $-\frac{u''(x)}{u'(x)} = \gamma$, where γ is a constant, one gets

$$u(x) = \begin{cases} (1 - e^{-\gamma x}), & \text{if } \gamma \neq 0 \\ \gamma, & \text{if } \gamma = 0. \end{cases} \tag{24}$$

For this family, γ determines the risk preference: (1) $\gamma > 0$ risk aversion, (2) $\gamma = 0$ risk-neutrality and (3) $\gamma < 0$ risk-seeking. [14]

One major drawback of utility functions is that they may lack intuitiveness and can be chosen somewhat arbitrarily.

Primer 3: Expected Utility.

A critical remark is that, as defined above, $Q > 0$ implies a net long inventory, whereas $Q < 0$ implies a net short inventory. This is also the convention we use and may confuse the reader since a negative order volume corresponds to a buy order.

The inventory is capped by a maximum inventory $\overline{q}$, and by a minimum inventory $\underline{q}$, both finite.

Before deriving an "optimal policy," one must specify exactly what this entails. In this particular case, the optimal policy is the strategy that maximizes the expected cash at time T. A running inventory penalty is also included and defined by the parameter ϕ.

The performance criterion is

$$H^\delta(t, x, S, q) = \mathbb{E}_{t,x,S,q}\left[X_T + Q_T(S_T - \alpha Q_T) - \phi \int_t^T (Q_u)^2 du\right], \tag{26}$$

sense. Q_t, q, Q, $\underline{q}$, and $\overline{q}$ all relate to inventory in some sense. Finally, we will in Section 2.4.3 introduce the notion of quartile where $\widetilde{Q}_n$ is used to denote the nth quartile.

where the final term represents the running inventory term and $\phi \geq 0$. The middle term represents the cash received when liquidating at time T, where $\alpha \geq 0$ represents the fees of taking liquidity. The market maker's (optimal) value function is

$$H(t, x, S, q) = \sup_{\delta^{\pm} \in \mathcal{A}} H^{\delta}(t, x, S, q). \tag{27}$$

By applying principles of stochastic differential equations and dynamic programming and assuming $\kappa^+ = \kappa^- = \kappa$, one arrives at the following optimal strategy (assuming $\lambda^+ = \lambda^-$)

$$\delta^{+,*}(t, q) = \begin{cases} \frac{1}{\kappa} - h(t, q-1) + h(t, q), & \text{if } q \neq \underline{q}, \\ +\infty, & \text{if } q = \underline{q}, \end{cases} \tag{28}$$

$$\delta^{-,*}(t, q) = \begin{cases} \frac{1}{\kappa} - h(t, q-1) + h(t, q), & \text{if } q \neq \overline{q}, \\ +\infty, & \text{if } q = \overline{q}, \end{cases} \tag{29}$$

where

$$h(t, q) = \frac{1}{\kappa} \log \omega(t, q), \tag{30}$$

with

$$\boldsymbol{\omega}(t) = [\omega(t, \underline{q}), \omega(t, \underline{q}+1), \ldots, \omega(t, \overline{q})]^\top = e^{\boldsymbol{A}(T-t)} \boldsymbol{z},$$

where

$$\boldsymbol{A}_{i,q} = \begin{cases} -\phi \kappa q^2, & \text{if } i = q, \\ \lambda^+ e^{-1}, & \text{if } i = q-1, \\ \lambda^- e^{-1}, & \text{if } i = q+1, \\ 0, & \text{otherwise}, \end{cases} \tag{31}$$

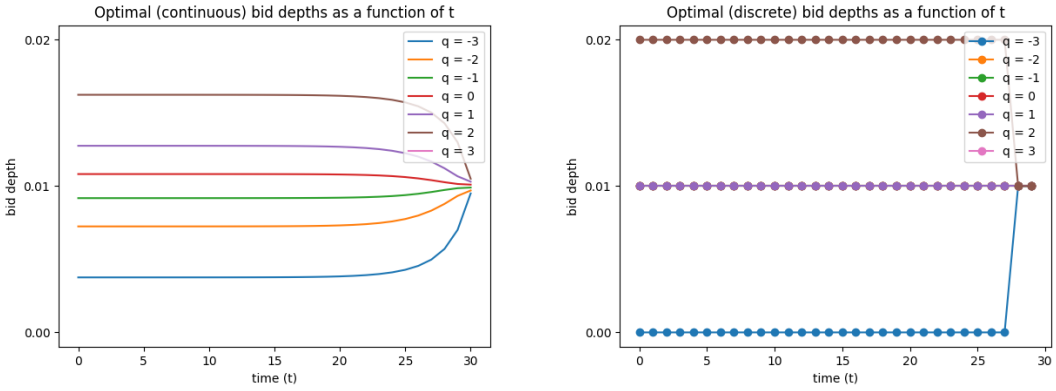
and where $\boldsymbol{z}$ is a $(\overline{q} - \underline{q} + 1)$-vector where the components are given by $z_j = e^{-\alpha \kappa j^2}$, $j = \underline{q}, \ldots, \overline{q}$. For the full derivation, see Appendix A.1.

With these solutions, we can now have a look at an example. Figure 2.5 shows the optimal bid depths obtained by this method: Figure 2.5a shows the continuous results while Figure 2.5b shows the depths rounded to the closest tick, which will be used to compare with the tabular Q-learning algorithm. Note that the discretized version will perform worse than the continuous solution and that there is no guarantee that it is the optimal discrete strategy.

In Figure 2.6 we can see some characteristics of the solution, the *fill rate*, and the *inventory drift*. In Figure 2.6a we see the fill rate, $\lambda^{\pm} e^{-\kappa^{\pm} \delta^{\pm,*}(q)}$, which is how fast the market maker's LOs are filled given the held volume. In Figure 2.6b we see the inventory drift, defined as

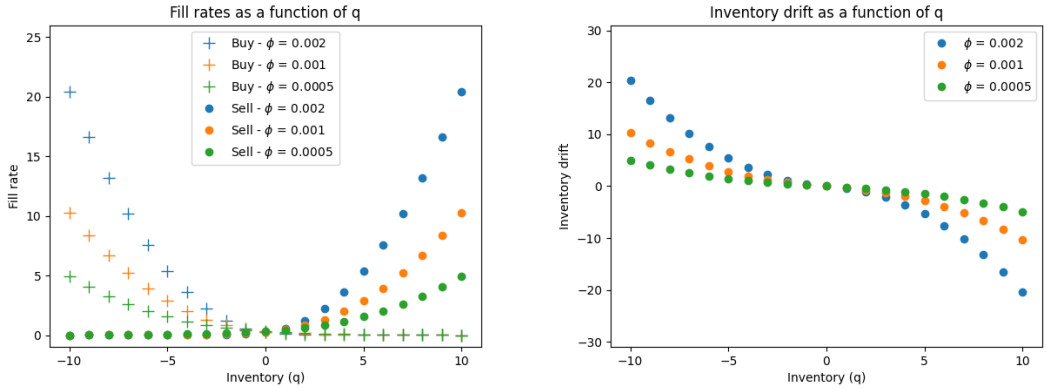
$$\lambda^- e^{-\kappa^- \delta^{-,*}(q)} - \lambda^+ e^{-\kappa^+ \delta^{+,*}(q)},$$

that is how the market maker's inventory is expected to drift given an inventory q.



(a) (b)

Figure 2.5: **(a)** the analytically optimal bid depths, and **(b)** the discretized version of this (see page 22). **Parameters:** $\phi = 10^{-5}$, $\lambda^{\pm} = 1$, $\kappa^{\pm} = 100$, $\overline{q} = -\underline{q} = 3$, $\alpha = 0.0001$, $\sigma = 0.01$, $\mu = 100$.



(a) (b)

Figure 2.6: **(a)** The fill rates given the optimal depths. +'s are buy MOs and **o**'s sell MOs. **(b)** The inventory drift given the optimal depths.
Parameters: $\phi \in \{2 \times 10^{-5}, 10^{-3}, 5 \times 10^{-4}\}$, $\lambda^{\pm} = 1$, $\kappa^{\pm} = 100$, $\overline{q} = -\underline{q} = 3$, $\alpha = 0.0001$, $\sigma = 0.01$, $\mu = 100$.

Discretization of the Simple Probabilistic Model

The primary purpose of using the SPM is to see if tabular Q-learning can be used to find (or even beat) the analytically optimal solution. To make it compatible with tabular Q-learning, the SPM has to be discretized in terms of time, price, and a performance criterion.

The time will be taking discrete values in terms of seconds, that is $t \in \{0, 1, \ldots, T\}$ where T is the end time of the episode. The price will take values that are multiples of the tick size δ, that is $m(t) \in \{m \mid m = k \cdot \delta, k \in \mathrm{N}\}$.

The reward used for the Q-learning needs to closely replicate the performance criterion used when deriving the analytically optimal solution (Equation (26)) to enable comparison. The performance criterion can be broken down into three parts:

1. X_t – the cash process at time $t = T$
2. $Q_T(S_T - \alpha Q_T)$ – the value received for liquidating the position at time $t = T$
3. $-\phi \int_t^T (Q_u)^2 du$ – the running inventory penalty between times t and T

While one might be tempted to grant the entire reward at the final time step, one should note that reward sparsity is damaging to the agent's learning (as discussed in Section 2.1.5.2).

Thus, the first two parts will be represented by two processes – the cash process, X_t, and the holding process, H_t. X_t represents the cash the MM is holding at time t, and H_t represents the value of the MM's inventory at time t. The holding value H_t is in SPM defined as the market maker's inventory multiplied by the mid price.

The value process, V_t, is then the cash process plus the holding value, which is defined as

$$V_t = X_t + H_t. \tag{32}$$

At the end of the final time step, when $t = T$, the market maker will also be forced to liquidate its position and receive the value $Q_T(S_T - \alpha Q_T)$, meaning that $V_T = X_{T^-} + Q_T(S_T - \alpha Q_T)$, where T^- indicates the instance just before the time final step ends.

Finally, the third part of the performance criterion is discretized by turning the integral into a sum and is defined by the function $g(q; \phi)$

$$g(q; \phi) = -\phi q^2 \Delta t, \tag{33}$$

but since we have that $\Delta t = 1$, this term is omitted below. Here, and in the rest of this thesis, Δ is the difference operator unless otherwise stated.

To combat sparse rewards, we thus suggest that the change of the value process plus the penalty function g should be used as a reward. That is,

$$R_t = \Delta V_t + g(Q_t; \phi) = V_t - V_{t-1} - \phi Q_t^2. \tag{34}$$

Since $V_0 = 0$, this would mean that

$$\sum_{\tau=1}^{T} R_\tau = V_T - V_0 - \phi \sum_{\tau=1}^{T} Q_\tau^2 = X_{T^-} + Q_T(S_T - \alpha Q_T) - \phi \sum_{\tau=1}^{T} Q_\tau^2, \tag{35}$$

which is the total profit of the episode minus the running inventory penalty – a good discrete approximation of Equation (26).

2.3.1.2 *Information-Based Market Making*

Information-based market making models are based on the assumption of *adverse selection*, i.e., that some of the market makers' counterparties are better-informed market participants than the market makers themselves. Therefore, market makers are at an informational disadvantage, and factors such as the order imbalance and market quality become important variables to infer information about the price movement direction and to devise an optimal market making strategy.

Multiple distinct approaches to information-based market making exist. For example, in the seminal paper on information-based market making from 1983, Glosten and Milgrom consider a risk-neutral market maker and show that the presence of informed traders widens the bid-ask spread that market makers quote (under optimal behavior). [28]

In [11], the authors derive optimal bid and ask depths where the mid price follows two different dynamics modeled with a stochastic differential equation. The first dynamic captures the movement of the "true" price, while the second captures the movement in the mid price resulting from arriving MOs. The apparatus used to derive optimal quotes is pretty sophisticated and is analogous to the one in the example of Cartea et al. in Section 2.3.1.1 and the solution differs only in one term.

Other articles in the domain have often incorporated price prediction to a varying degree, such that the market maker can adjust its quotes and not get caught off-guard. One such approach is described in [1], where the focus is to merely infer insights into how the market will move (direction and magnitude) based on order flow – order flow autocorrelation, order flow aggressiveness, the shape of the order book, and the evolution of tick data. The mathematical repertoire presented is rather rudimentary and includes multiple linear regression, autocorrelation, and likelihood computations. Unlike [11], no explicit formulas for optimal bid and ask depths are presented.

2.3.2 *Previous Work: Reinforcement Learning and Market Making*

This section covers previous work on reinforcement learning and market making. Our literature study on reinforcement learning and market making in this thesis is largely inspired by the excellent and recently published review article *Reinforcement Learning Approaches to Optimal Market Making* by Gašperov et al. [25]. Additionally, we have searched on Google Scholar for

```
'market making' + 'reinforcement learning'
```

in an attempt to find articles that are even more recent and/or not covered by Gašperov et al.

In the remainder of this section, we discuss interesting findings with regards to the problem formulation of market making (e.g., state space, action space, rewards, etc) and various reinforcement learning aspects.

2.3.2.1 *State Space Representation*

To appropriately design the state space is of utmost importance in reinforcement learning. On the one hand, the state space should contain sufficiently detailed data to be able to infer optimal behaviors in specific situations, but on the other hand, a sparse state space is desirable in order to facilitate quick learning and to ensure the efficiency of RL algorithms. A brief discussion regarding some state space variables found in the literature on reinforcement learning for market making is presented next.

Inventory

Inventory-based models are widely used. Indeed, Gašperov et al. establish in their study that a colossal 91% of articles use an inventory-based state space [25]. It is easy to understand why, as it is in the market maker's best interest to mitigate market risks of holding large inventories. Neglecting the current inventory levels can lead to large overexposures to directional moves in the asset's price and as such large losses.

Different approaches to incorporating the inventory as a state space variable exist. Some authors use the nominal quantity without any further adjustments. Chan and Shelton in [12] let the inventory at time t be $q_t \in \{\underline{q}, \dots, \overline{q}\}$, where $\overline{q}$ ($\underline{q}$) corresponds to the maximum long (short)

position allowed. The authors do not specify what upper bound they used, but from experimental data, it is clear that the inventory at times exceeded 600. Using such a large discrete grid of inventory levels (i.e., $\approx 2 \times 600$) significantly affects the performance of tabular RL algorithms.

Other authors, aiming for lower computational complexity, reduce the inventory states to a number of bins. For example, Lim et al. provide one such instance in [48], where there are seven states for the inventory given by a *small*, *medium*, and *large* inventory imbalance, in either a positive or negative direction, as well as a zero inventory imbalance state.

Time

Time-based models are somewhat less standard than inventory-based models but do appear in, for instance [48, 58, 62, 65, 79]. Interestingly, we have not been able to find a single study that uses time as a state space variable but not the market maker's inventory. Typically, the trading period $[0, T]$ is discretized into $\{0, \Delta t, 2\Delta t, \dots, T\}$, and the time state variable is either the time passed or the time remaining. A remark is that most of the studies that we have identified using the time state variable are based on the analytical Avellaneda-Stoikov model discussed in Section 2.3.1.1 (this was also pointed out by Gašperov et al. [25]).

Imbalances

A popular choice for state space variables includes various measures of imbalance. As defined in Table 2.1, the order imbalance is for instance used in [12, 49, 53, 60, 64]. The terminology varies in the literature, and the quantity is also referred to as *queue*, *book*, and *volume* imbalance. Interestingly, in one of their experiments, Chan and Shelton use the order book imbalance as the *only* state space variable [12].

Trade flow imbalance is another imbalance measure used in [60]. It measures the magnitude of buyer and seller initiated transactions and is defined as

$$TFI_t = \frac{gain_t - loss_t}{gain_t + loss_t}, \tag{36}$$

where $gain_t = \sum_\ell BI^w_{\ell=0}$ and $loss_t = \sum^w_{\ell=0} SI_\ell$. Here, BI and SI are the buyer and seller initiated transactions, and w is a time window (5, 15, and 30 minutes in [60]).

A motivation for including imbalance variables is that studies have shown that trade and order imbalances provide a predictive ability relating to the future price of an asset [15, 77, 78]. As such, imbalance state variables can be argued to appertain to the information-based market making framework discussed in Section 2.3.1.2.

Relative Strength Index

The *relative strength index* (RSI) for a period n is defined as

$$RSI(n) = 100 - \frac{100}{1 + \frac{EMA(U,n)}{EMA(D,n)}}, \tag{37}$$

where $EMA(\cdot, n)$ is n period exponential moving average function. $U = \{U_i\}_{i\geq 0}$ and $D = \{D_i\}_{i\geq 0}$ are the upward and downward price change processes, respectively. They are computed as

$$U_i = [P(i) - P(i-1)]_+, \text{ and} \tag{38}$$
$$D_i = [P(i-1) - P(i)]_+. \tag{39}$$

Here $P(i)$ refers to the asset's price at period i. The RSI, or a version thereof, is used in [49, 51, 60]. In [51] it is not clear which n is used, but in [49], $n \in \{30, 90, 270\}$ (the time unit is not provided) is used and in [60], $n =\in \{5, 15, 30\}$ minutes is used. The RSI is commonly used in technical analysis as an indicator to evaluate if an asset has been overbought or oversold.

Volatility

Volatility is included as a variable in the state spaces in [32, 58, 64]. Using their notation, Patel [58] defines the n period EMA at time t as

$$\text{EMA}_t^n = \frac{2P(t)}{n+1} + \frac{\sum_{j=t-n}^{t} P(j)}{n}\left(100 - \frac{2}{n+1}\right), \tag{40}$$

where $P(i)$ it the price at period i. He then defines the volatility of m periods:

$$\text{Volatility}_t^m = \frac{\text{EMA}_{t,n} - \text{EMA}_{t-m,n}}{\text{EMA}_{t-m,n}}. \tag{41}$$

A motivation for including the volatility is that the probability of adverse market movements and losses increases as the volatility is higher. For that reason, the market maker may choose to adjust the quoting strategy based on the prevailing volatility. Indeed, in a theoretical framework akin to the Avellaneda-Stoikov model, Guéant shows (with closed-form approximations) that depending on the current inventory of the market maker, a change in the volatility will change the optimal bid and ask prices by either widening/tightening the spread or skewing the prices [30].

Price Data

Various price data, including the bid price, ask price, mid price, bid-ask spread, the mid-price movement, the difference between the market maker's ask and the best ask (same for the bid), and the bid and ask sizes, are used in e.g., [32, 45, 62, 64, 80].

LOB Data

Few previous works incorporate extensive LOB data into the state space. Sadighian does so in [60, 61] and Liu does so in [49]. Notably, both authors use deep reinforcement learning as to why a small state space is less critical. Both Sadighian and Liu render the LOB as stationary prices by using the *price level distance to mid price*[6] for d price levels on both sides of the mid price in conjunction with either the cumulative notional value or the cumulative volume on each price

[6] Sadighian calls it "price level distance to midpoint" whereas Liu calls it "price level distance to mid price." We have chosen to use Liu's designation for the measure.

level. The price level distance to the mid price on the ith price level on the bid (b) or ask (a) side from the mid price $m(t)$ at time t is defined as

$$\xi_{t,i}^{b,a} = \frac{p_{t,i}^{b,a}}{m(t)} - 1, \tag{42}$$

where $p_{t,i}^{b,a}$ is the price. [60] We note that the similar, and arguably more intuitive, measure that is the number of ticks away from the mid price does not seem to be used in any articles (other than for the best bid and ask prices).

2.3.2.2 *Action Space Representation*

Previous research on RL and MM is mainly concerned with pricing, i.e., selecting the bid and ask prices. This is natural since the prices directly impact how much and in which manner the market maker will trade. Some other aspects are also considered in various experiments, such as the option to reduce the inventory by placing a market order.

Bid and Ask Prices

There exist some different approaches to action spaces for bid and ask prices in the literature. A popular design is to let the action be chosen among a predetermined set of bid and ask pairs. One such example is [60], see Table 2.3. Similar action spaces are used in [62, 64]

Action ID	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Bid	0	0	0	4	4	4	4	9	9	9	9	14	14	14	14
Ask	4	9	14	0	4	9	14	0	4	9	14	0	4	9	14

Table 2.3: Example of action space with a predetermined set of bid and ask pairs. 15 out of the 17 actions comprising the action space in [60]. The bid and ask numbers refer to the number of ticks/price levels from the mid price. The two other actions not included are doing nothing and placing a market order to clear the inventory.

Another possible action space construction is to not predetermine the bid and ask pairs, but allow the agent to freely choose bid and ask depths independently of each other. This is for instance what is done in [79].

One additional approach is provided by Chan and Shelton in [12], where the action is the *change* of the MM's current bid and ask prices. The actions are discrete and limited by maximum allowable changes for the bid and ask prices. Similar actions are used in [45].

A critical remark is that all previous works that we have studied only allow the market maker to place orders on one bid and one ask level at a time. In practice, some market makers place limit orders on several bid and ask levels.

Volumes

It is relatively uncommon to construct the action space such that the agent can choose the limit order volume, but it is done in [7, 45]. More generally, the volume is simply fixed at for instance 1 share (e.g., [12]) or 100 shares (e.g., [48]).

MiFID II and Market Making
The Markets in Financial Instruments Directive (2014/65/EU) of the European Parliament and of the Council, also known as *MiFID II*, is a legislative framework for securities markets, investment intermediaries, and trading venues in the European Union. One of its main objectives is increased investor protection and related to this, the framework governs how and when market makers must fulfill certain duties. Specifically, the Commission Delegated Regulation (EU) 2017/575 [18] which supplements 2014/65/EU states that market makers (in equity markets) shall (among other things):

(a) provide firm quotes, which means that the orders and quotes it provides under the rules of a trading venue can be matched against an opposite order or quote

(b) provide simultaneous two-way quotes, which means that both the bid and the ask prices must be present in the order book at the same time

(c) provide two quotes deemed of comparable size, which means that their sizes do not diverge by more than 50% from each other

(d) provide competitive quotes, which means that they are posted at or within the maximum bid-ask range set by the trading venue and imposed upon every investment firm that has signed a market making agreement with that trading venue

Primer 4: MiFID II and Market Making.

Market Orders

Spooner et al. include an action that is to at time t post a market order to clear (parts of) the market maker's inventory volume Q_t, i.e., $\omega_x = -\alpha Q_t$ (rounded to nearest feasible order size), $\alpha \in (0, 1]$ [64]. Notably, the authors make no mention of the value of α used in their numerical experiments. A similar setting is presented by Ganesh et al., where an action is to hedge a fraction of the current inventory (i.e., place a market order) [24]. Sadighian also includes an action which is to post a market order with a size equal to the market maker's entire inventory [60].

No Action

Some authors have chosen to include an action which is to do nothing, i.e., provide no quotes. In [60], this means to provide no bid and no ask prices, whereas in [80] the market maker can choose to provide no bid, provide no ask, or provide neither. In the EU, the legislative framework MiFID II stipulates that market makers must *"carry out this market making continuously during a specified proportion of the trading venue's trading hours, except under exceptional circumstances, with the result of providing liquidity on a regular and predictable basis to the trading venue"* (Article 17(3)a MiFID II, [17]), meaning that a market maker may not stop providing two-side prices when or how they wish. Therefore, in some cases, it does make sense to consciously not include the action to do nothing in the action space. Additionally, not providing continuous two-way prices for clients may harm the business relations. See Primer 4 for some more details about MiFID II.

2.3.2.3 *Rewards*

As previously discussed, formulating and designing an appropriate reward is critical for the success of any application of reinforcement learning. In complex cases, it may be challenging to translate goals into a reward function. For instance, it may be desirable in market making to teach an RL-based MM agent to be risk-averse, but how should one translate that ambiguous goal into something quantifiable?

Gašperov et al. find that all previous studies on reinforcement learning for market making use PnL-based rewards (PnL = *"profit and loss"*). This makes perfect sense, as the ultimate aim of market makers is to maximize their (potentially risk-adjusted) returns. The PnL-based component of the reward can be constructed in numerous ways, and other non-PnL-based components are often included to achieve a desired behavior. [25]

Reward Density

Before we expand on the specific components of the reward, we briefly discuss the reward sparsity in various studies. A well-known issue in reinforcement learning is that sparse rewards negatively affect learning, and many authors seem to have considered this issue. Very few papers, e.g., [12, 45, 79], compute the reward only on the terminal time step / at the end of an episode. Lim et al. formulate the issue of sparse rewards and utility nicely in a non-mathematical way:

> *"Existing approaches to the market making problem generally seek to maximise the expected utility of some economic measure of the agent. These utilities are meant to evaluate the performance of the agent at the end of a trading period and do not accurately represent immediate rewards at each timestep t."* [48]

For more about utility, see Primer 3.

PnL

The large variety of formulations of PnL measures used throughout the literature is too extensive to cover in this thesis entirely, but we will present a few to demonstrate the variety.

The *unrealized PnL* is in [61] defined as

$$\Delta UPnL_t = Q_t \Delta m(t), \tag{43}$$

where $\Delta m(t)$ is the relative change in mid price, $\Delta m(t) := \frac{m(t)}{m(t-1)} - 1$. An advantage of this measure is the dense feedback since zero-valued rewards are very improbable.

The *realized PnL change* is also defined as

$$\Delta RPnL_t = RPnL_t - RPnL_{t-1}, \tag{44}$$

where $RPnL_t$ is the realized PnL up until time step t [61].

In [65], the *"PnL"* value (commonly called the *change in mark-to-market*, which is the term we use) from the time step $[t-1, t]$ is defined as

$$\Delta MtM_t = \delta_t^- q_t^b - \delta_t^+ q_t^a + Q_t \Delta m(t), \tag{45}$$

where q_t^b and q_t^a are the volumes bought and sold, respectively, in the time interval $[t-1, t]$. The notation is similar to the one used to denote the volume on the best bid and ask at time t, $q^b(t)$

and $q^a(t)$, but they are not the same. The measure described here is also referred to as *positional PnL* and *unrealized PnL with realized fills* in other articles. Note that we define the volumes consistently throughout this thesis such that $q_t^b \geq 0$ and $q_t^a \leq 0$, why the minus sign before $\delta_t^+ q_t^a$ enters the expression.

Spooner et al. define a *symmetrically dampened PnL*

$$\Delta SDPnL_t = \delta_t^- q_t^b - \delta_t^+ q_t^a + Q_t \Delta m(t) - \eta Q_t \Delta m(t) \tag{46}$$

and an *asymmetrically dampened PnL*

$$\Delta AsDPnL_t = \delta_t^- q_t^b - \delta_t^+ q_t^a + Q_t \Delta m(t) - [\eta Q_t \Delta m(t)]_+ \tag{47}$$

where η is a scale factor. To quote Spooner et al., the scale factor

> *"[...] reduces the reward that the agent can gain from speculation. Specifically, the symmetric version dampens both profits and losses from speculation, while asymmetric dampening reduces the profit from speculative positions but keeps losses intact. In both cases, the amount of reward that can be gained from capturing the spread increases relative to the amount of reward that can be gained through speculation, thus encouraging market making behaviour."* [64]

In [64], the impact of η is varied and its impact on the daily PnL is investigated. The authors find that $\eta = 0.6$ seems to be a good choice that reduces variance. The difference between the symmetrically and asymmetrically dampened PnL measures may seem minute, but the standard deviation in the normalized daily PnL differs by at least one order of magnitude across different numerical experiments in [64].

In [48], the reward at time $t < T$ is

$$R_t = a(V_t - V_{t-1}) + \exp\{b\tau_t\} sgn(|Q_t| - |Q_{t-1}|), \tag{48}$$

where a and b are constants, V_t is the *value* of the agent (not further defined by the authors, but presumably it refers to the MtM-value or a comparable measure), and τ_t is the remaining trading time. At $t = T$, the reward is given by

$$R_T = \alpha - \exp\{-\gamma(RPnL_T - Q_T p_T^L(Q_T))\}, \tag{49}$$

where α is some constant, γ is a risk aversion parameter, $RPnL$ is defined as above, $p_t^L(q)$ is the volume-weighted average liquidation price of the volume q at time t. The form of the reward on the terminal time T is very akin to the CARA-utility function. A few other papers, e.g., [7, 79], also employ some utility of the PnL as the reward.

Inventory Penalization

In a similar fashion to the inclusion of penalization of the running inventory in the example of Cartea et al. [11], many RL applications in market making also include some kind of penalization in addition to a PnL component. One such example has already been presented above; namely, the symmetrically and asymmetrically dampened PnL measures by Spooner et al. [64]. Other examples include [12, 26, 65]. It varies in the literature whether the absolute or squared value of the inventory is used as a penalizing term.

Sharpe Ratio

Sadighian ([61]) uses the *Differential Sharpe Ratio*, an online version of the Sharpe Ratio, as a risk-adjusted measure of return. The standard Sharpe Ratio of a portfolio p is defined as

$$SR_p = \frac{R_p - R_f}{\sigma_p} \tag{50}$$

where R_p is the portfolio's return, R_f is the risk-free rate of return, and σ_p is the standard deviation of the portfolio's excess return. The reader is referred to [61] for the Sadighian's definition of the Differential Sharpe Ratio.

Other Rewards

There are many more reward functions in the literature, two of which are presented here.

First, Chan and Shelton include *market quality* measures, including the bid-ask spread, market depth, and time-to-fill of a market order [12].

Furthermore, Sadighian defines a *trade completion* reward function that discourages the agent from long-term price speculation by actively rewarding the opening and closing of "positions with a targeted price-to-loss ratio." [61]

2.3.2.4 *Other Reinforcement Learning Aspects*

The following section will explore some remaining relevant aspects of reinforcement learning for market making.

Algorithm

As stated above, several reinforcement learning algorithms can be used to train the agent. They all have their advantages and disadvantages and thus choosing the *best* algorithm is heavily influenced by the nature of the problem and how it is implemented.

Going back to the article written by Gašperov et al. [25], we find that 12 out of 23 used DRL algorithms while the remaining used non-deep RL algorithms. The algorithms used in the papers mentioned in Section 2.3.2 are summarized in Table 2.4.

Method	**Deep?**	**Number of uses**	**Sources**
Q-learning	No	5	[48, 51, 62, 64, 80]
SARSA	No	4	[12, 32, 53, 64]
Monte Carlo	No	1	[12]
R-learning	No	1	[64]
Less known methods	No	4	[31, 33, 65, 73]
Actor-Critic methods (e.g., A2C)	Yes	5	[7, 12, 30, 60, 61]
Proximal Policy Optimization (PPO)	Yes	3	[24, 60, 61]
DQN	Yes	2	[58, 62]
DDQN	Yes	1	[58]
Less known methods	Yes	3	[5, 26, 79]

Table 2.4: Summary of reinforcement learning methods used in the papers mentioned in Section 2.3.2.

Since the problems the authors try to solve in their respective articles differ pretty substantially, one cannot deem one of the above algorithms objectively superior for our task. However, it does make the choice easier.

Learning Rate

Less than half of the included articles mention anything about the learning rate schemes they use. While the spread of the learning rates is quite broad, most values lie between 10^{-2} and 10^{-4}. Furthermore, there seems to be no pattern regarding the usage of constant versus decaying learning rates [24, 30, 48, 53, 60–62, 64]. Looking into the articles using a decaying learning rate, both linear and exponential decay are represented.

Interestingly, the learning rate for the critics in actor-critic models is much lower, all in the vicinity of 10^{-8} [30].

Exploration Rate

While most articles mention the importance of exploration for convergence to a good policy, few mention which exploration method they use, and even fewer mention the exploration rate scheme they use; however, those who mention which method they use all use the ϵ-greedy approach. Furthermore, of the four articles explicitly stating their exploration rate scheme, three use a decaying learning rate, [49, 64] and one uses a constant rate [53]. The scheme of those using a decaying learning rate can be summarized as follows: start with a exploration rate of ~ 1 and decay linearly to ~ 0.05 during half the episodes, after which the exploration rate is kept constant at that level.

Discount Factor

As mentioned in Section 2.1.1, the discount factor γ is used to ensure convergence for episodes of infinite length. However, choosing $\gamma < 1$ can also aid convergence.

Like the factors discussed above, most articles on RL for market making state the thought behind and the importance of the discount factor, but very few disclose the actual value they use.

Of the articles included in this literature review, five of them stated the value they used for the discount factor – they were $\gamma \in \{0.9, e^{-1}, 0.9, 1, 0.97\}$ [32, 33, 53, 62, 64]. Note that the value $\gamma = e^{-1}$ is much lower than what it usually is in practice [33]. Thus we conclude that choosing the discount factor such that it is in the vicinity of $0.9 - 1$ should yield good results.

2.3.3 *Benchmark Strategies*

In addition to comparing the market making strategies obtained from reinforcement learning with analytically optimal solutions, it may also be interesting to use other benchmarks. Two such benchmarks are fixed offset and random strategies. Among analytical solutions, the Avellaneda-Stoikov model is the most popular choice [25].

2.3.3.1 *Fixed Offset*

A simple – and in no sense optimal – market making strategy is the *fixed offset* strategy. The strategy entails continuously posting buy and sell limit orders at a predetermined number of ticks away from the prevailing mid price (or another reference point). There exist enhanced offset

strategies, such as the volatility-dependent offset and the trend-dependent offset, but in this thesis, we only present the fixed offset strategy as this will be used as a benchmark. [1] The fixed offset strategy is used as a benchmark in for instance [26, 32, 48, 64, 79].

2.3.3.2 *Random Strategy*

Another benchmark market making strategy is when the market maker randomly selects the depth and/or volume on which to post buy and sell limit orders. In this strategy, an action is selected by sampling uniformly from the entire action space. The random strategy is used in for instance [24, 48, 80]. In this thesis, the order volumes are also fixed for the random strategy.

2.4 EVALUATION METRICS

In order to successfully train an agent to make markets optimally, it is essential to define what constitutes desirable and undesirable results clearly. In this section, some of these techniques will be explored.

While reinforcement learning has been used to solve harder and harder problems in recent years, the reproducibility of state-of-the-art DRL methods has become increasingly more difficult. In the article *Deep Reinforcement Learning that Matters*, Henderson et al. discuss intrinsic (e.g., random seeds or environment properties) and extrinsic (e.g., hyperparameters or codebases) factors that affect reproducibility. Factors such as *hyperparameters*, *network architecture*, *reward scale*, *random seeds and trials*, *environments* (e.g., OpenAI Gym), and *codebases* significantly impacted the results. Due to the unstable nature of most RL algorithms, metrics such as *average cumulative reward* and *maximum reward* are typically inadequate for comparison. Instead, they suggest using more robust metrics such as *confidence bounds*, *power analysis*, and *significance*. Furthermore, they underline the importance of reporting *"[...]* all *hyperparameters, implementation details, experimental setup, and evaluation methods for both baseline comparison methods and novel work"* to not let future researchers waste time on futile efforts of reproducing state-of-the-art works [35].

Since this thesis concerns applying (D)RL to market making – a relatively new application – considering these points is crucial.

2.4.1 *Confidence bounds*

Typically one uses resampling with replacement to generate a statistically relevant mean and confidence bound. This technique can then be used to gain insight into what, e.g., the 95% confidence interval of the results is.

Given a sample $\{x_i\}_{i=1}^n$, which is assumed to be normally distributed, a two-sided confidence bound of level α is given by

$$\overline{x} \pm \lambda_{\frac{\alpha}{2}} \frac{s}{n}, \tag{51}$$

where $\overline{x}$ is the mean of the sample, $\lambda_{\alpha/2}$ is the α-quantile of the normal distribution, s the sample standard deviation, and n the sample size. [9]

2.4.2 *Significance*

Significance is important to use for the reported gains when deciding on a reinforcement learning algorithm. However, many standard methods, such as k-fold t-tests for supervised learning, cannot be applied to reinforcement learning. [35]

In this paper, *Welch's t-test* will be used to compare strategies statistically. It is a two-sample location test that tests the null hypothesis that two populations have equal means. Its test statistic is

$$t = \frac{\bar{X}_1 - \bar{X}_2}{\sqrt{s_{\bar{X}_1}^2 + s_{\bar{X}_2}^2}} \tag{52}$$

where $s_{\bar{X}_i} = s_i/\sqrt{N_i}$. Here, $\bar{X}_i$ is the ith sample mean, $s_{\bar{X}_i}$ the ith standard error, s_i the ith standard deviation, and N_i the ith sample size. Whenever the degrees of freedom, approximated by

$$\nu \approx \frac{\left(\frac{s_1^2}{N_1^2} + \frac{s_2^2}{N_2^2}\right)^2}{\frac{s_1^4}{N_1^2\nu_1} + \frac{s_2^4}{N_2^2\nu_2}}, \tag{53}$$

reaches 30 and above, one may use the Gaussian distribution instead of the Student's t distribution in the p-value testing. Above, $\nu_i = N_i - 1$. In throughout this entire thesis, $\nu >> 30$, so a Gaussian distribution will be used to decide the p-values. [56]

The p-value is the probability of obtaining test results at least as extreme as the observed results, that is

$$p = \mathbb{P}(t(\boldsymbol{X}) \geq t(\boldsymbol{x})) \tag{54}$$

where $\boldsymbol{X}$ is the sample variable, $\boldsymbol{x}$ an observation of the sample variable and $t(\cdot)$ is the test-statistic. [9]

2.4.3 *Boxplots*

Boxplots are used to display the following characteristic of numerical data: locality, spread, and skewness. Given a sample of size n, its quartiles are used; the minimum $\widetilde{Q}_0$, the first quartile $\widetilde{Q}_1$, the median $\widetilde{Q}_2$, the third quartile $\widetilde{Q}_3$ and the maximum $\widetilde{Q}_4$.

Firstly, the boxplot consists of a box drawn between $\widetilde{Q}_1$ and $\widetilde{Q}_3$ with a horizontal line in the middle denoting the median, $\widetilde{Q}_2$. Secondly, it consists of a set of whiskers, which can be defined in several ways. [9] In this thesis, it shows 1.5 times the interquartile range (IQR), defined as $IQR = \widetilde{Q}_3 - \widetilde{Q}_1$. All points outside this range are displayed as circles.

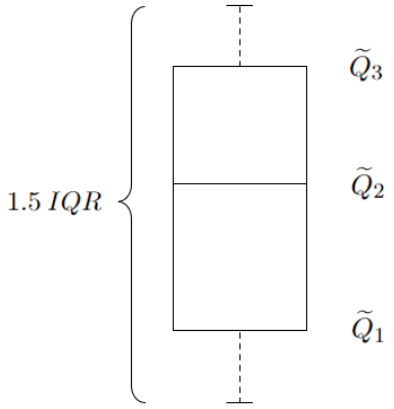


Figure 2.7: A graphical illustration of a boxplot. Showing the first quartile ($\widetilde{Q}_1$), the median ($\widetilde{Q}_2$) and the third quartile ($\widetilde{Q}_3$). Also displaying 1.5 times the interquartile range (IQR), which is the total size of the whiskers.

3 EXPERIMENTS AND RESULTS

This chapter constitutes the central part of this thesis with various experiments and corresponding results. The first section deals with experiments on the simple probabilistic model and compares market making strategies obtained via reinforcement learning with analytically optimal strategies. In the second section, the Markov chain LOB model is used, and both tabular and deep reinforcement learning methods are used and compared. Results for all methods are presented and analyzed.

All experiments were performed on either an *Intel(R) Core(TM) i7-11700K* on a personal computer or *Compute Optimized vCPU*s on Google Cloud Platform (GCP).

For a summary of the results, we encourage the time-restricted reader to skip ahead to Section 3.3 as Section 3.1 and Section 3.2 are exhaustive and not necessarily essential in order to grasp the main results.

3.1 SIMPLE PROBABILISTIC MODEL

Our first experiments tackle recreating the analytically optimal strategies of the simple probabilistic model (SPM) presented in Section 2.3.1.1. This will be done with tabular Q-learning.

To limit the size of the Q-table, the length of the episodes is limited to $T = 5$. The analytically optimal strategies thus differ slightly from those presented in Figure 2.5. These new strategies are presented in Figure 3.1. Note that $Q_t = \overline{q} = 3$ is not plotted for the optimal bid depths since the depth is put as $+\infty$ – the same is true for $Q_t = \underline{q} = -3$ for the ask depths.

3.1.1 *Model and Training Parameters*

We will now discuss the model's parameters and the training algorithm in depth.

3.1.1.1 *State Space*

The state space consists of two one-dimensional variables – the time t and the inventory Q_t. This state space was chosen because it contains the same variables used to derive the analytically optimal solution and results in a relatively small Q-table.

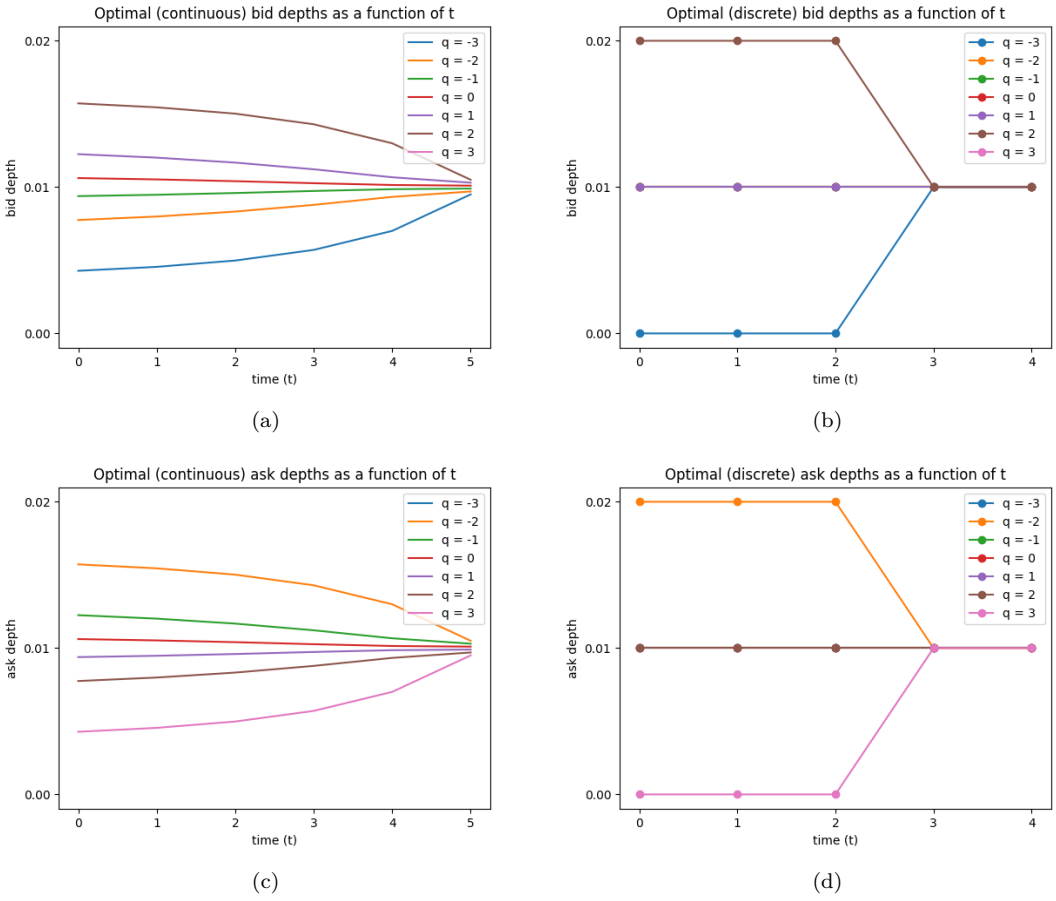
In order to make the state space finite, the inventory and the time are restricted in the following way

$$(t, Q_t) \in \mathcal{S} = \{0, 1, \dots, T\} \times \{\underline{q}, \dots, \overline{q}\}, \tag{55}$$

where $T = 5$ and $-\underline{q} = \overline{q} = 3$. With these parameter values we get $|\mathcal{S}| = 6 \cdot 7 = 42$.

3.1.1.2 *Action Space*

Similarly, the action-space also consists of two integer-valued one-dimensional variables – the first one representing the bid depth and the second one the ask depth. The values of the variables



(a) (b)

(c) (d)

Figure 3.1: **(a)** Optimal continuous bid depths, **(b)** optimal discrete bid depths, **(c)** optimal continuous ask depths, and **(d)** optimal discrete ask depths.
Parameters: $\phi = 10^{-5}$, $\lambda^{\pm} = 1$, $\kappa^{\pm} = 100$, $\overline{q} = -\underline{q} = 3$, $\alpha = 0.0001$, $\sigma = 0.01$, $\mu = 100$.

here represent the number of ticks away from the mid-price that the bid and ask orders should be posted.

The action-space is also restricted, by a parameter $\bar{d}$, in the following way

$$(\delta^-, \delta^+) \in \mathcal{A} = \{0, \dots, \bar{d} - 1\} \times \{0, \dots, \bar{d} - 1\}. \tag{56}$$

Since the optimal depth is at most two ticks away from the mid price, we choose $\bar{d} = 4$ to give the agent the possibility to make a "wrong" decision (i.e., selecting a depth greater than two). With this choice of $\bar{d}$, the size of the action-space is $|\mathcal{A}| = 4 \cdot 4 = 16$. Thus, the total size of the Q-table becomes $|\mathcal{S}| \cdot |\mathcal{A}| = 42 \cdot 16 = 672$.

One should also note that this limitation of the action-space makes it impossible for the agent to learn the analytically optimal depths at the volumes $\overline{q}$ and $\underline{q}$ since $+\infty$ is not included.

3.1.1.3 *Reward*

The reward suggested in the Section 2.3.1.1 will be used; that is

$$R_t = \Delta V_t - \phi Q_t^2, \tag{57}$$

where

$$V_t = \begin{cases} X_t + H_t, & 0 \leq t < T \\ X_{T^-} + Q_T(S_T - \alpha Q_T), & t = T. \end{cases} \tag{58}$$

Note that the agent receives no penalty for exceeding the inventory limits (this is possible since it cannot set the depth to $+\infty$).

3.1.1.4 *Training Parameters and Hyperparameters*

The agent is trained for 15,000,000 (1.5×10^7) episodes, ten separate times. Exploring starts for the starting volume is used to find the best strategy for $t = 0$ and increase the chances of visiting states with high absolute volumes early in the episodes. An ϵ-greedy policy is used to aid convergence to the optimal strategy. The exploration rates of both these methods are decreased linearly during training. The learning rate is decreased exponentially. The discount rate was set to $\gamma = 1$ since the episodes are of finite length.

The schemes for these parameters are summarized in Table 3.1 and in Figure B.1 in Appendix B.1. An exponentially decaying learning rate was used since it was observed that a higher learning rate only was needed for a short period of time for the agent to get close to the optimal action-values, which made a linearly decaying scheme unnecessary in terms of computational power.

For all experiments in this thesis that involve Q-learning, including this one, the initial Q-matrix was set to the null matrix.

	Starting rate	**Cut-off**	**Ending rate**	**Decreasing type**
Exploring starts	1	50%	0.05	Linear
ϵ-greedy	1	50%	0.05	Linear
Learning rate	0.5	100%	$5 \cdot 10^{-6}$	Exponential

Table 3.1: The schemes of the training parameters. *Cut-off* shows how far into the training the final rate is reached.

3.1.2 *Environment*

The original environment is explained in detail in Section 2.2.3.1, and the discretized is explained in Section 2.3.1.1.

The discretized version of the environment is implemented in Python using the standard design of environments in OpenAI's toolkit `Gym` to facilitate reinforcement learning (see [10]). The general scheme for the step function is presented in pseudocode in Algorithm 1. For the precise code, please see `simple_model_mm` on our GitHub repository, see Section 1.5. The model parameters used are presented in Table 3.2. We wrote and implemented the Q-learning algorithm ourselves rather than using a Python package for this. The training was done on the Intel(R) Core(TM) i7-11700K processor.

Algorithm 1 SPM's environment's STEP function

function STEP(`action` $\boldsymbol{a}$)
  update the mid price
  if $t \leq T$ **then**
    remove the current bid and ask prices
    set the new bid and ask prices according to the given action $\boldsymbol{a}$
    simulate the number of MO arrivals, a_t^b and a_t^s
    simulate the number of MOs that will be executed, e_t^b and e_t^s
    if $\underline{q} \leq Q_{t-1} - e_t^b + e_t^s \leq \overline{q}$ **then**
      execute the orders
    else
      adjust e_t^b or e_t^s such that $\underline{q} \leq Q_{t-1} - e_t^b + e_t^s \leq \overline{q}$
      execute the orders
  if $t = T$ **then**
    liquidate the MM's position
  calculate the reward R_t
  $t \leftarrow t + dt$

ϕ	10^{-5}	$\underline{q}$	-3
λ^+	1	$\overline{q}$	3
λ^-	1	α	10^{-4}
κ^+	100	σ	0.01
κ^-	100	μ	100

Table 3.2: Parameters of the SPM

3.1.3 *Results*

As stated above, the Q-learning was run ten separate times for 1.5×10^7 episodes each. A comparison of the ten different runs can be seen in the boxplots in Figure 3.2 and in Table 3.3. The largest difference in mean reward between two runs is less than 0.5%, indicating that the Q-learning on this particular problem is not especially sensitive to the seed used to simulate the environment. More evidence of this can be found in Figure 3.3. For example, the shaded area (representing ± one standard deviation) for the average reward and the state-value plot is small, meaning that the runs converge similarly. Here one should also note that the average reward

Run	$\overline{r}$	σ_r	$v_*(0,0)$
1	3.529×10^{-2}	3.213×10^{-2}	3.531×10^{-2}
2	3.514×10^{-2}	3.203×10^{-2}	3.522×10^{-2}
3	3.517×10^{-2}	3.207×10^{-2}	3.522×10^{-2}
4	3.528×10^{-2}	3.183×10^{-2}	3.525×10^{-2}
5	3.524×10^{-2}	3.212×10^{-2}	3.527×10^{-2}
6	3.524×10^{-2}	3.197×10^{-2}	3.524×10^{-2}
7	3.529×10^{-2}	$\mathbf{3.180 \times 10^{-2}}$	3.526×10^{-2}
8	3.529×10^{-2}	3.206×10^{-2}	$\mathbf{3.533 \times 10^{-2}}$
9	$\mathbf{3.532 \times 10^{-2}}$	3.209×10^{-2}	3.528×10^{-2}
10	3.518×10^{-2}	3.207×10^{-2}	3.517×10^{-2}

Table 3.3: The average reward received when following the obtained strategies for 10^6 episodes, the standard deviation of that reward and the state-value at $(t, Q_t) = (0, 0)$ following the strategy starting with $(1, 1)$ in the SPM. Bold values highlight the best result: the highest reward or the lowest standard deviation.

plot will have lower values than the state-value due to exploring starts, and the ϵ-greedy policy is included in the former.

The confidence interval with $\alpha = 0.95$ of the mean reward for all ten runs is $(0.03522, 0.03526)$ with the mean standard error 0.0320.

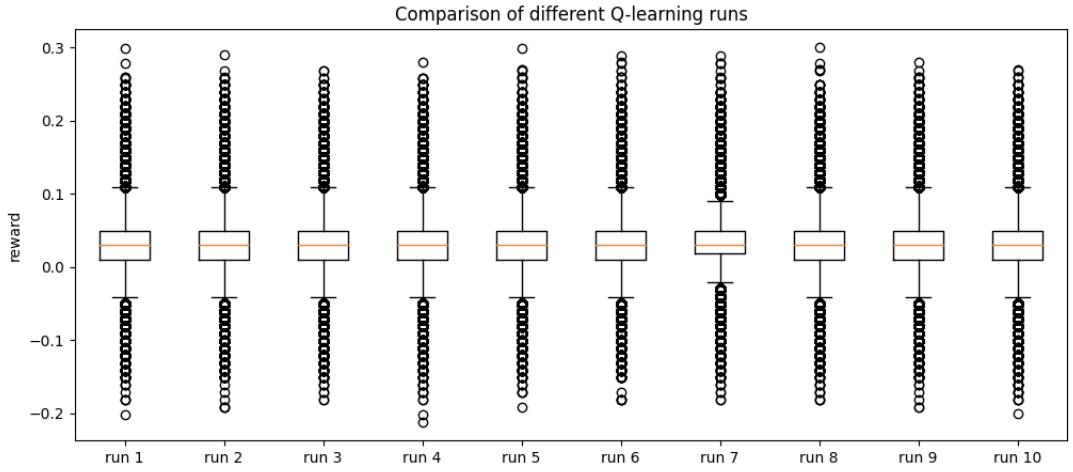


Figure 3.2: Boxplot of the reward of ten different Q-learning runs when evaluated for 10^6 episodes.

Figure 3.4 shows the optimal strategies obtained by averaging the ten runs' Q-tables. They are presented in two ways – the form used in Figure 3.1 as well as with heat maps. Note that in Figure 3.4a the depths for $Q_t = \overline{q}$ are set to ∞ and similarly for the depths $Q_t = \underline{q}$ in Figure 3.4b.

First and foremost, it is evident that the strategies are not the same as the analytically optimal strategies. Secondly, one can also notice that the bid and ask sides are somewhat asymmetrical (unlike the analytical strategy). Thirdly, almost all actions lie at a depth of one tick. Finally, in Table 3.4 one observes that the mean strategy finds the correct action (according to the

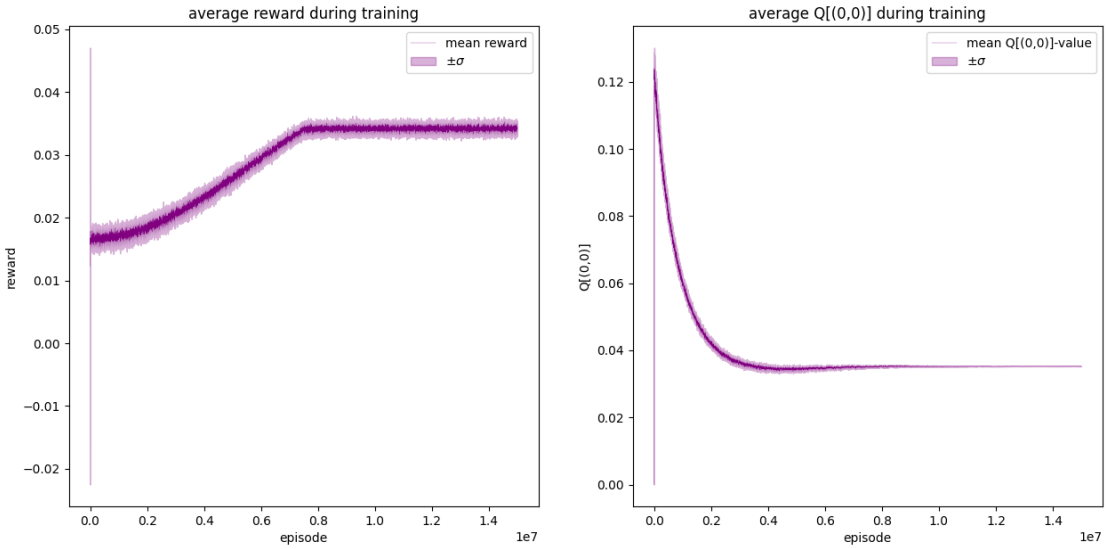


Figure 3.3:
Training of the SPM for 1.5×10^7 episodes.
Left: the average reward during training with a shaded area showing ± one standard deviation.
Right: the state-value at $(t, Q_t) = (0, 0)$ during training with a shaded area showing ± one standard deviation.

discrete analytically optimal) for 85% of the states. While these observations indicate suboptimal solutions, one should first compare them to benchmarks.

Run	Ask-depth	Bid-depth	Total
1	25 (83.3%)	25 (83.3%)	50 (83.3%)
2	26 (86.7%)	24 (80.0%)	50 (83.3%)
3	26 (86.7%)	24 (80.0%)	50 (83.3%)
4	26 (86.7%)	**27(90.0%)**	**53(88.3%)**
5	25 (83.3%)	**27(90.0%)**	52 (86.7%)
6	**27(90.0%)**	24 (80.0%)	51 (85.0%)
7	25 (83.3%)	25 (83.3%)	50 (83.3%)
8	25 (83.3%)	24 (80.0%)	49 (81.7%)
9	26 (86.7%)	25 (83.3%)	51 (85.0%)
10	25 (83.3%)	24 (80.0%)	49 (81.7%)
Mean strategy	26 (86.7%)	25 (83.3%)	51 (85.0%)

Table 3.4: How many of the optimal actions of the Q-learning runs that are the same as the discrete analytically optimal in the 30 different states. Bold values highlight the best result: the highest accuracy.

For this part, four different strategies are used as benchmarks. The two first strategies are two versions of the analytically optimal – the first is the optimal depths rounded to the closest tick (*analytical discrete*), and the second uses the continuous values for the depth (*analytical continuous*). The third strategy is where the MM always puts the bid and ask price two ticks away from the mid price (*constant*). The fourth and final strategy consists of the MM randomly choosing an action from the action-space (*random*).

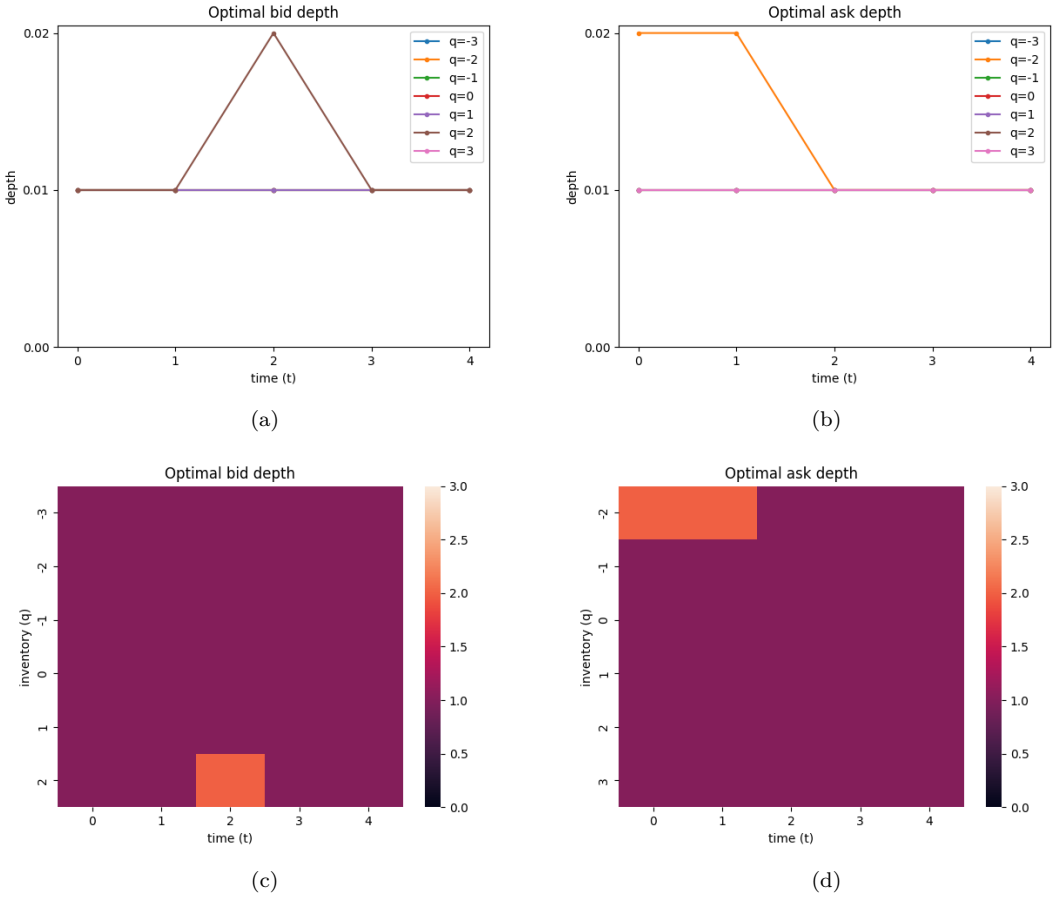


Figure 3.4: These four panels are obtained from the Q-matrix obtained by averaging the Q-matrices of all ten runs. **(a)** shows the optimal bid depths, **(b)** shows the optimal ask depths, **(c)** shows a heat map of the optimal bid depths, and **(d)** shows a heat map of the optimal ask depths.

A comparison of the ten different runs can be seen in the boxplot in Figure 3.2 and in Table 3.3. Unsurprisingly, the constant and the random strategies perform the worst. However, one surprisingly notices that the mean strategy outperforms both of the analytically optimal strategies (discrete and continuous). We also note that the best single Q-learning strategy is beaten by the mean strategy, indicating some noise in the strategies being averaged out.

Looking further into the stability of the strategies, we notice that excellent stability of the optimal actions and state-values is achieved with a small absolute inventory. However, the strategies are less stable for higher absolute volumes, especially for the early parts of the episode. This is explored further in Appendix B.2.

Table 3.3 and Figure 3.5 indicate that the Q-learning algorithm could find better strategies than the analytically optimal – which seems like an oxymoron. The critical thing to notice is that our discretization of the original model of Cartea et al. most likely renders the analytically optimal strategy suboptimal. This is due to two reasons.

Strategy	$\overline{r}$	σ_r
Q-learning		
- *best run*	3.529×10^{-2}	3.211×10^{-2}
- *mean strategy*	$\mathbf{3.542 \times 10^{-2}}$	3.205×10^{-2}
Analytical		
- *discrete*	3.506×10^{-2}	3.132×10^{-2}
- *continuous*	3.537×10^{-2}	3.060×10^{-2}
Benchmark		
- *constant*	2.674×10^{-2}	$\mathbf{2.848 \times 10^{-2}}$
- *random*	1.852×10^{-2}	3.520×10^{-2}

Table 3.5: The average reward received when following the different strategies for 10^6 episodes. Bold values highlight the best result: the highest reward or the lowest standard deviation.

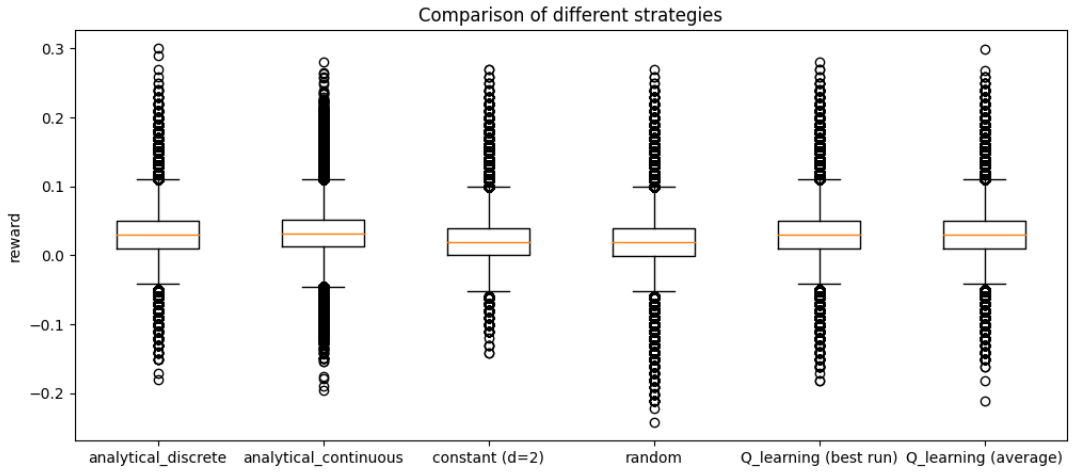


Figure 3.5: Boxplot of the reward of different strategies when evaluated for 10^6 episodes.

Firstly, the model is discretized in terms of time. In the original model MOs arrive one at a time at continuous times, while in the discretized version, the MOs arrive in bundles once every

time step. While the market maker can adapt its depths after every MO arrives in the former model, more trades can happen in the latter. This can be explained with an example: in the continuous model, when $Q_t = \overline{q}$ and a MO sell arrives at $t_1 = 0.5$ and a MO buy arrives at $t_2 = 0.7$ only the MO buy will be executed since the inventory cap cannot be exceeded. However, in the discrete model, these MOs will be handled as arriving simultaneously, $t = 1$ ($dt = 1$), allowing the MM to execute both of them without exceeding the cap and thus earning the whole spread.

Secondly, the analytically optimal depths are discretized by rounding them to the closest tick. This is quite a heuristic approach – there is no guarantee that this is sensible. For instance, rounding a depth of 0.004 to 0 (at the mid price) instead of to 0.01 (one tick away from the mid price) means that the market maker will not earn any spread at all on that trade, albeit it may be used to reduce the penalty received from holding a large absolute volume.

With these things in mind, we can now interpret the results. Note that the bid depth at $Q_t = \overline{q}$ and ask depth at $Q_t = \underline{q}$ can be ignored when comparing with the analytical strategies. In the original model, these are put to $+\infty$ in order not to allow any trades, but this cannot be done with the chosen action space. Since trades that would have led to the inventory exceeding the limit are nullified, the optimal depths at the inventory limits are only meaningful when the number of executed buy and sell MO cancel each other. Thus, these depths (the bid depth at $Q_t = \overline{q}$ and ask depth at $Q_t = \underline{q}$) have been excluded from Figure 3.4 – however, they are still included in the heatmaps in Figure B.2, partly explaining the instabilities of levels $Q_t = \overline{q}$ and $Q_t = \underline{q}$.

Looking at the strategies in Figure 3.4, one observes that the optimal action is almost always 1 in depth for both the bid and ask. It only differs for the bid depth at $Q_t = 2 = \overline{q} - 1$ and ask depth at $Q_t = -2 = \underline{q} + 1$, similarly to the analytically optimal strategy in Figures 3.1b and 3.1d. However, it only differs for a few values of t, and they are not the same for the bid and ask side, implying that the training has not fully converged since the environment is symmetrical. The (almost) constant depth learned by the agent supports the theory that rounding to the closest tick is not the best way to discretize the analytically optimal solution. Looking at the mean rewards in Table 3.5 we also notice that the difference between the strategies is minimal (except for constant and random), indicating that small changes in the strategy do not have a large effect on the average reward. This is probably due to the short time period used for the episodes and the low volatility chosen ($\sigma = 0.01$, see Table 3.1).[1]

Another interesting observation is the standard deviation of the different strategies. Unsurprisingly, the random strategy has the largest standard deviation, and the constant (with $\delta^{\pm} = 2$) has the lowest. However, quite surprisingly, the Q-learning strategies have a higher standard deviation than both versions of the analytically optimal strategies. The analytical continuous having the lowest standard deviation can be explained by it not having to round its depths. Nevertheless, the higher standard deviation of the Q-learning strategies may mean that they are inferior to the analytically optimal solutions when measured with utility functions such as HARA and CARA or portfolio theory measures such as mean-variance analysis.[2]

[1] Note that this is the volatility over a 1-second period and not the annualized volatility.

[2] For the reader interested in a thorough introduction to portfolio theory and mean-variance analysis, we recommend [40].

		Analytical	
		- *discrete*	- *continuous*
Q-learning	- *best run*	1.28×10^{-7}	0.974
	- *mean strategy*	5.33×10^{-16}	0.173

Table 3.6: The p-values for which the null hypothesis that the analytical strategies' average rewards are larger than the Q-learning's strategies' average rewards can be rejected. Welch's t-test with a common sample size of $n = 10^6$.

Further exploring the variance of the strategies, one may also look at the statistical significance of the Q-learning strategies outperforming the analytical strategies. In Table 3.6, the p-values of rejecting the null hypothesis that the analytical strategies rewards are larger than the Q-learning's strategies rewards are presented. The low values in the left column show that both the Q-learning strategies outperform the analytical discrete strategy with high statistical significance. However, the right column shows that the higher mean reward for the average Q-learning strategy than the analytical continuous is not statistically significant, despite the large sample size of 10^6.

3.2 MARKOV CHAIN LOB MODEL

Our subsequent experiments use the Markov chain LOB model described in Section 2.2.3.2. Initially, we keep the state-action space relatively small (without loss of any LOB model complexity) and use tabular Q-learning (Section 3.2.2), after which we move on to deep reinforcement learning on a much greater state space (Section 3.2.3). In addition, two different settings of the model will be used – a short-horizon (SH) and a long-horizon (LH) – which will be explained in more detail below. Finally, before proceeding to the experiments and the results, we briefly cover the Markov chain LOB environment in further detail.

3.2.1 *Environment*

This section describes the limit order book environment and its various aspects.

Action Frequency

In our setting, the market maker will only have the chance to update its quotes every dt seconds. It is technically feasible in this model to allow the market maker to update its quotes arbitrarily often, e.g., following virtually every order book event. However, a more reasonable assumption is that the market maker is somewhat too slow to do this, which is why we have chosen to introduce this constraint, in combination with the fact that the event frequency in this model is not particularly high. To provide some context here, the mean number of events per second in the MC model is around 4.8. Furthermore, SEB typically updates its FX prices around 20 times per second. They have the technical ability to update the prices significantly more often, but other financial actors do not necessarily appreciate such a high frequency. A higher frequency of price updates can clog various client systems and cause queues and price lags, which are not appreciated by anybody. Therefore, we choose to be conservative and will in our experiments let $dt = 1$ and $dt = 5$. In their research paper on reinforcement learning and market making, Lim et al. let their agent submit new orders every 10 seconds, i.e., $dt = 10$ in our setting [48].

Model Setup

Two settings will be considered for the Markov chain model – the short-horizon (SH) and the long-horizon (LH) – defined by their trading interval length. For the SH we set $dt = 5$ and $T = 25$, equaling five actions per episode, and for the LH we set $dt = 1$ and $T = 1,000$, equaling a thousand actions per episode. In order for the market maker to be able to trade the same average volume per second, the order size has been set to 25 for the SH and 5 LH.

The SH setting was chosen to get reasonably comparable results to the Q-learning on the SPM. The LH setting was chosen to simulate real-world market making more closely as it equals ~ 15 minutes of real time.

Note that the state space representation will differ while the underlying environment will remain the same for Q-learning and DDQN.

Order Depth

An important distinction compared to the SPM is how we have defined the order depths in the MC model. Instead of specifying the order depths as the distances from the mid price, we define

the bid depth δ^- as the number of ticks between the market maker's bid price and the best ask price, excluding the market maker's ask price. Equivalently, the ask depth δ^+ is the number of ticks between the market maker's ask price and the best bid price, excluding the market maker's bid price; this is illustrated in Figure 3.6.

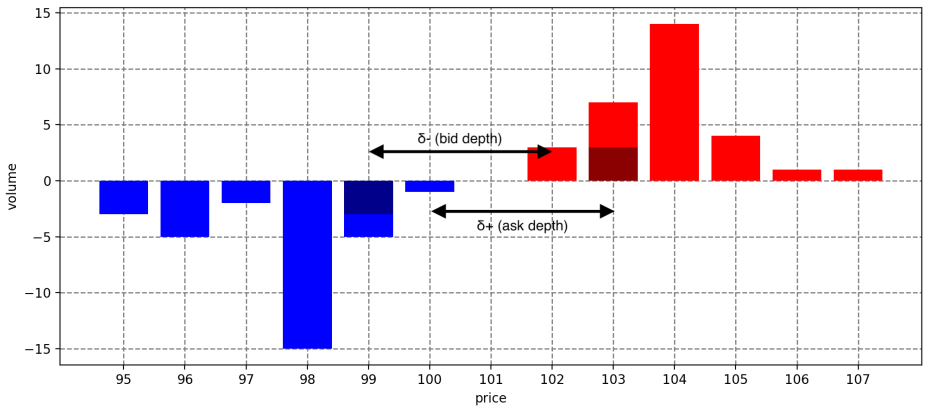


Figure 3.6: The order depths in the MC LOB model are illustrated. The market maker's orders are marked in a darker shade.

Invalid Actions

The action space is constant for all states in the model, which may result in invalid actions under certain circumstances. Consider an action with order depths (δ^-, δ^+). If the sum of the order depths does not exceed the bid-ask spread (momentarily before the market maker places the orders), i.e., $\delta_t^- + \delta_t^+ \leq s(t^-)$, the order depths are considered invalid, and adjustments to the depths are made automatically. To illustrate the problem that arises, see Figure 3.7. There, $s(t^-) = 5$ and $a(t^-) = 9,996$, and we suppose that the market maker's order depths are $\delta_t^\pm = 2$, which means that the market maker will post bids on $9,994$ and asks on $9,993$, and the MM would trade with itself since the two orders would cross. This is undesirable partly because it fills no purpose and partly because *wash trading* (trading with oneself) is illegal in many markets, so the order depths must be adjusted.

To adjust the order depths, we define the quantity $\Delta = [s(t^-) - (\delta_t^+ + \delta_t^-) + 1]_+$ (we call it the *order depth shortfall*), which is the number of ticks by which the spread exceeds the sum of the order depths. (Note that $[\cdot]_+$ denotes the positive part.) To select new order depths, we update them as

$$\delta_t^\pm \leftarrow \delta_t^\pm + \lceil \Delta/2 \rceil. \tag{59}$$

Here, $\lceil \cdot \rceil$ denotes the ceiling function. Note that the adjustment in Equation (59) is only made to invalid actions per our definition of the order depth shortfall since $\Delta = 0$ for all valid actions.

For the example described above, the updated order depths would be $\delta_t^\pm = 3$, and the LOB would be as in Figure 3.8.

Notably, the adjustment above is also made for the fixed offset and random strategies, and a fixed offset $\delta^\pm = 1$ can be interpreted as the strategy that posts two-way prices as tightly as possible.

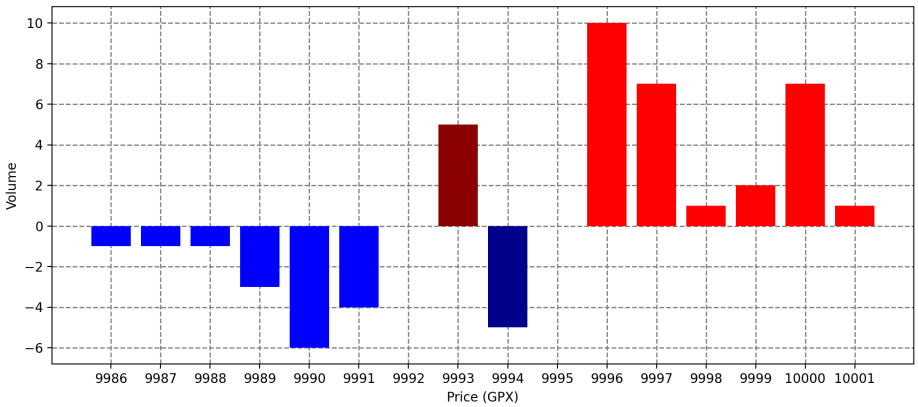


Figure 3.7: The LOB when the market maker posts cross orders. The market maker's orders are marked in a darker shade.

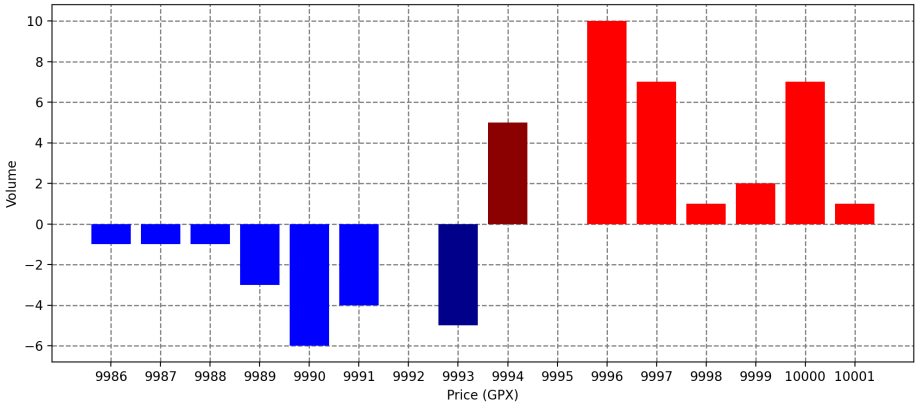


Figure 3.8: The LOB after the market maker's order depths have been adjusted. The market maker's orders are marked in a darker shade.

Market Impact

We made two interesting observations during our initial experimentation with the MC LOB model and market-making strategies. First, for a fixed offset strategy with $\delta^{\pm} = 1$ and a constant order volume of 5, the market maker succeeded in stabilizing the market with its increased liquidity, see Figure 3.9. The plot on the left shows the mean mid price for 100 episodes with $T = 1,000$ where no market maker is present, with $\pm$ a standard deviation shaded in purple. The middle plot shows the same but when a market maker with the aforementioned strategy is present. The figure clearly shows that the standard deviation is notably smaller, which may be interpreted as increased price stability. Lastly, the right-most figure illustrates our second exciting observation. A market maker providing skewed prices does have a noticeable price impact on the mid price. Here, the market maker continuously provides two-way prices with depths $\delta^{-} = 1$ and $\delta^{+} = 2$, a behavior that drives the prices upwards.

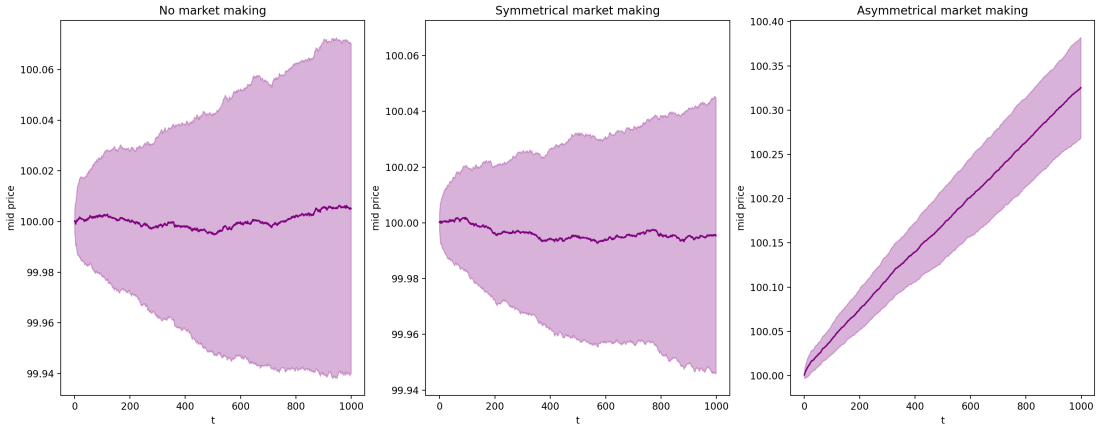


Figure 3.9: A comparison of how symmetrical and asymmetrical market making strategies impact the mid price when evaluated for $1,000$ episodes. The standard deviations are at the final time step 0.065 and 0.049 for the *no market making* (**left**) and the *symmetrical market making* (**middle**), respectively. The drift for the *asymmetrical market making* (**right**) is approximately $4.8 \cdot 10^{-2}$ ticks per second.

Order Priority

A final remark is that we adopt NYSE's order precedence hierarchy and let the secondary precedence rule be that our market maker agent has first priority at a given price level (see Primer 1). This is purely to achieve a lesser computational and time complexity.

Learning Environment

The reinforcement learning environment for the Markov chain LOB model follows the standard design of environments in OpenAI's toolkit `Gym` (see [10]). The general scheme for the step function is presented in pseudocode in Algorithm 2. For the precise code, please see `mc_environment` and `mc_environment_deep` on our GitHub repository, see Section 1.5.

3.2.2 *Q-Learning*

We implement a standard Q-learning algorithm to train the market maker agent in the Markov chain LOB model. It will mainly be used as a baseline for deep reinforcement learning. The training was done on GCP with eight compute-optimized vCPUs.

3.2.2.1 *State Space*

Unlike in the simple probabilistic setting, we do not introduce a limit on the maximum inventory. Thus, using the "raw" inventory as a state space variable is not feasible as the state space would be prohibitively large. Instead, we use bins akin to how Lim et al. [48] treats the inventory.

In the LH setting, we do a similar treatment to the time state space variable, and in the SH setting, we keep the time as it is. Concerning both instances, there are $\nu^{(T)}$ $(= 5)$ steps in the time state space variable, with the difference being that dt changes. To distinguish these experiments from the DDQN experiments, we name the short-horizon setting for Q-learning (SH-Q) and the long-horizon setting for Q-learning (LH-Q).

Algorithm 2 MC environment's STEP function

function STEP(`action` $\boldsymbol{a}$)
  if $t < T - dt$ **then**
    if $\boldsymbol{a}$[`market order`] = `true` **then**
      if $Q_t < 0$ **then**
        `order size` $\leftarrow \min(-Q_t, q^a(t))$
        place MB with volume `order size`
      else
        `order size` $\leftarrow -\min(Q_t, -q^b(t))$
        place MS with volume `order size`
    place limit orders with default size on depths $\boldsymbol{a}$[`depths`][`bid`] and $\boldsymbol{a}$[`depths`][`ask`]
  else
    liquidate inventory
  simulate `events` in LOB for dt seconds
  for `event` in `events` **do**
    if `event`[`type`] = `MB` and `event`[`price`] = `mm ask` **then**
      match against mm's sell order, depending on priority and remaining order volume
    else if `event`[`type`] = `MS` and `event`[`price`] = `mm bid` **then**
      match against mm's buy order, depending on priority and remaining order volume
  remove market maker's currently outstanding limit orders
  $t \leftarrow t + dt$

The state space $\mathcal{S}$ is in the LH-Q case defined as

$$\mathcal{S}_{LH-Q} = \mathcal{T} \times \mathcal{Q} = \{1, 2, \dots, \nu^{(T)}\} \times \{-\nu^Q, \dots, \nu^Q\}, \tag{60}$$

and in the SH-Q case

$$\mathcal{S}_{SH-Q} = \mathcal{T} \times \mathcal{Q} = \{0, dt, \dots, T-1\} \times \{-\nu^Q, \dots, \nu^Q\}. \tag{61}$$

Here we have $\nu^{(T)}$ bins for the time state space variable and $2\nu^Q + 1$ bins for the inventory state space variable. We let $\nu^{(T)} = 5$ and $\nu^Q = 3$, yielding a state space size of $|\mathcal{S}_{LH-Q}| = |\mathcal{S}_{SH-Q}| = |\mathcal{T}| \cdot |\mathcal{Q}| = 35$.

At time t, the time state space variable will be

$$\ell = \left\lceil \frac{t}{T} \eta^T \right\rceil + \mathbb{1}_{t=0}, \tag{62}$$

in LH-Q case, and for the SH-Q case it will simply be $\ell = t$. The inventory state space variable will be

$$k = \min\left\{ \left\lceil \frac{Q_t}{\kappa} \right\rceil, \nu^Q \right\} \mathbb{1}_{Q_t > 0} + \max\left\{ \left\lfloor \frac{Q_t}{\kappa} \right\rfloor, -\nu^Q \right\} \mathbb{1}_{Q_t < 0}, \tag{63}$$

for both settings, where $\kappa \in \mathbb{N}_+$ is the size of the bins. Another way of framing it is:

$$k = \begin{cases} -\nu^Q, & \text{if } Q_t < (-\nu^Q + 1)\kappa \\ -i, & \text{if } -i\kappa \leq Q_t < (-i+1)\kappa, 1 \leq i < \nu^Q \\ 0, & \text{if } Q_t = 0 \\ j, & \text{if } (j-1)\kappa < Q_t \leq j\kappa, 1 \leq j < \nu^Q \\ \nu^Q, & \text{if } Q_t > (\nu^Q - 1)\kappa \end{cases} \tag{64}$$

Finding a Suitable Inventory-Grouping

An empirical investigation of the running inventory of a market maker employing the fixed offset strategy with $\delta^{\pm} = 1, 2$ supports our choice of $\kappa = 5$ when $T = 1000$ and $\kappa = 2$ when $T = 25$. These choices were made to achieve an even distribution over the inventory groups. A further discussion of this topic can be found in Appendix B.3.

3.2.2.2 *Action Space*

In this section, the action space is defined as

$$\mathcal{A} = \mathcal{D} = \{1, \ldots, \bar{d}\}^2, \tag{65}$$

where $\mathcal{D}$ is the order depth action space. Above, $\bar{d} = 5$ and is the upper limit of the order depth of the market maker. If, as described and illustrated in the previous section, the sum of the order depths is strictly less than the prevailing bid-ask spread, the actual depths are updated accordingly, whereas the action remains the same; this is because we find it desirable to maintain a constant action space, mostly to reduce the time complexity of the learning.

It should be noted that the market order is not a part of the action space for the Q-learning, which is why the section in Algorithm 2 concerning market orders may be overlooked in the experiment with Q-learning and the MC LOB model.

3.2.2.3 *Reward*

The reward used to train the market maker agent is almost the same as for the SPM,

$$R_t = \Delta V_t, \tag{66}$$

where

$$V_t = \begin{cases} X_t + H_t, & 0 \leq t < T \\ X_{T^-} + p^L(\mathcal{L}(T^-), Q_{T^-})Q_{T^-}, & t = T \end{cases} \tag{67}$$

where $p^L(\mathcal{L}(T^-), Q_{T^-})$ is the volume-weighted average liquidation price for the quantity Q_{T^-} in the LOB $\mathcal{L}(T^-)$. It should be noted that the holding value H_t is defined as the liquidation value of the market maker's inventory at time t, somewhat different from the SPM.

The difference is that the running inventory penalty is removed. This is because of two main reasons: (1) the running inventory penalty was used for the SPM to make it comparable with the analytical strategies, and (2) removing the penalty makes the total episodes reward equal to the total PnL, a more interpretable reward.

3.2.2.4 *Results*

In this section, the results from the two settings of the Markov chain LOB model are presented and analyzed.

Recall that SH-Q refers to the model where the time state space variable is $\ell = t$ (see the discussion in Section 3.2.2) and that LH-Q refers to the model where the time state space variable is $\ell = \lceil \frac{t}{T}\eta^T \rceil + \mathbb{1}_{t=0}$. Additionally, the model parameters are as in Table 3.7. Table 3.8 illustrates the training parameter schemes used for the two models in this section. Note that exploring starts is not used here; this is because we are not interested in learning the depths for any other inventory levels at $t = 0$ than $Q_t = 0$. This differs from the SPM where we were interested in the depths for all states since we wanted to compare with the depths of the analytical strategies.

Parameter	**SH-Q**	**LH-Q**
κ	2	5
T	25	$1,000$
dt	5	1
γ	1	1
`default_order_size`	25	5
`n_episodes`	$1,600,000$	$40,000$
`n_actions_taken`	$8,000,000$	$40,000,000$
`n_environment_seconds`	$40,000,000$	$40,000,000$

Table 3.7: Model parameters for the SH-Q and the LH-Q setting.

	Starting rate	**Cut-off**	**Ending rate**	**Decreasing type**
ϵ-greedy	1	50%	0.05	Linear
Learning rate	0.5	100%	10^{-5}	Exponential

Table 3.8: The schemes of the training parameters for the SH-Q and the LH-Q setting. *Cut-off* shows how far into the training the final rate is reached.

Short-Horizon setting for Q-learning

Recall that SH-Q is the setting where we let the time state space variable be the time itself and let $dt = 5$ and $T = 25$, leading to five instances where the agent can make an action. For the market maker to be able to trade the same average volume per second, the default order size has been scaled by a factor of 5 (i.e., the market maker's order size is 25).

Some motivation for the rather small T might be due. First and foremost, we are keen to keep the state space small. At the same time, we want the agent to submit new orders with a relatively high frequency. Furthermore, we did not want to group the time marks here (we do this in the next section), but instead aimed to let the time state space variable represent the elapsed time. To appease all three goals, we arrived at letting $dt = 5$ and $T = 25$. Also, it should be noted that in the (original/analytical) SPM, $T = 30$.

The Q-learning was run eight times à 1,600,000 episodes. The results of the eight separate runs are displayed in Figure 3.10 and Table 3.9. We notice that the spread of the average rewards between the runs is small in absolute terms, but there is a relative difference of 10 between the largest and smallest average reward. An exploration of the stability of the runs can be found in Appendix B.4.

Run	$\overline{r}$	σ_r	$\overline{r}$ per action	$\overline{r}$ per second	$v_*(0,0)$	$a^*(0,0)$
1	6.78×10^{-5}	8.553×10^{-3}	1.36×10^{-5}	2.71×10^{-6}	4.973×10^{-6}	(4,4)
2	1.07×10^{-4}	8.195×10^{-3}	2.14×10^{-6}	4.28×10^{-6}	-3.817×10^{-5}	(4,5)
3	4.18×10^{-5}	$\mathbf{3.744 \times 10^{-3}}$	8.36×10^{-6}	1.67×10^{-6}	-2.352×10^{-5}	(5,5)
4	$\mathbf{1.54 \times 10^{-4}}$	7.482×10^{-3}	$\mathbf{3.08 \times 10^{-5}}$	$\mathbf{6.17 \times 10^{-6}}$	$\mathbf{1.331 \times 10^{-5}}$	(4,5)
5	8.38×10^{-5}	5.988×10^{-3}	1.68×10^{-6}	3.35×10^{-6}	-7.148×10^{-6}	(5,5)
6	5.44×10^{-5}	7.766×10^{-3}	1.09×10^{-5}	2.18×10^{-6}	-2.057×10^{-5}	(5,4)
7	5.66×10^{-5}	6.359×10^{-3}	1.13×10^{-5}	2.64×10^{-6}	-8.281×10^{-6}	(5,5)
8	-4.22×10^{-5}	8.971×10^{-3}	-8.44×10^{-6}	-1.69×10^{-6}	-2.364×10^{-5}	(5,5)

Table 3.9: The average reward received when following the obtained strategies from Q-learning in the SH-Q setting for 5×10^4 episodes. Bold values highlight the best result: the highest reward or the lowest standard deviation.

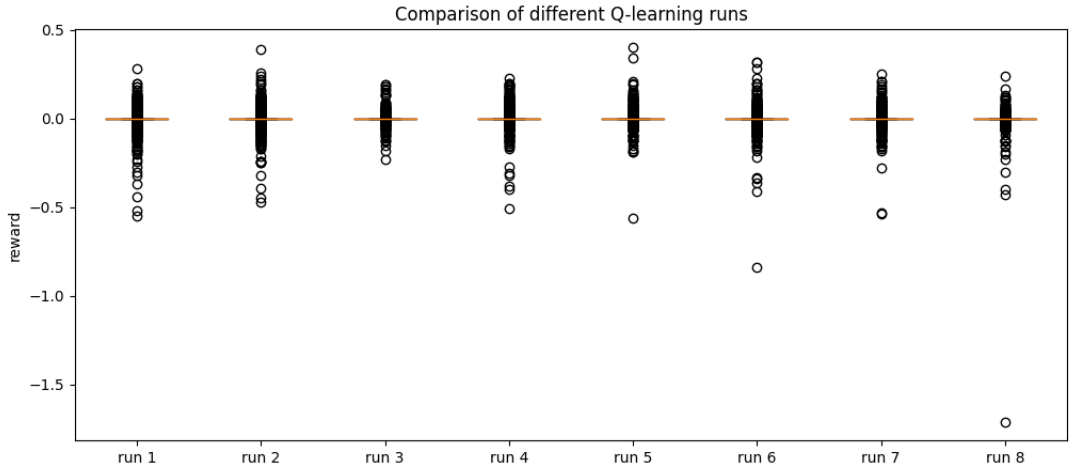


Figure 3.10: Boxplot of the reward of eight different Q-learning runs in the SH-Q setting when evaluated for 5×10^4 episodes.

Figure 3.11 plots the average reward and the state-value at $(t, Q_t) = (0, 0)$. The value of $Q(0,0)$ converges towards the average reward of an entire episode, indicating that the Q-learning has converged, which is desirable as this is the algorithm's goal. However, the state values in Table 3.9 does not agree precisely with the average rewards, indicating that the algorithm has not fully converged.

The strategies are displayed in Figure 3.12. However, these strategies do not look particularly much like we expected. For example, we expected a gradient such that the bid depth for a large positive inventory is the largest and gradually decreases towards the minimum depths for the largest negative inventory (and the reverse for the ask side). It is pretty hard to argue that a apparent gradient is visible in the heat maps. The strategy is to place bids and asks at quite a large depth, the clear exception being at $t = 4$ when the optimal depth is significantly smaller.

A potential explanation for these strategies is that by placing the orders at a large depth, the agent will on average not trade, but when a trade occurs, the immediate gain relative to the mid price will be larger. However, since the overall time horizon considered here is relatively short

(only 25 seconds), the risk of building a net inventory that will need to be liquidated in a very short period may be costly.

Very similar strategies were obtained for a large array of different combinations of parameters. Some other potential reasons for the lack of a gradient are discussed next.

First and foremost, the fact that $dt = 5$ increases the likelihood that the market maker gets to trade during a time step – even with a larger order depth. This may explain why the optimal depth almost consistently is large. There are issues with this. For one, the market maker barely makes a profit on average (the average reward of the mean strategy is 2.5×10^{-5}). Additionally, if the market maker's order volume on either side is bought or sold, the market maker will not come in with new quotes until after the dt seconds have passed, which may mean that the market maker is without quotes on one or two sides for an undesirably long time (see Primer 4). While we tried to combat this by scaling the default order size, this can instead cause other problems.

Secondly, the minimal state space in this model may contain far too little information for the market maker to infer optimal actions. Moreover, the current state of the LOB has a large impact on the arrival of future events, and our agent can observe none of this information. As the reader may remember, in the SPM, the arrival rates of orders matched against the market maker's only depended on the order depth of the market maker's orders, in quite a stark contrast to this setting. This may seriously impede the learning of solid strategies.

Additionally, the short time horizon implies that the market maker will not be able to trade particularly much and will only maintain relatively small positions in absolute terms. Large absolute inventories may thus never inflict substantial losses, and the agent will not learn to be risk-averse. This may be an important factor in why the agent does not learn a gradient strategy in this setting.

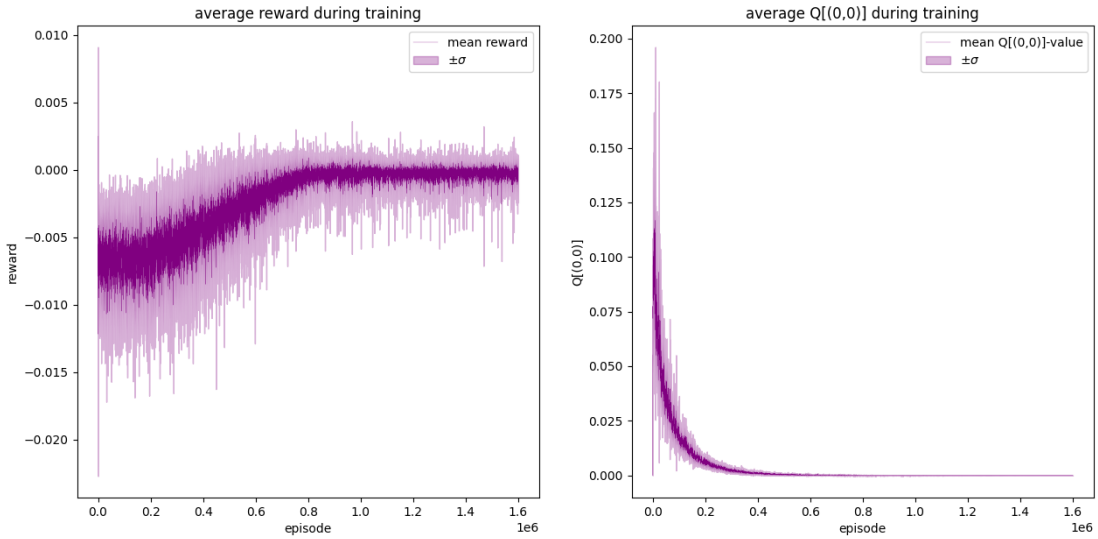
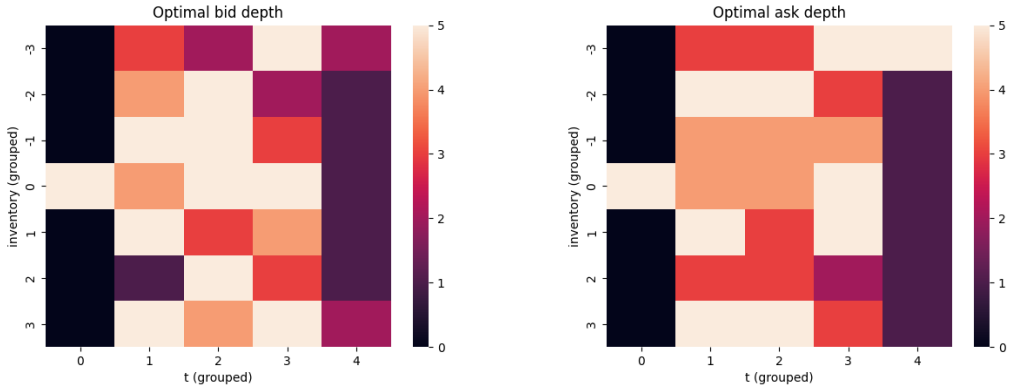


Figure 3.11:
Training in the SH-Q setting for 1.6×10^6 episodes.
Left: the average reward during training with a shaded area showing ± one standard deviation.
Right: the state-value at $(t, Q_t) = (0, 0)$ during training with a shaded area showing ± one standard deviation.



(a) (b)

Figure 3.12: These two panels are obtained from the Q-matrix obtained by averaging the Q-matrices of all eight runs in the SH-Q setting. **(a)** shows a heat map of the optimal bid depths, and **(b)** shows a heat map of the optimal ask depths.
As the reader clearly can tell, the "optimal" strategy at $t = 0$, $Q_t \neq 0$, is the zero-depth. This is not the case – the heat map looks like this since the market maker will always start with inventory zero at time zero. All states where the displayed optimal depth is zero are states that are thus unvisited.

A comparison between the best run and the mean Q-learning strategy against the benchmarking strategies can be seen in Figure 3.13 and Table 3.10. The Q-learning strategies outperform the benchmarks, both the best run and the mean strategy. Furthermore, the standard deviation of the reward is an order of magnitude smaller than the reward. Notably, the mean strategy has a very small standard deviation.

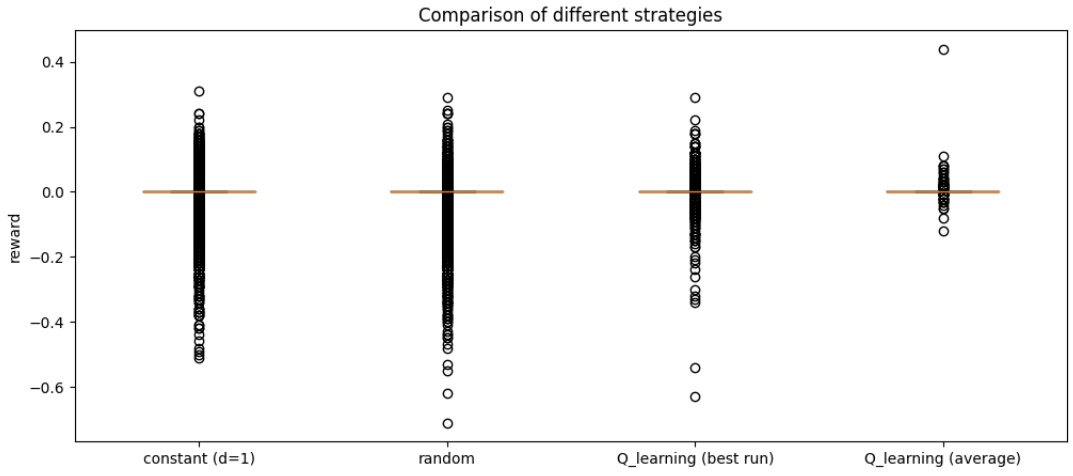


Figure 3.13: Boxplot of the reward of different strategies in the SH-Q setting when evaluated for 5×10^4 episodes.

The resulting inventory, cash, and value processes of this strategy are illustrated in Figure 3.14. However, as is clear, the market maker does typically not get to trade particularly much (which is due to the relatively short time horizon considered in this setting).

Strategy	$\overline{r}$	σ_r
Q-learning		
- *best run*	$\mathbf{1.5 \times 10^{-4}}$	7.482×10^{-3}
- *mean strategy*	2.5×10^{-5}	$\mathbf{2.433 \times 10^{-3}}$
Benchmark		
- *constant*	-1.3×10^{-3}	2.733×10^{-2}
- *random*	-5.3×10^{-3}	2.890×10^{-2}

Table 3.10: The average reward received when following different strategies in the SH-Q setting for 5×10^4 episodes. Bold values highlight the best result: the highest reward or the lowest standard deviation.

It should be noted that this model setup does not result in a fully observable observation space (information about the LOB and some information regarding the exact inventory is lost). Therefore, Q-learning is not guaranteed to converge in the limit. Nonetheless, Figure 3.11 indicates that the Q-learning has converged, and the average rewards are tiny, suggesting that a more robust method is needed to find better strategies. However, first, we will look at a different way of framing the problem, the LH-Q setting.

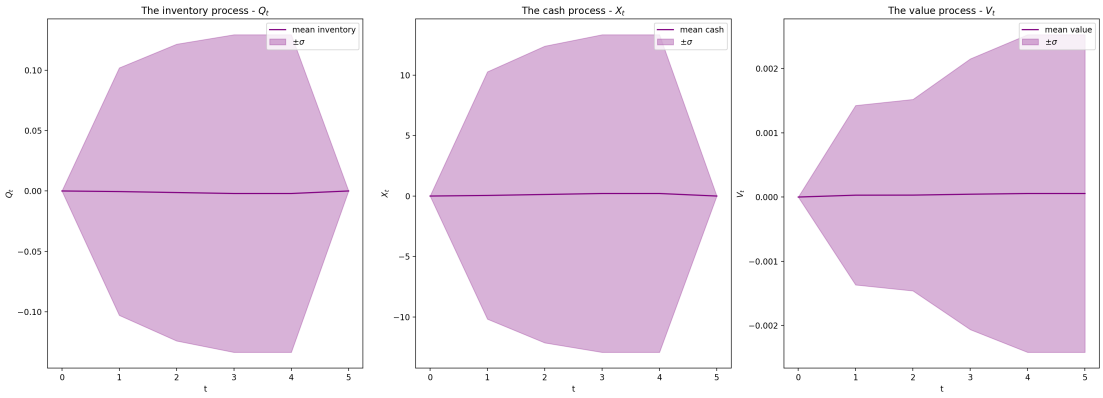


Figure 3.14: The average of the inventory, cash process, and value processes when following the mean strategy obtained from training in the SH-Q setting. Also showing $\pm$ one standard deviation in the shaded area. Evaluated for 5×10^4 episodes.

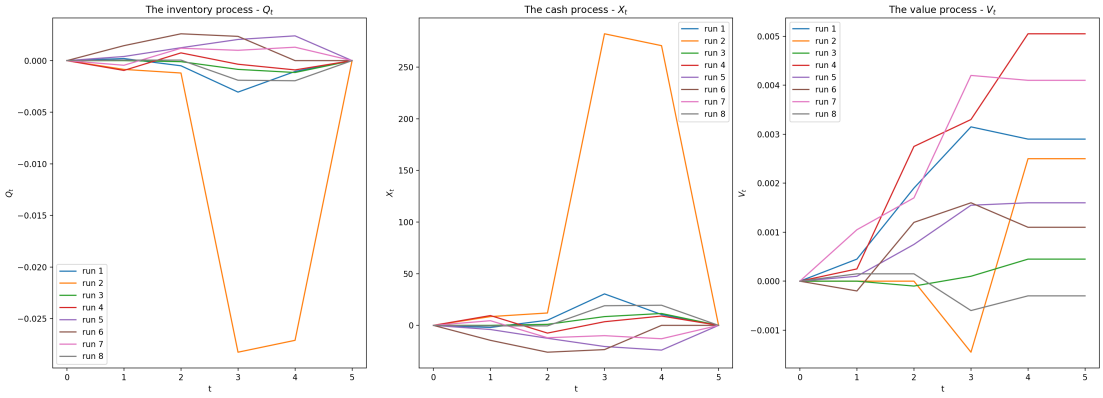


Figure 3.15: The average of the inventory, cash process, and value processes for all eight individual runs obtained from training in the SH-Q setting. Evaluated for 5×10^4 episodes.

Long-Horizon setting for Q-learning

We continue our experiments with the LH-Q setting by binning the time as described previously. As clearly illustrated above, Q-learning in the SH-Q setting failed to find intuitive and financially savvy strategies. However, the problems could (probably) have been remedied by decreasing dt and increasing the number of time steps in the state space, but this would have added undesirable computational complexity.

In this model, we let $T = 1,000$ and $dt = 1$. We consider $T = 1,000$ to be a sufficiently long time horizon for the market maker to do some actual trading, with results that hopefully may generalize into a longer session (say, an entire trading day less opening and closing auctions). Also, $dt = 1$ represents a granular enough, but not overly granular, time resolution for the market maker.

The Q-learning in this setting was run eight separate times for 40,000 episodes each, which is the same number of seconds the environment was run for during the training in the SH-Q setting. A comparison of the eight different runs can be seen in the boxplots in Figure 3.16 and in Table 3.11. However, we immediately notice that the average rewards and the standard deviations of the runs differ significantly, indicating that the Q-learning has not converged. This can also be seen in Figure 3.17 where the shaded area representing ± one standard deviation is very large and does not decrease in size during training.

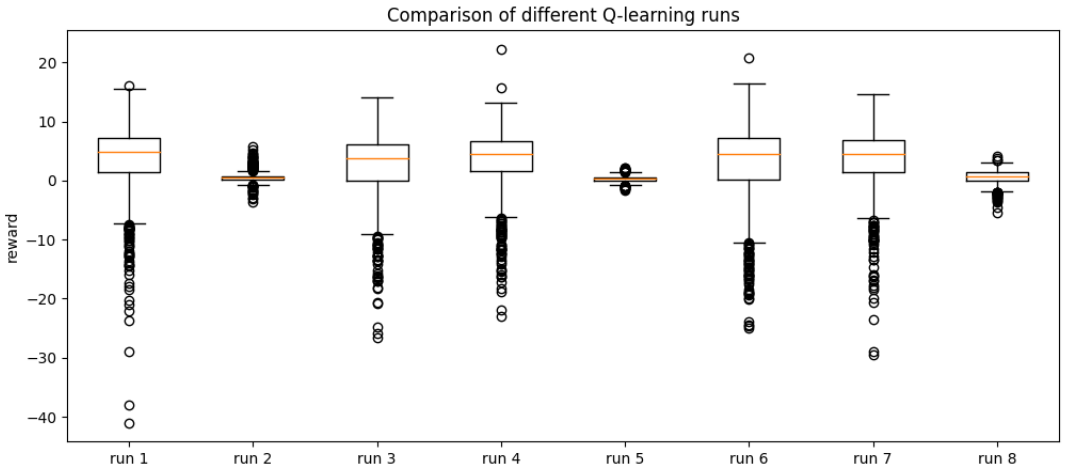


Figure 3.16: Boxplot of the reward of eight different Q-learning runs on the LH-Q setting when evaluated for 1,000 episodes.

When investigating Figure 3.17 further, we also notice that the average reward and state-value for the starting state do not converge to the same value – indicating that the algorithm has not converged. Further evidence of non-convergence can be found in Appendix B.5, where the stability of the strategies is explored.

It has been challenging to provide a convincing explanation of this phenomenon – $Q(0,0)$ increasing indefinitely – but this is our current proposal.

Looking at the update-equation of Q-learning,

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha\Big[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)\Big], \tag{68}$$

given $\gamma = 1$ and the time grouping, it may be reduced to the following

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha R_{t+1}, \tag{69}$$

if $S_{t+1} = S_t$ (which can happen in the time-grouped model) and the optimal action remains the same. If this happens frequently, the state-value will increase indefinitely if the rewards are positive. The normal interpretation of the state-value disappears – the state-value is no longer the expected reward for the rest of the episode, given the current state and an action.

Run	$\overline{r}$	σ_r	$\overline{r}$ **per action**	$\overline{r}$ **per second**	$v_*(0,1)$	$a^*(0,0)$
1	**3.648**	5.954	$\mathbf{3.648 \times 10^{-3}}$	$\mathbf{3.648 \times 10^{-3}}$	2.910×10^2	(2,5)
2	5.343×10^{-1}	7.880×10^{-1}	5.343×10^{-4}	5.343×10^{-4}	3.167×10^2	(2,5)
3	2.405	5.574	2.405×10^{-3}	2.405×10^{-3}	2.935×10^2	(3,2)
4	3.473	5.160	3.473×10^{-3}	3.473×10^{-3}	3.093×10^2	(2,5)
5	3.216×10^{-1}	$\mathbf{4.916 \times 10^{-1}}$	3.216×10^{-4}	3.216×10^{-4}	3.108×10^2	(1,4)
6	2.935	6.510	2.935×10^{-3}	2.935×10^{-3}	2.954×10^2	(1,5)
7	3.473	5.309	3.473×10^{-3}	3.473×10^{-3}	2.972×10^2	(5,2)
8	6.552×10^{-1}	1.087	6.552×10^{-4}	6.552×10^{-4}	$\mathbf{3.189 \times 10^2}$	(1,5)

Table 3.11: The average reward received when following the obtained strategies from Q-learning in the LH-Q setting for $1,000$ episodes. Bold values highlight the best result: the highest reward or the lowest standard deviation.

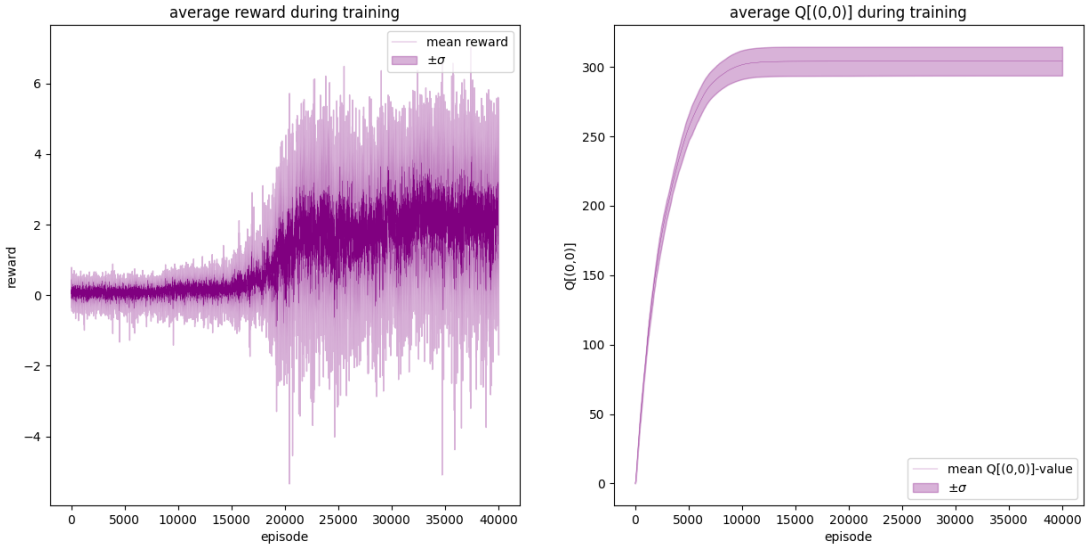


Figure 3.17:
Training in the LH-Q setting for 4×10^4 episodes.
Left: the average reward during training with a shaded area showing ± one standard deviation.
Right: the state-value at $(t, Q_t) = (0, 0)$ during training with a shaded area showing ± one standard deviation.

An important remark is that the Markov property is broken even further in this setting (recall that the full LOB has to be included in the state). To illustrate this, consider the same parameters

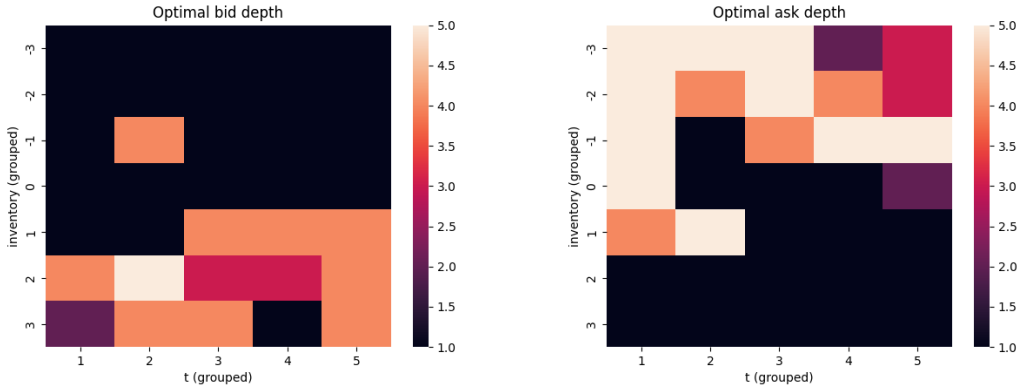
as specified above, $\eta^T = 5$ and $T = 1,000$. Further, consider the state process $\boldsymbol{S}_t$, $(\boldsymbol{S}_t)_{t=0}^T$, which takes the value $\boldsymbol{s}_t = (\ell_t, k_t)$. Next, fix $\boldsymbol{s}_{t-2} = (2, k_{t-2})$ and $\boldsymbol{s}_{t-1} = (3, k_{t-1})$. Then it is true that (ignoring any influence of the agent's actions),

$$\mathbb{P}(\boldsymbol{S}_t = (\ell_t, k_t) \mid \boldsymbol{S}_{t-1} = \boldsymbol{s}_{t-1}) \neq \mathbb{P}(\boldsymbol{S}_t = (\ell_t, k_t) \mid \boldsymbol{S}_{t-1} = \boldsymbol{s}_{t-1}, \boldsymbol{S}_{t-2} = \boldsymbol{s}_{t-2}) \tag{70}$$

since on the right-hand side, the conditioning implies that $\ell_t = 3$ a.s. (recall how the time state space variable is calculated), which is not known immediately from the conditioning on the left-hand side, implying that the two probabilities are not necessarily equal. Thus, the state transition does not satisfy the Markov property, and the process is not a Markov decision process. As previously discussed, this implies that there are no guarantees on the convergence of Q-learning. A similar argument could be used to argue that the process is a *partially observable* Markov decision process (partially observable MDPs are of a non-Markovian nature [50]). Liu et al. note that partially observable RL problems can be notoriously difficult to solve [50].

The analog argument does not entirely hold regarding the binning of inventory since the inventory may move in either direction, and conditioning on additional past states does not yield significantly more useful information than merely conditioning on the current state. However, the process is still only partially observable, but the unobservable information (the exact inventory) is probably not as important as the time, which most likely can explain why the Q-learning succeeded in the SH-Q setting but not in the LH-Q setting.

The mean strategy provides financially intuitive strategies despite the unstable strategies and the seemingly non-convergent algorithm. Figure 3.18 displays these strategies, and one can see the desired gradient of the depth when $q : \underline{q} \to \overline{q}$. However, since we have no analytically optimal strategies for the MC model, we must look at benchmarks to judge the results.



(a) (b)

Figure 3.18: These two panels are obtained from the Q-matrix obtained by averaging the Q-matrices of all eight runs in the LH-Q setting. **(a)** shows a heat map of the optimal bid depths, and **(b)** shows a heat map of the optimal ask depths.

Looking at the boxplot in Figure 3.19, and the average rewards and standard deviations in Table 3.12, one notices that the best Q-learning run outperforms all other strategies. However, its standard deviation is very high, which is undesirable for real-world market makers. The mean Q-learning strategy is also the worst-performing one, with a negative average reward. This

is further proof that the Q-learning runs have not converged and are sensitive to the random starting seed used for the environment.

Strategy	$\overline{r}$	σ_r
Q-learning		
- *best run*	**3.648**	5.954
- *mean strategy*	-3.678×10^{-1}	$\mathbf{3.729 \times 10^{-1}}$
Benchmark		
- *constant*	3.355×10^{-1}	7.101×10^{-1}
- *random*	4.852×10^{-2}	7.067×10^{-1}

Table 3.12: The average reward received when following different strategies in the LH-Q for 1,000 episodes. Bold values highlight the best result: the highest reward or the lowest standard deviation.

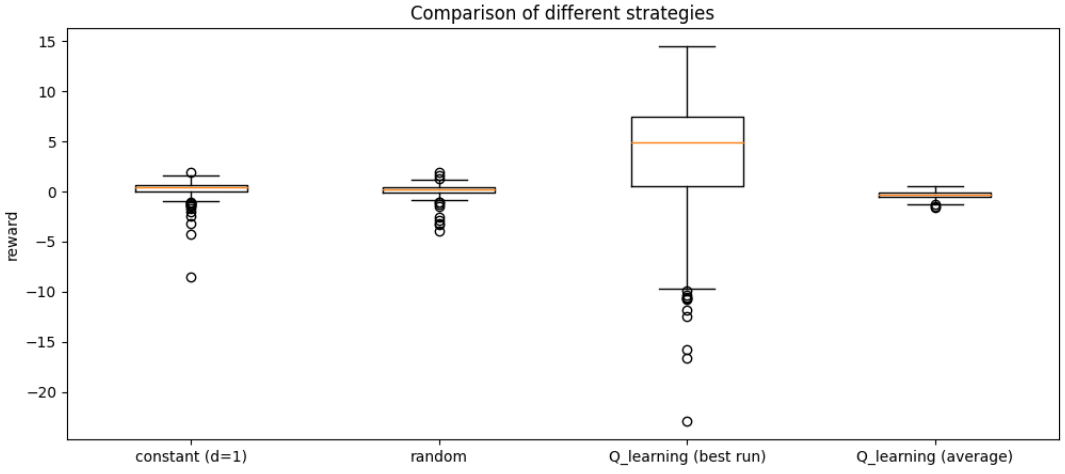


Figure 3.19: Boxplot of the reward of different strategies in the LH-Q setting when evaluated for 1,000 episodes.

An additional interesting remark is that from looking at the strategies in Figure 3.18, we see that the bids are generally at a smaller depth than the asks (when on positive inventory versus negative inventory, respectively), meaning that the market maker will tend to hold more positive inventory than negative. This is clearly illustrated in Figure 3.20, where for the first time steps, the market maker tends to build a positive inventory, which is sold off after around $t = 400$. This strategy is inferior, demonstrated in addition to the numbers in Table 3.12, on the right in Figure 3.18 where the value process is plotted. How the market maker skews the prices is suboptimal; as is visible, the value consistently declines on average. When studying the eight individual strategies, we see that the market maker can still make money despite clear skewing and inventory building on average. Evidently, the averaging needed for the mean strategy results in a poor market making strategy. A possible explanation for this is that the different runs converged to different local optima – the average of which was not an optimum. It is noteworthy that while six out of eight runs skew towards a positive inventory on average (see the optimal actions in Table 3.11), this outcome is not particularly unlikely, given an equal chance of skewness in both

directions. Furthermore, it is also the strategies that on average skew the most that earn the highest reward; however, this is somewhat countered by the strategies' higher standard deviations.

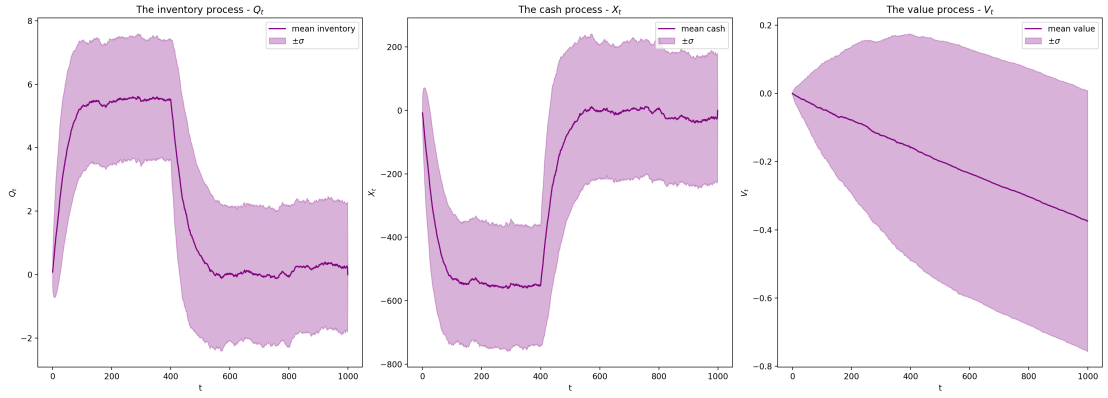


Figure 3.20: The average of the inventory, cash process, and value processes when following the mean strategy obtained from training in the LH-Q setting. Also showing $\pm$ one standard deviation in the shaded area. Evaluated for $1,000$ episodes.

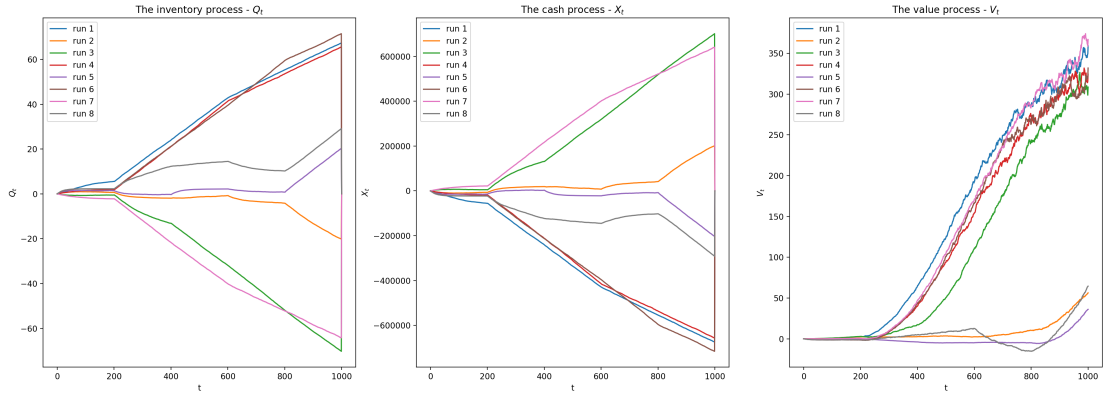


Figure 3.21: The average of the inventory, cash process, and value processes for all eight individual runs obtained from training in the LH-Q setting. Evaluated for $1,000$ episodes.

Figure 3.21 also shows some limitations of the time-grouping used in the LH-Q setting. Firstly, almost no value is created during the first time step (0-200 seconds). Secondly, the value creation in the final step (801-1,000 seconds) is volatile. This is most likely due to the market maker not being able to distinguish how close it is to the forced liquidation at $T = 1,000$ during the final step, forcing it to adopt a more risk-averse strategy. Finally, in the left panel, the inventory process reaches absolute values of higher than ten during most of the episodes, meaning that most of the time is spent in the inventory groups farthest away from 0.

3.2.3 *Deep Reinforcement Learning*

This section presents the results from using deep reinforcement learning for market making and seeks to answer our second research question (i.e., if deep RL can outperform Q-learning in market making).

To train the agent with deep reinforcement learning methods, we use the open-source `PFRL` package in Python, which is a deep reinforcement learning library using `PyTorch`. `PFRL` was developed by [23]. To speed up training, the environment was parallelized using the `PFRL` class `MultiprocessVectorEnv`. The training was done on the Intel(R) Core(TM) i7-11700K processor.

We continue to use our two model setups, the short- and long-horizon. However, we here name them the short-horizon for DDQN (SH-DDQN) and the long-horizon for DDQN (LH-DDQN).

3.2.3.1 *State Space*

The state space $\mathcal{S}$ in this section is defined (the same for both settings) such that the state at time t is

$$\boldsymbol{s}_t \propto (t, Q_t, s(t), q^{a(t)-\delta}, \dots, q^{a(t)-d\delta}, q^{b(t)+\delta}, \dots, q^{b(t)+d\delta}) \in \mathbb{R}^{3+2d}, \tag{71}$$

where the notation has been somewhat abused. The first two elements of the state, t and Q_t, represent the time and the current inventory, while the remaining elements represent the LOB. As previously, $d = 10$, meaning that $\dim \mathcal{S} = 23$. Again, we have that $\delta = 1$.

Above, $\propto$ is used to highlight that all state space variables are normalized to (approximately) keep the state in $[-1, 1]^{3+2d}$. All state space variables have not been scaled equally. Scaling can have a large effect on the quality of an artificial neural network and may enable an ANN to *"run more efficiently and may help to avoid getting bogged down in local extrema"* [43].

Note that the state space constructed in Equation (71) makes the process a fully observable Markov decision process in contrast to the state spaces in the SH-Q and LH-Q settings.

3.2.3.2 *Action Space*

The action space in this section, $\mathcal{A}$, is defined as

$$\mathcal{A} = \mathcal{D} = \{1, \dots, \bar{d}\} \times \{1, \dots, \bar{d}\}, \tag{72}$$

where $\mathcal{D}$ is the order depth action space. Above, $\bar{d} = 5$ and is the upper limit of the order depth of the market maker. It should be noted that if, as described and illustrated in the previous section, the sum of the order depths is strictly less than the prevailing bid-ask spread, the actual depths are updated accordingly, whereas the action remains the same. This is because we find it desirable to maintain a constant action space, mostly to reduce the time complexity of the learning.

3.2.3.3 *Network Architecture*

Two different network architectures were used for the two DDQNs.

In the SH-DDQN setting, a simple network consisting of three linear layers was used. It is specified in Table 3.13. To aid convergence in the LH-DDQN setting, a larger network equipped with batch normalization was adopted. The full network is specified in Table 3.14. Note that this larger architecture did not improve performance in the SH-DDQN setting and was thus only used for the LH-DDQN setting.

Parameter	SH-DDQN
Linear layer 1	In: 23, Out: 64
Activation function	ReLu
Linear layer 2	In: 64, Out: 64
Activation function	ReLu
Linear layer 3	In: 64, Out: 25

Table 3.13: The neural network architecture used for the SH-DDQN.

Parameter	LH-DDQN
Linear layer 1	In: 23, Out: 64
Activation function	ReLu
Linear layer 2	In: 64, Out: 64
Batch normalization	Yes
Activation function	ReLu
Linear layer 3	In: 64, Out: 64
Activation function	ReLu
Linear layer 4	In: 64, Out: 64
Batch normalization	Yes
Activation function	ReLu
Linear layer 5	In: 64, Out: 25

Table 3.14: The neural network architecture used for the LH-DDQN.

In Table 3.15 the hyperparameters of the DDQN algorithm are specified for the two settings. The variable `steps` represents the number of time steps (dt) the agents are trained for. The number of steps for the SH-DDQN setting was chosen to equal the same amount of training in terms of episodes as for the SH-Q setting. However, the number of steps for the LH-DDQN setting was chosen to equal the amount of training time used for the SH-DDQN setting, that is ~ 24 hours. In both settings, ten environments were parallelized.

Parameter	SH-DDQN	LH-DDQN
`buffer_size`	40,000	50,000
`replay_start_size`	40,000	50,000
`target_update_interval`	4,000	5,000
`update_interval`	4	4
`steps`	8,000,000	10,000,000
`num_episodes`	1,600,000	10,000
`num_environments`	10	10
`reward_scale`	100	0.01
γ	1	1
`learning_rate`	3×10^{-7}	7×10^{-5}
`batch_size`	64	64
`optimizer`	Adam	Adam
`exploration_rate`	$1 \rightarrow 0.05$ during first 50%	$1 \rightarrow 0.05$ during first 50%

Table 3.15: Hyperparameters used for training in the SH- and LH-DDQN setting.

3.2.3.4 *Rewards*

The reward used to train the market maker agent is identical to the one used for SH-Q and LH-Q,

$$R_t = \Delta V_t, \tag{73}$$

where

$$V_t = \begin{cases} X_t + H_t, & 0 \leq t < T \\ X_{T^-} + p^L(\mathcal{L}(T^-), Q_{T^-})Q_{T^-}, & t = T. \end{cases} \tag{74}$$

It should be noted that the holding value H_t is defined as the liquidation value of the market maker's inventory at time t.

Reward scale

As per the guidelines of Henderson et al. [35], a scaling of the reward was used to improve convergence. To clip the total expected reward over an episode close to $[0, 1]$, a reward scale of 100 was used in the SH-DDQN setting and 0.01 in the LH-DDQN setting. This is also specified in Table 3.15.

3.2.3.5 *Results*

This section will present the results of training in the SH-DDQN and the LH-DDQN.

SH-DDQN and Neutral LOB

In our first experiment with deep reinforcement learning, we use essentially the same model parameters as in the SH-Q setting. That is, we let the total time horizon be $T = 25$ and $dt = 5$. Additionally, the order size is yet again 25. Compared to the Q-learning in the SH-Q setting, we do not group the inventory (as is evident from the state space specified in Equation (71) above). Additionally, in this experiment, we let the initial state of the LOB during an episode be "neutral" during training. By a neutral LOB, we mean the LOB where

$$(a(t), q^{b(t)+\delta}, \ldots, q^{b(t)+d\delta}) = (10001, 0, 1, \ldots, 1) \tag{75}$$

$$(b(t) - a(t), q^{a(t)-\delta}, \ldots, q^{a(t)-d\delta}) = (-2, 0, -1, \ldots, -1) \tag{76}$$

Illustrated graphically, this is shown in Figure 3.22.

The results of this experiment will not be discussed in particular detail. Instead, we recommend the reader interested in the details to study the results in Sections B.6 - B.12 in the appendix. The main results from this setting that we want to highlight to the reader are the average rewards during evaluation and the average loss during training and compare them with the results from when the initial LOB state is randomized both during training and evaluation. The average reward is displayed in Table B.1, and we may see that five out of eight average rewards for individual runs are negative and that the average reward of the mean strategy is 2.32×10^{-5}. Furthermore, we may in B.7 see that the average loss during training converges to twice the loss achieved when training using a randomized LOB.

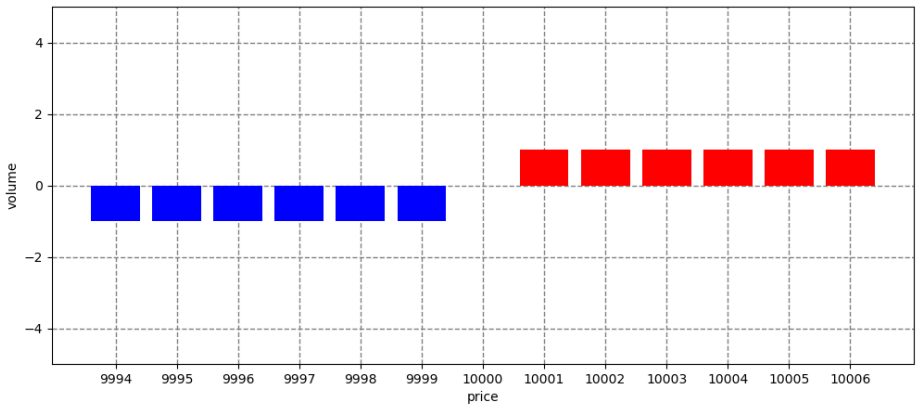


Figure 3.22: A "neutral" limit order book.

SH-DDQN and Randomized LOB

The overall setting is the same as in the previous experiment. The main differences are that (1) the initial LOB of an episode is randomized during the training phase, and (2) the initial LOB of an episode is randomized during evaluation. On the contrary, in the previous experiment, both LOBs were neutral. The LOB is randomized by drawing a random state from a bank of 100,000 states. The state-bank was generated by saving the current state every second when simulating an environment for 100,000 seconds.

The reason for including both experiments is that we want to highlight the large difference in the performance of the DDQN with and without the randomization.

The DDQN was run eight times à 8,000,000 time steps. The results of these runs are displayed in Figure 3.23 and Table 3.16. Among the eight individual runs, five mean rewards were positive in the randomized setting instead of only three in the neutral setting. While the negative rewards are undesirable, these strategies beat the Q-learning strategies on average – more on this in Section 3.3. An exploration of the stability of the runs can be found in Appendix B.13.

Run	$\overline{r}$	σ_r	$\overline{r}$ **per action**	$\overline{r}$ **per second**	$v_*(0,0)$
1	-3.62×10^{-5}	9.97×10^{-3}	-7.24×10^{-6}	-1.45×10^{-6}	3.71×10^{-5}
2	-1.80×10^{-6}	8.99×10^{-3}	-3.60×10^{-7}	-7.20×10^{-8}	5.88×10^{-5}
3	-1.80×10^{-4}	1.20×10^{-2}	-3.59×10^{-5}	-7.18×10^{-6}	7.07×10^{-5}
4	2.50×10^{-5}	$\mathbf{6.27 \times 10^{-3}}$	4.80×10^{-6}	9.60×10^{-7}	1.84×10^{-5}
5	1.54×10^{-4}	8.38×10^{-3}	3.07×10^{-5}	6.14×10^{-6}	5.68×10^{-5}
6	9.60×10^{-5}	7.09×10^{-3}	1.92×10^{-5}	3.84×10^{-6}	6.46×10^{-5}
7	$\mathbf{2.12 \times 10^{-4}}$	8.28×10^{-3}	$\mathbf{4.24 \times 10^{-5}}$	$\mathbf{8.47 \times 10^{-6}}$	$\mathbf{7.20 \times 10^{-5}}$
8	9.44×10^{-5}	6.65×10^{-3}	1.89×10^{-5}	3.78×10^{-6}	4.48×10^{-5}

Table 3.16: The average reward received when following the obtained strategies from DDQN in the SH-DDQN setting with randomized resets for 5×10^4 episodes. Bold values highlight the best result: the highest reward or the lowest standard deviation.

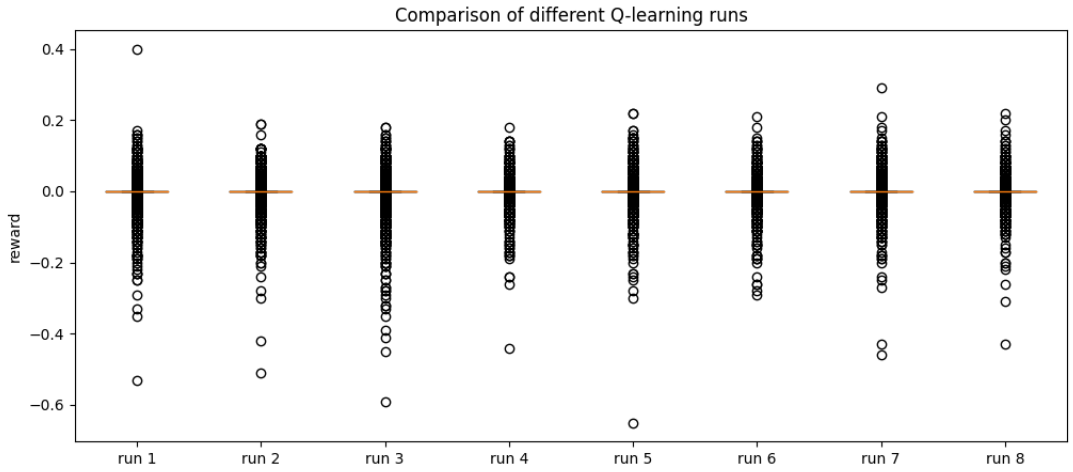


Figure 3.23: Boxplot of the reward of eight different DDQN runs in the SH-DDQN setting with randomized resets when evaluated for 5×10^4 episodes.

Figure 3.24 plots the average reward, the state-value at $(t, Q_t) = (0, 0)$, and the loss during training. The state value is estimated by calculating the average over 10,000 random states where $t = 0$ and $Q_t = 0$. $\hat{v}(0, 0)$ converges towards the average reward of an entire episode, indicating that the training has converged, which is desirable as this is the algorithm's goal.

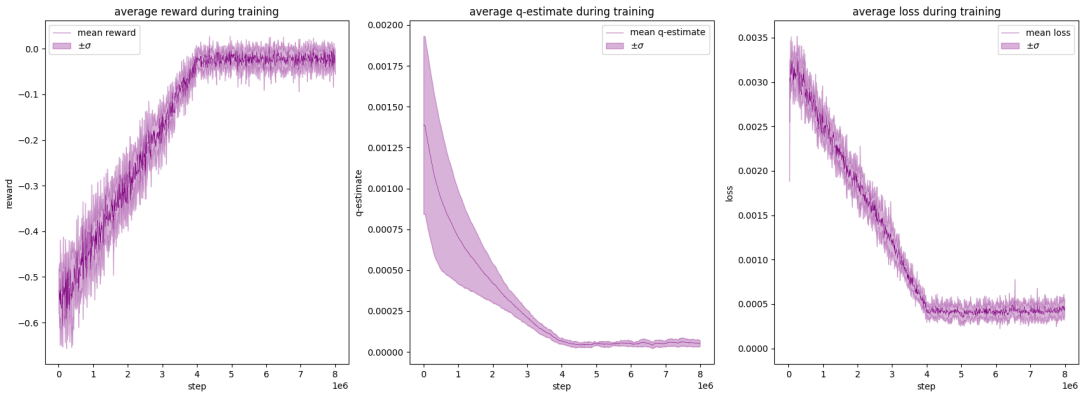
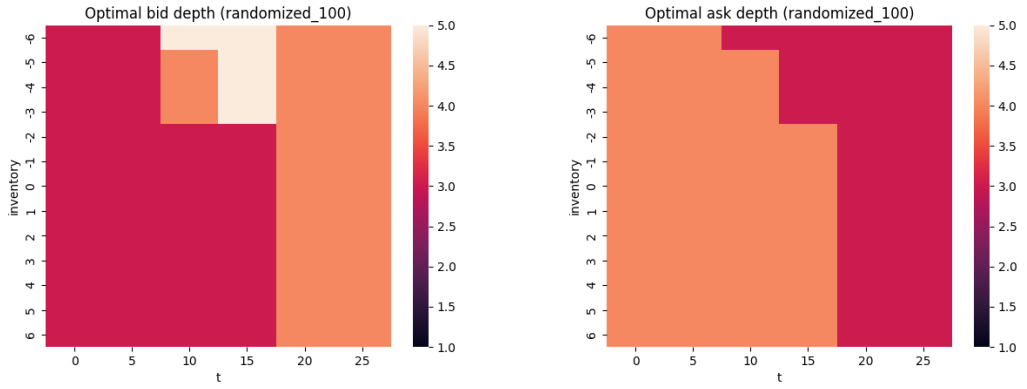


Figure 3.24:
Training in the SH-DDQN setting for 8×10^6 steps.
Left: the average reward during training with a shaded area showing ± one standard deviation.
Middle: the state-value at $(t, Q_t) = (0, 0)$ during training with a shaded area showing ± one standard deviation.
Right: the average loss during training with a shaded area showing ± one standard deviation.

The obtained mean strategy are displayed in Figure 3.25. Unintuitively, the strategy looks the opposite of what we expected; it suggests larger bid depths when the inventory is small and larger ask depths when the inventory is large. This indicates that the strategy on average will skew to large net positions, either long or short.



(a) (b)

Figure 3.25: These two panels are obtained by averaging the action-values over 100 randomized states from the eight agents trained in the SH-DDQN setting. **(a)** shows a heat map of the optimal bid depths, and **(b)** shows a heat map of the optimal ask depths.

The best run and the mean strategy are compared against the benchmarking strategies in Figure 3.26 and Table 3.17. Both the best run and the mean strategy outperforms the benchmarks. Furthermore, the mean strategy performs significantly better in the randomized setting compared to in the neutral one: 1.41×10^{-4} versus 2.32×10^{-5} in average rewards. On top of that, the standard deviation of the mean strategy is about half for the randomized setting.

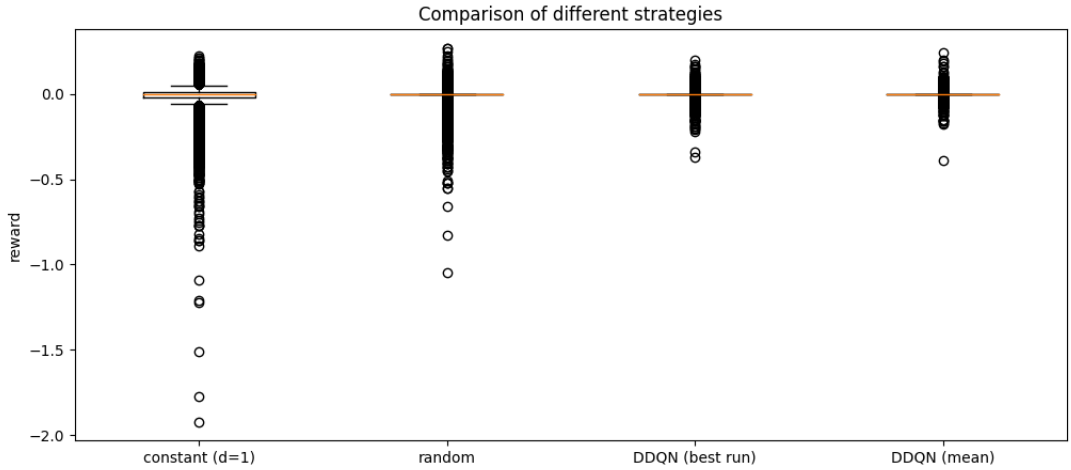


Figure 3.26: Boxplot of the reward of different strategies in the SH-DDQN setting when evaluated for 5×10^4 episodes.

Strategy	$\overline{r}$	σ_r
DDQN		
- *best run*	$\mathbf{2.12 \times 10^{-4}}$	8.28×10^{-3}
- *mean strategy*	1.41×10^{-4}	$\mathbf{5.61 \times 10^{-3}}$
Benchmark		
- *constant*	-1.19×10^{-2}	5.58×10^{-2}
- *random*	-5.38×10^{-3}	2.98×10^{-2}

Table 3.17: The average reward received when following different strategies in the SH-DDQN for 5×10^4 episodes. Bold values highlight the best result: the highest reward or the lowest standard deviation.

The resulting inventory, cash, and value processes of the mean strategy and the individual runs are illustrated in Figure 3.27 and Figure 3.28 respectively. Figure 3.28 illustrates that no particular skewing direction is beneficial for higher rewards, but most of the runs choose a short position.

In order to provide a more intuitive understanding of the strategies and how they change depending on the current LOB, an example is shown in Figure 3.29. The two LOB states used for this are shown in Figure 3.30 – a "balanced" and a "buy-heavy" LOB. In this figure, the agent increases its bid depths and decreases its ask depths, indicating an increased desire to decrease its inventory. The strategies for the two remaining scenarios, "sell-heavy" and "large spread," are shown in Appendix B.14 and B.15 respectively. For a more quantifiable analysis of the different states, see Table 3.18 for some financial measures.

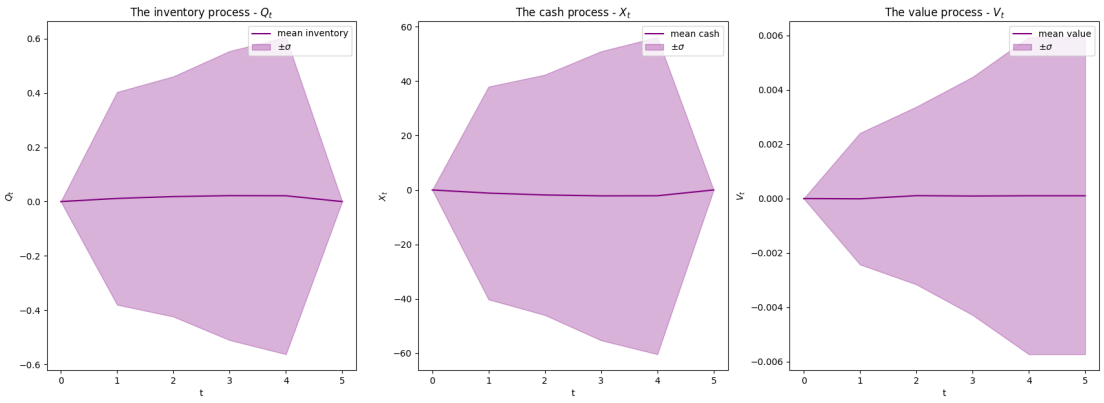


Figure 3.27: The average of the inventory, cash process, and value processes when following the mean strategy obtained from training in the SH-DDQN setting. Also showing ± one standard deviation in the shaded area. Evaluated for 5×10^3 episodes.

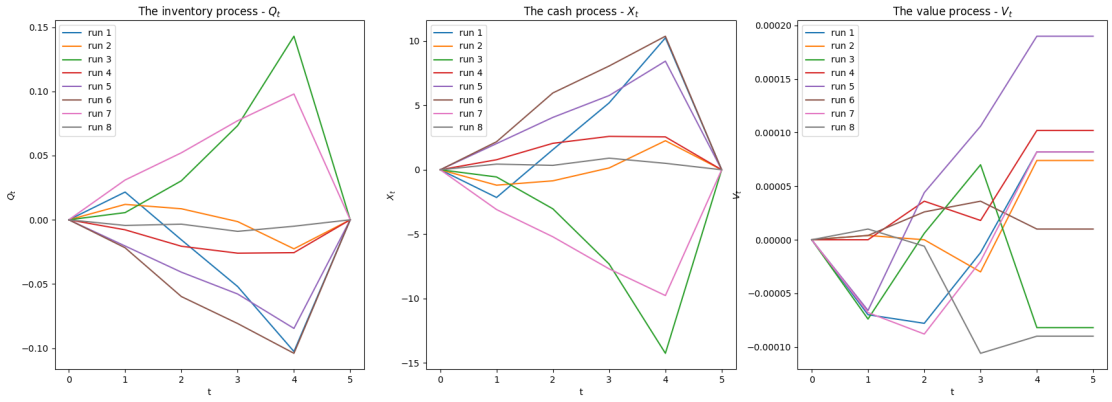


Figure 3.28: The average of the inventory, cash process, and value processes for all eight individual runs obtained from training in the SH-DDQN setting. Evaluated for 5×10^3 episodes.

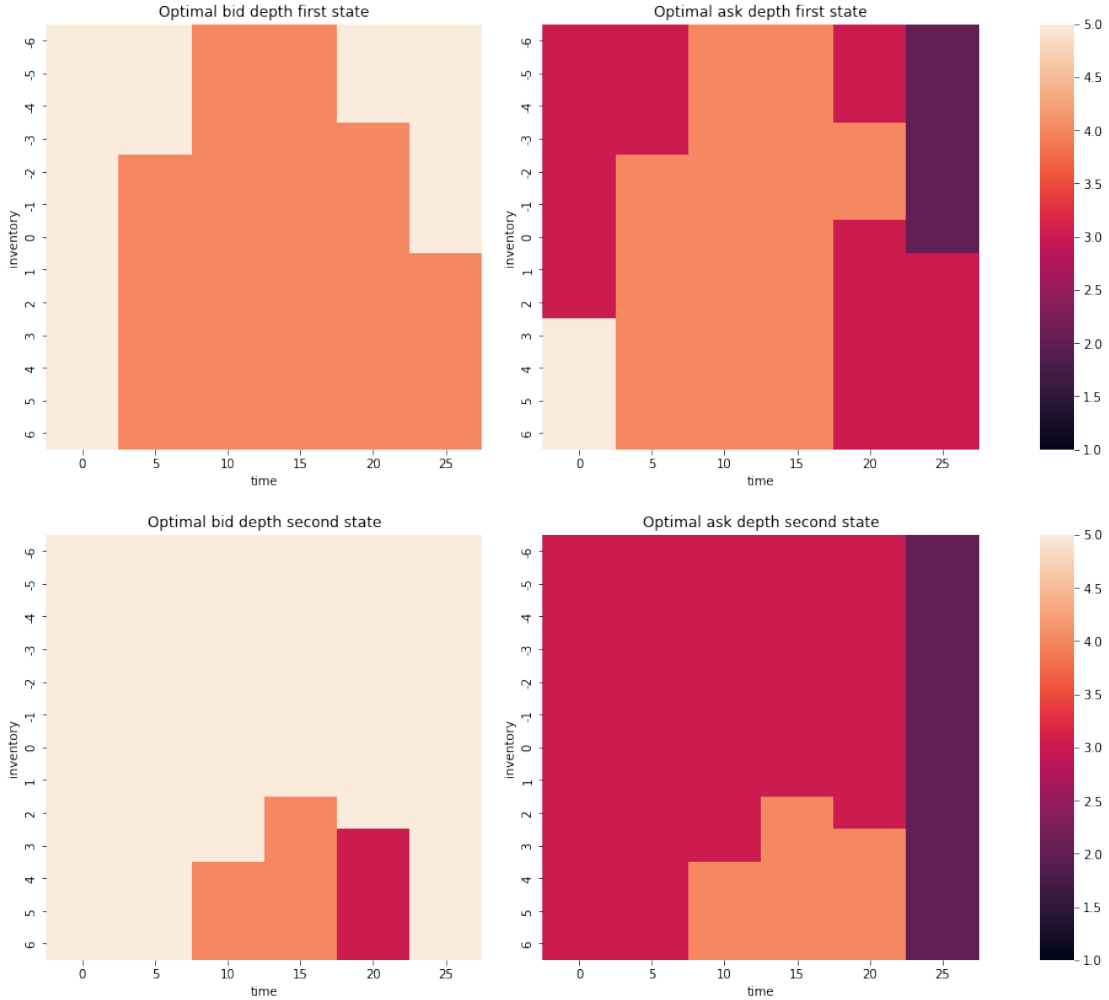


Figure 3.29: Difference in strategy between (**top**) neutral to (**bottom**) buy-heavy LOB in the SH-DDQN setting.

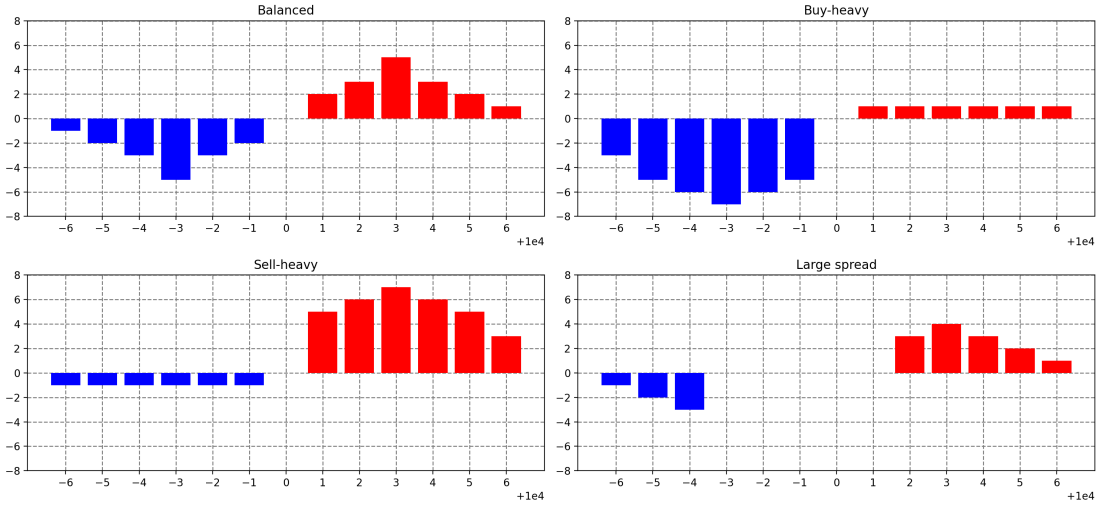


Figure 3.30: Illustrations of the four LOBs used in the scenario analysis. The volume on all levels outside of the displayed price levels is one (negative for buy volumes).

LOB state	$m(t)$	$s(t)$	$\mu(t)$	$\rho_{10}(t)$
Balanced	100.00	0.02	100.0000	0.00
Buy-heavy	100.00	0.02	100.0067	0.60
Sell-heavy	100.00	0.02	99.9933	−0.60
Large spread	98.00	0.06	99.9967	−0.36

Table 3.18: Some financial measures of LOBs used in scenario analysis. Mid price – $m(t)$, bid-ask spread – $s(t)$, micro price – $\mu(t)$ and order imbalance for 10 levels – $\rho_{10}(t)$.

LH-DDQN and Randomized LOB

This experiment uses DDQN and a randomized LOB in the LH setting, as described earlier in this section (Section 3.2.3). Forasmuch as employing a randomized LOB compared to a neutral LOB in the SH-DDQN experiments proved superior, the same is done here.

The DDQN was run eight times à 10,000,000 steps. The results of these runs are displayed in Figure 3.31 and Table 3.19. All runs provided strategies resulting in relatively large positive rewards with a small variance between runs. Most strategies also had a standard deviation smaller than their average reward. An exploration of the stability of the runs can be found in Appendix B.16.

Figure 3.32 plots the average reward, the state-value at $(t, Q_t) = (0, 0)$, and the loss during training. The state value is estimated by calculating the average over 10,000 random states where $(t, Q_t) = (0, 0)$. While $\hat{v}(0, 0)$ is negative towards the end of training and does not seem to have converged, the high rewards indicate that the agent has learned something valuable.

Run	$\overline{r}$	σ_r	$\overline{r}$ **per action**	$\overline{r}$ **per second**	$v_*(0,0)$
1	2.55	1.92	2.55×10^{-3}	2.55×10^{-3}	1.12
2	2.92	3.28	2.92×10^{-3}	2.92×10^{-3}	2.52
3	2.63	2.04	2.63×10^{-3}	2.63×10^{-3}	0.94
4	4.73	2.76	4.73×10^{-3}	4.73×10^{-3}	1.54
5	1.45	**1.31**	1.45×10^{-3}	1.45×10^{-3}	0.48
6	**4.80**	3.63	$\mathbf{4.80 \times 10^{-3}}$	$\mathbf{4.80 \times 10^{-3}}$	1.39
7	2.22	2.80	2.22×10^{-3}	2.22×10^{-3}	**3.30**
8	3.94	5.02	3.94×10^{-3}	3.94×10^{-3}	1.06

Table 3.19: The average reward received when following the obtained strategies from DDQN in the LH-DDQN with randomized resets for $1,000$ episodes. Bold values highlight the best result: the highest reward or the lowest standard deviation.

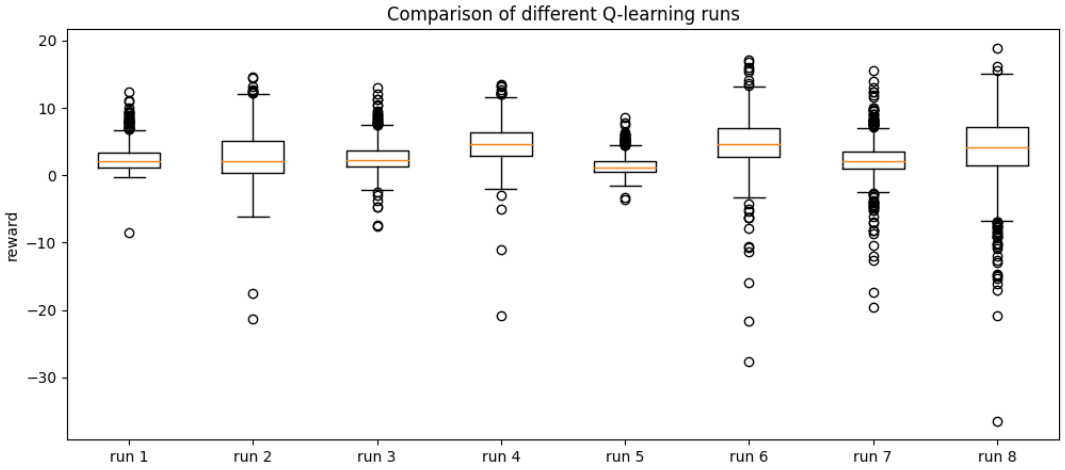


Figure 3.31: Boxplot of the reward of eight different DDQN runs in the LH-DDQN setting with randomized resets when evaluated for $1,000$ episodes.

The obtained mean strategy is displayed in Figure 3.33. Like in the SH-DDQN setting, the strategy seems to be "flipped" compared to our expectations. From the figure, one can see that strategy is very adaptive, taking both inventory and time into consideration. Looking closer into the proposed order depths, the mean strategy would, on average, seek a long position at the beginning of the episode and start selling it off later.

The best run and the mean strategy are compared against the benchmarking strategies in Figure 3.34 and Table 3.20. Both the best run and the mean strategy significantly outperform the benchmarks. However, the standard deviations of the DDQN strategies are higher than the standard deviations of both benchmarking strategies.

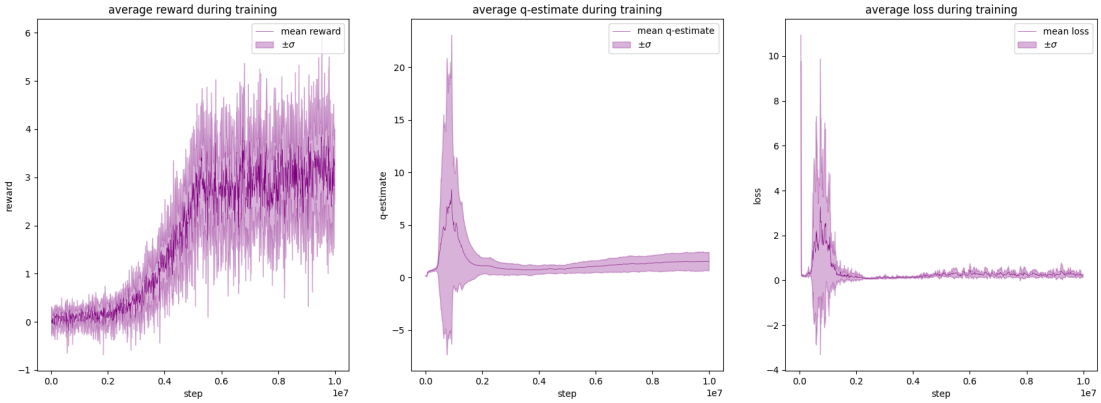
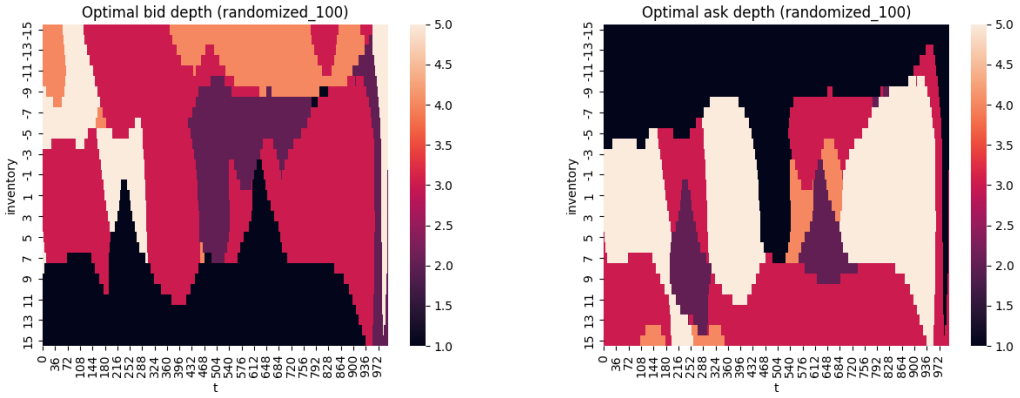


Figure 3.32:
Training in the LH-DDQN setting for 10^7 steps.
Left: the average reward during training with a shaded area showing ± one standard deviation.
Middle: the state-value at $(t, Q_t) = (0, 0)$ during training with a shaded area showing ± one standard deviation.
Right: the average loss during training with a shaded area showing ± one standard deviation.



(a) (b)

Figure 3.33: These two panels are obtained by averaging the action-values over 100 randomized states from the eight agents trained in the LH-DDQN. **(a)** shows a heat map of the optimal bid depths, and **(b)** shows a heat map of the optimal ask depths.

Strategy	$\overline{r}$	σ_r
DDQN		
- *best run*	**4.80**	3.63
- *mean strategy*	4.03	3.15
Benchmark		
- *constant*	0.187	2.15
- *random*	0.0402	**0.723**

Table 3.20: The average reward received when following different strategies in the LH-DDQN for 1,000 episodes. Bold values highlight the best result: the highest reward or the lowest standard deviation.

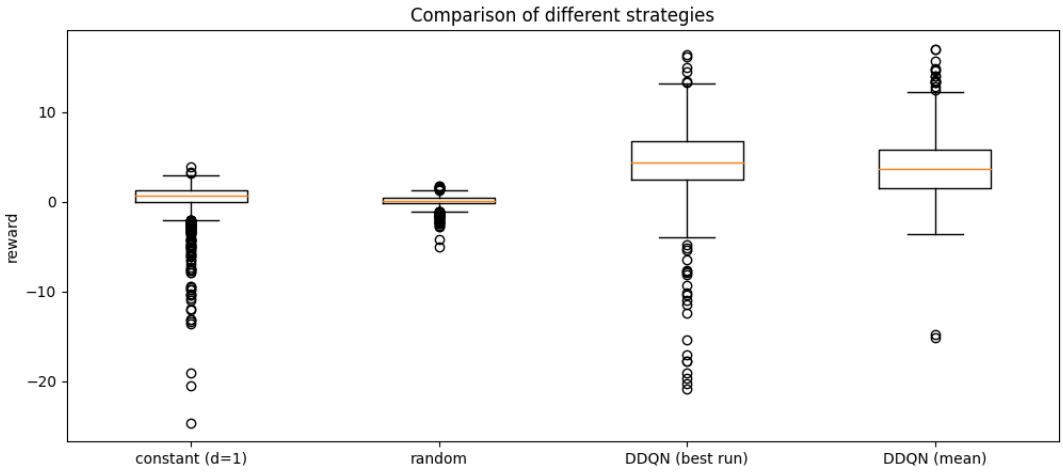


Figure 3.34: Boxplot of the reward of different strategies in the LH-DDQN setting when run for $1,000$ episodes.

The resulting inventory, cash, and value processes of the mean strategy and the individual runs are illustrated in Figure 3.35 and Figure 3.36. Figure 3.36 illustrates that no particular skewing direction is beneficial for higher rewards, just like in the SH-DDQN setting. One also observes that while the value creation is slow initially, it accelerates significantly after about one-third of the episode for all runs.

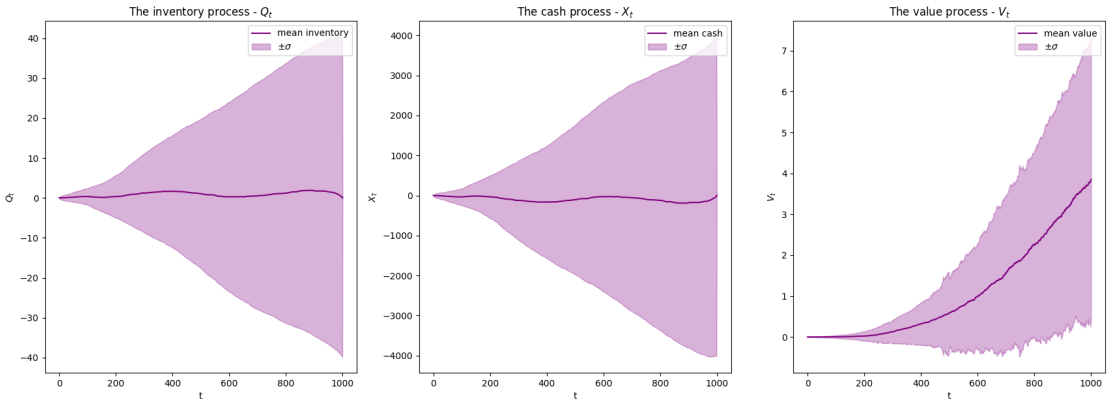


Figure 3.35: The average of the inventory, cash process, and value processes when following the mean strategy obtained from training in the LH-DDQN setting. Also showing $\pm$ one standard deviation in the shaded area. Evaluated for $1,000$ episodes.

Once again, an example of the mean strategy in two different states is provided and shown in Figure 3.37. The change in strategy between the two states is quite subtle; however, Figure 3.38 shows the actual absolute difference between the strategies. From these figures, it is evident that the change in strategy is relatively small; most changes occur at small absolute inventories. The strategies for the two remaining scenarios, "sell-heavy" and "large spread," are shown in Appendix B.17 and B.18, respectively.

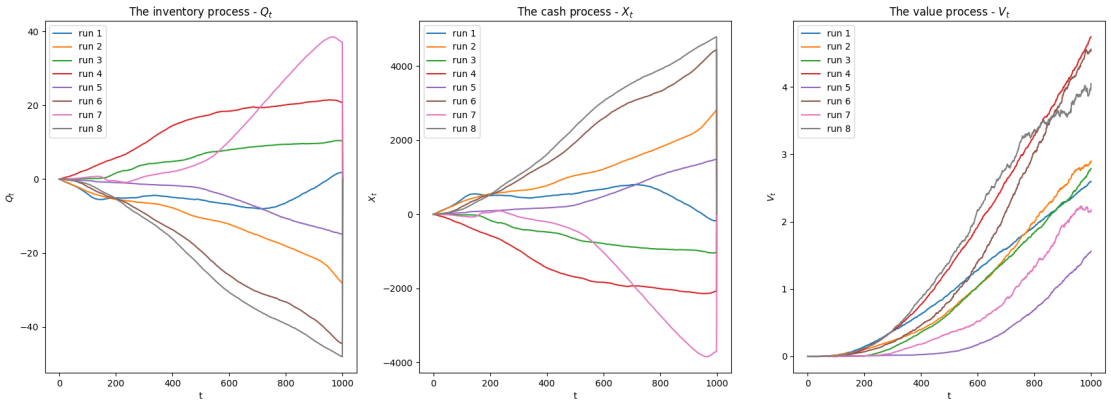


Figure 3.36: The average of the inventory, cash process, and value processes for all eight individual runs obtained from training in the LH-DDQN setting. Evaluated for $1,000$ episodes.

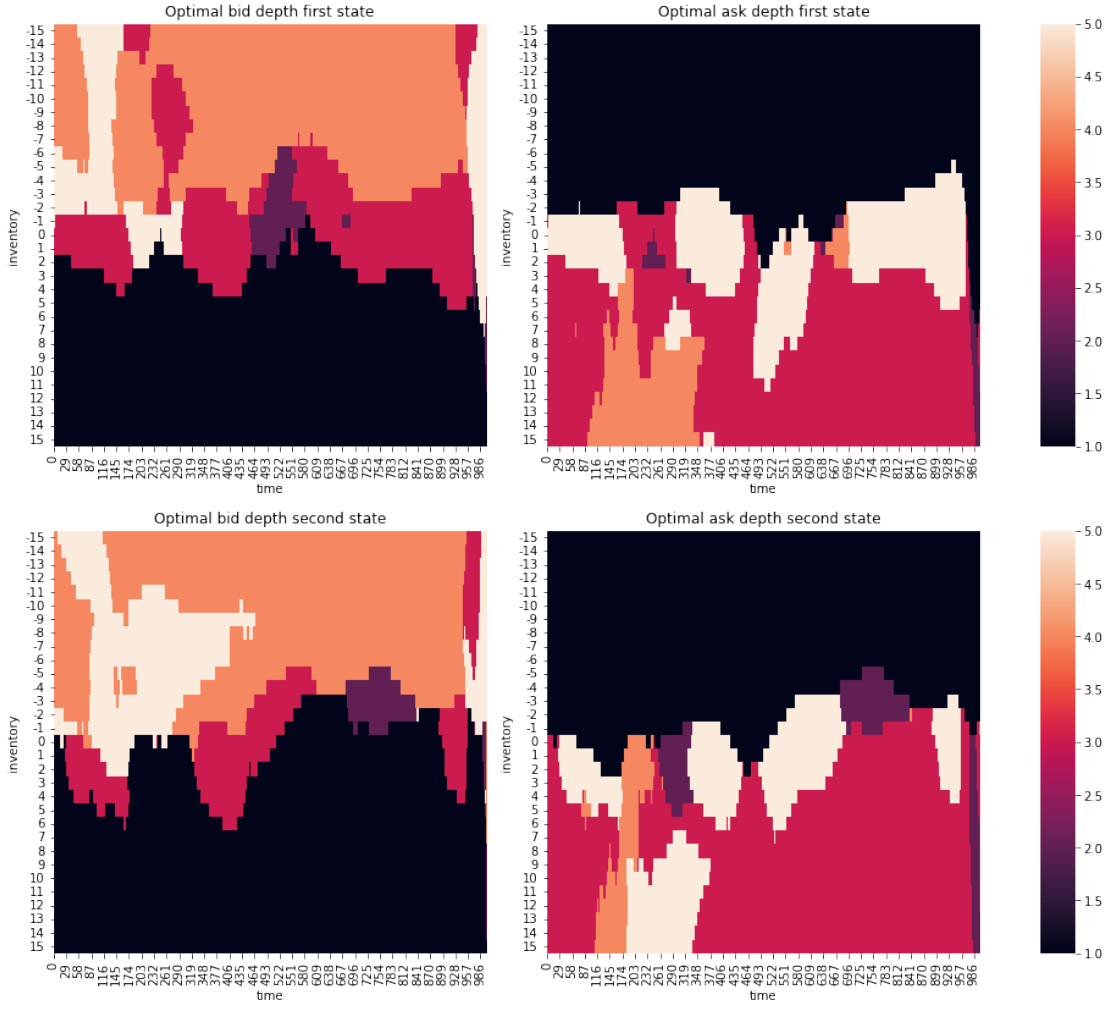


Figure 3.37: Change from (**top**) neutral to (**bottom**) buy-heavy LOB in the LH-DDQN setting.

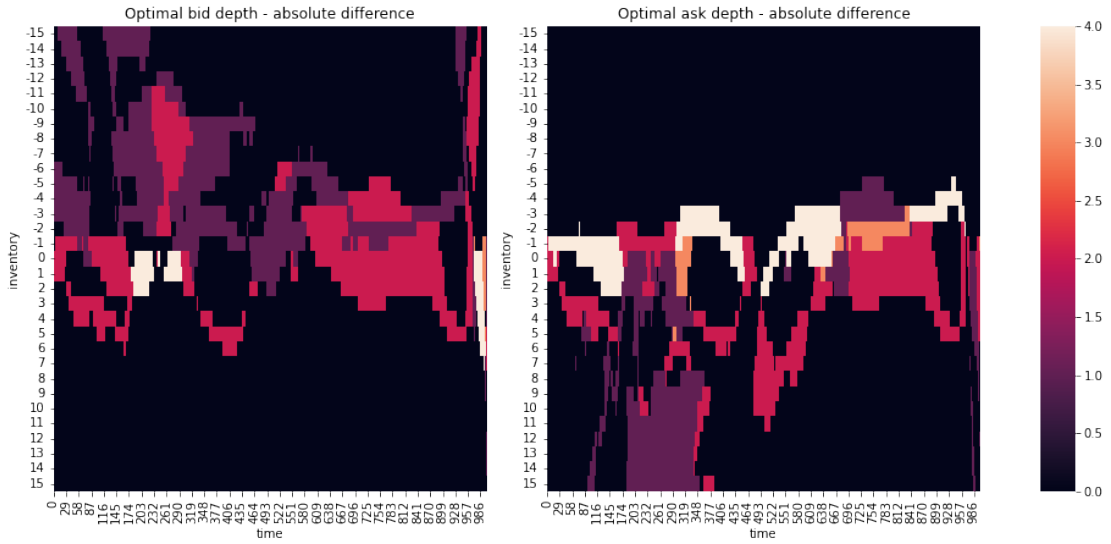


Figure 3.38: The absolute difference in order depth between the two strategies based on the neutral and buy-heavy LOB.

3.3 SUMMARY OF RESULTS

This section summarizes the results presented in Section 3.1 and Section 3.2. This is done through comparing average rewards and standard deviations, hypothesis tests, and strategy visualization. The outline of this section is:

- 3.3.1 Simple Probabilistic Model
- 3.3.2 Short-Horizon MC Model
- 3.3.3 Long-Horizon MC Model

3.3.1 *Simple Probabilistic Model*

Starting with the SPM, the goal was to answer **RQ1** – *are reinforcement learning methods able to outperform analytically derived strategies?* Table 3.21, which summarizes the analytical, benchmarking, and Q-learning strategies, shows that the mean Q-learning strategy has the highest average reward despite its relatively high standard deviation. That the Q-learning strategy is the highest performing one is somewhat confirmed by Table 3.6. However, the p-value is not low enough to confidently declare that the mean strategy outperforms the analytical continuous strategy.

Strategy	$\overline{r}$	σ_r	$\overline{r}$ **per action**	$\overline{r}$ **per second**
Q-learning				
- *best run*	3.529×10^{-2}	3.211×10^{-2}	7.058×10^{-3}	7.058×10^{-3}
- *average of runs*	3.524×10^{-2}	3.202×10^{-2}	7.048×10^{-3}	7.048×10^{-3}
- *mean strategy*	$\mathbf{3.542 \times 10^{-2}}$	3.205×10^{-2}	$\mathbf{7.084 \times 10^{-3}}$	$\mathbf{7.084 \times 10^{-3}}$
Analytical				
- *discrete*	3.506×10^{-2}	3.132×10^{-2}	7.012×10^{-3}	7.012×10^{-3}
- *continuous*	3.537×10^{-2}	3.060×10^{-2}	7.074×10^{-3}	7.074×10^{-3}
Benchmarks				
- *constant*	2.674×10^{-2}	$\mathbf{2.848 \times 10^{-2}}$	5.348×10^{-3}	5.348×10^{-3}
- *random*	1.852×10^{-2}	3.520×10^{-2}	3.704×10^{-3}	3.704×10^{-3}

Table 3.21: The rewards of the Q-learning, analytically optimal and benchmark strategies in the SPM setting. Evaluated for 10^6 episodes. Bold values highlight the best result: the highest reward or the lowest standard deviation.

In Figure 3.39, a visualization of the inventory, cash, and value process of the different strategies when run for $1,000,000$ episodes is presented. It is clear that the benchmarking strategies are the worst off; however, the difference between the analytical and Q-learning strategies is hard to distinguish. Worth noting is that the asymmetry present in the mean strategy (see Figure 3.4) is very observable.

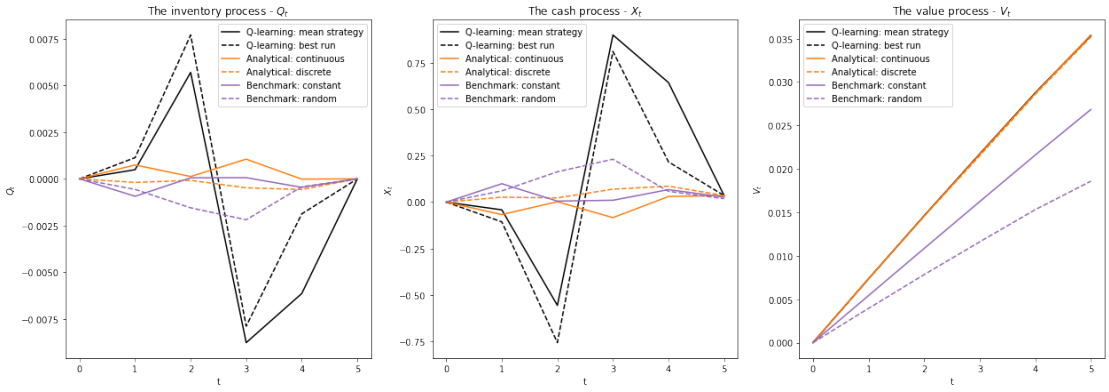


Figure 3.39: The average of the inventory, cash, and value processes when following the different strategies in the SPM setting. Simulated for 10^6 episodes.

3.3.2 *Short-Horizon MC Model*

Continuing with the SH setting, we may partly answer **RQ2** – *can deep reinforcement learning algorithms learn better performing market making strategies than tabular Q-learning?* The performance of the different strategies is summarized in Table 3.22. All strategies derived from reinforcement learning methods beat the benchmarks. Comparing Q-learning and DDQN, one observes that the latter slightly outperforms the former when looking at the best run. Furthermore, the small difference in $\overline{r}$ between the best run and the mean strategy of DDQN indicates that it is more stable in this setting. The conclusion that DDQN outperforms Q-learning in this setting is further supported by Table 3.23 where most p-values are small.

In general, one can also conclude that it is hard for a market maker to earn a profit in this setting; the available time in the short-horizon is not enough to overcome the forced liquidation.

One should also note that despite this setting having the same episode length as the SPM, the results are not comparable since the underlying models differ significantly.

Strategy	$\overline{r}$	σ_r	$\overline{r}$ **per action**	$\overline{r}$ **per second**
Q-learning				
- *best run*	1.5×10^{-4}	7.97×10^{-3}	3.1×10^{-5}	6.2×10^{-6}
- *average of runs*	6.5×10^{-5}	7.31×10^{-3}	1.3×10^{-5}	2.6×10^{-6}
- *mean strategy*	2.5×10^{-5}	$\mathbf{2.43 \times 10^{-3}}$	5.0×10^{-6}	1.0×10^{-6}
DDQN				
- *best run*	$\mathbf{2.1 \times 10^{-4}}$	8.28×10^{-3}	$\mathbf{4.2 \times 10^{-5}}$	$\mathbf{8.5 \times 10^{-6}}$
- *average of runs*	4.5×10^{-5}	8.64×10^{-3}	9.1×10^{-6}	1.8×10^{-6}
- *mean strategy*	1.4×10^{-4}	5.61×10^{-3}	2.8×10^{-5}	5.6×10^{-6}
Benchmarks				
- *constant*	-1.3×10^{-3}	2.73×10^{-2}	-2.6×10^{-4}	-5.2×10^{-5}
- *random*	-5.3×10^{-3}	2.89×10^{-2}	-1.1×10^{-3}	-2.1×10^{-4}

Table 3.22: The rewards of the Q-learning, DDQN, and benchmark strategies in the SH setting. Evaluated for 5×10^4 episodes. Bold values highlight the best result: the highest reward or the lowest standard deviation.

		Q-learning	
		- best run	*- mean strategy*
DDQN	*- best run*	0.122	8.18×10^{-7}
	- mean strategy	0.591	1.30×10^{-5}

Table 3.23: The p-values for which the null hypothesis that the Q-learning strategies' average rewards are larger than the DDQN strategies' average rewards can be rejected. In the SH setting. Welch's t-test with a common sample size of $n = 5 \times 10^4$.

In Figure 3.40, a visualization of the inventory, cash, and value process of the different strategies when run for $1,000,000$ episodes is presented. Clearly, the benchmarking strategies perform the worst. While the Q-learning and DDQN seem almost indistinguishable in the value process, they differ significantly in the inventory and cash processes. The DDQN strategies on average skew toward a positive inventory, while the Q-learning strategies seem more reluctant to trade, which also can be seen when comparing the strategies in Figure 3.12 and Figure 3.25.

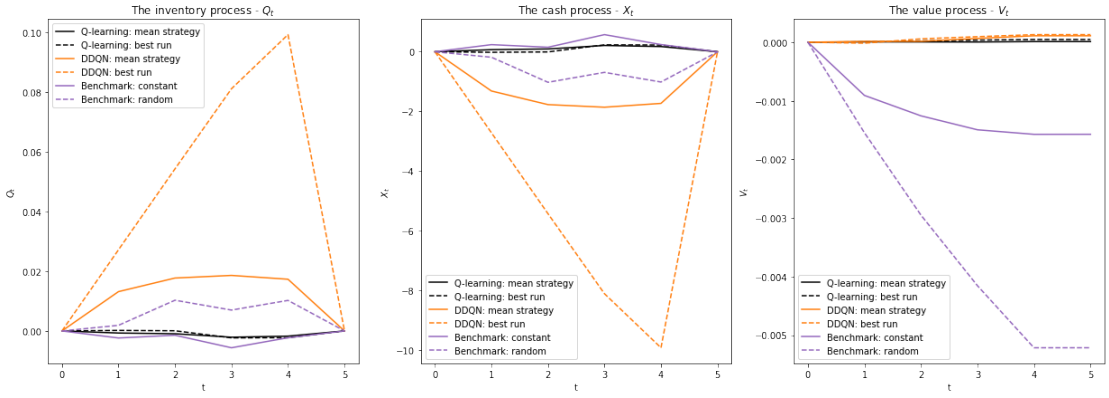


Figure 3.40: The average of the inventory, cash, and value processes when following the different strategies in the SH setting. Simulated for 5×10^4 episodes.

3.3.3 *Long-Horizon MC Model*

Lastly, with the LH setting, we hope to strengthen our hitherto conclusions made regarding **RQ2** – *can deep reinforcement learning algorithms learn better performing market making strategies than tabular Q-learning?* Once again, the performance of the different strategies is presented in Table 3.24. Like in the SH setting, DDQN outperforms Q-learning in terms of average reward and stability, which is greatly supported by Table 3.25. On the contrary, not all reinforcement learning strategies beat the benchmark strategies; the mean Q-learning strategy is the worst with a negative reward, suggesting significant variance in the training.

Furthermore, looking at the reward per second, the reinforcement learning is about 1,000 times better in the LH setting than in the SH. This is most likely due to the longer episodes outweighing the forced liquidation's negative effects.

Strategy	$\overline{r}$	σ_r	$\overline{r}$ **per action**	$\overline{r}$ **per second**
Q-learning				
- *best run*	3.65	5.95	3.65×10^{-3}	3.65×10^{-3}
- *average of runs*	2.18	4.55	2.18×10^{-3}	2.18×10^{-3}
- *mean strategy*	-3.68×10^{-1}	$\mathbf{3.73 \times 10^{-1}}$	-3.68×10^{-4}	-3.68×10^{-4}
DDQN				
- *best run*	**4.80**	3.63	$\mathbf{4.80 \times 10^{-3}}$	$\mathbf{4.80 \times 10^{-3}}$
- *average of runs*	3.16	3.04	3.16×10^{-3}	3.16×10^{-3}
- *mean strategy*	4.03	3.15	4.03×10^{-3}	4.03×10^{-3}
Benchmarks				
- *constant*	3.36×10^{-1}	7.10×10^{-1}	3.36×10^{-4}	3.36×10^{-4}
- *random*	4.85×10^{-2}	7.07×10^{-1}	4.85×10^{-4}	4.85×10^{-4}

Table 3.24: The rewards of the Q-learning, DDQN, and benchmark strategies in the LH setting. Evaluated for $1,000$ episodes. Bold values highlight the best result: the highest reward or the lowest standard deviation.

		Q-learning	
		- *best run*	- *mean strategy*
DDQN	- *best run*	8.75×10^{-8}	0
	- *mean strategy*	0.0365	0

Table 3.25: The p-values for which the null hypothesis that the Q-learning strategies' average rewards are larger than the DDQN strategies' average rewards can be rejected. In the LH setting. Welch's t-test with a common sample size of $n = 1,000$.

In Figure 3.41, a visualization of the inventory, cash, and value process of the different strategies when run for $1,000$ episodes is presented. The mean Q-learning strategy and the benchmarking strategies perform the worst. The difference between the mean DDQN strategy and the best Q-learning run is small in terms of value, but they clearly diverge at the end of the episode, most likely due to the time-grouping in the LH-Q setting (for the reader who skipped the detailed results it may be interesting to visit 3.2.2 Q-Learning to read more about this binning). Another interesting observation is that there seems to be no particular way to skew to earn more profits. Whereas the best runs of Q-learning and DDQN on average skew in different directions, the mean DDQN strategy does not systematically skew in a specific direction and still earns a comparable profit. However, from the strategy's standard deviation and heatmap visualization, it is evident that it skews in the direction of its current inventory, which averages out to no particular skewing over multiple episodes. The distribution of skewness direction seems completely random when looking at the individual runs in Table 3.11 and Table 3.19, which further confirms that no particular skewing direction is superior.

Looking at the mean DDQN strategy in Figure 3.41, one can also notice that the profitmaking does not decelerate during the end of the episode as for the other strategies.

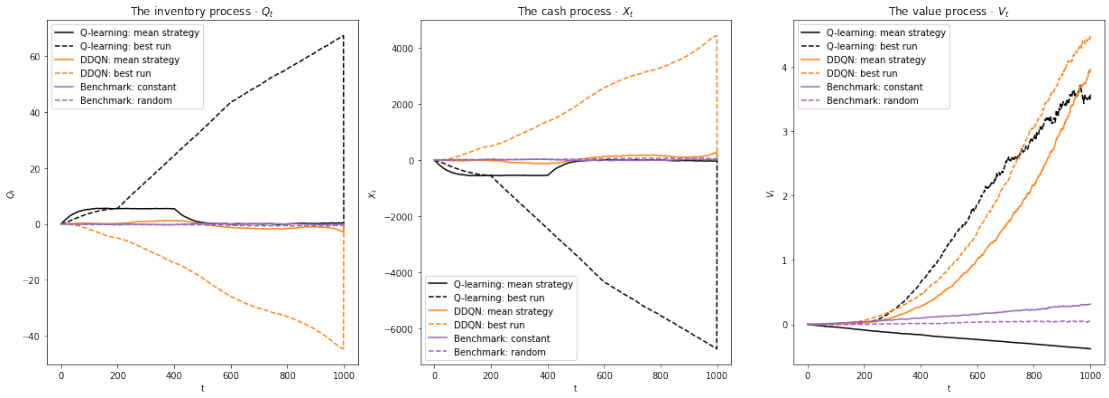


Figure 3.41: The average of the inventory, cash, and value processes when following the different strategies in the LH setting. Simulated for $1,000$ episodes.

4 DISCUSSION AND FUTURE WORK

4.1 DISCUSSION

In this section, the objective and the research questions will be discussed and answered. This will be done by comparing the different market making strategies presented in the report regarding average reward, standard deviation, and other characteristics. The different model settings and reinforcement learning algorithms will be examined, including topics such as convergence, stability, and the prospects of profitable market making. Furthermore, the intricacies of the framing of the MDP will be elaborated on, more specifically, the state space, the action space, and the reward signal. Finally, an analysis of the implementation possibilities of the learned market making strategies is presented.

Reinforcement Learning versus Analytical Strategies

Research question 1 (**RQ1**) was phrased as:

Are reinforcement learning methods able to outperform analytically derived strategies?

Despite this question being challenging to answer, the tests run on the SPM can be of help. Analytically optimal strategies are, per definition, unable to be beaten; however, they are based on crude models of reality, limiting their use in real-world markets. The SPM is a discretized version of the model provided in [11], an adaptation that most likely renders the optimal strategies suboptimal, similar to other translated analytical strategies. Using Q-learning, strategies that outperform the analytical ones in terms of average reward were found, showing that reinforcement learning can adapt to the new way of simulating the market.

In terms of the actual strategies obtained from the Q-learning, they differ slightly from the analytical strategies. While 85% of the optimal actions across all possible states are identical, the small differences lead to a significant difference in inventory buildup. The Q-learning strategies on average skew slightly toward a positive inventory while the analytical strategies are symmetrical, thus averaging an inventory of 0.

Although this experiment is by no means conclusive concerning analytical versus reinforcement learning derived strategies, it is an example of model-free machine learning outperforming traditional strategies in market making. The difference may be even larger in real-world markets which are model-free.

The Simple Probabilistic Model versus the Markov Chain Model

In order to compare the two environments, a similar problem formulation was constructed, a short episodic trading window ending with a forced liquidation of the market maker's current position. Despite this, the results are far from comparable due to model differences, mainly due to their differences in complexity and arrival intensities.

Looking at the differences, one first notices that in the SPM, the strategies are on average 100 times more profitable, even including the running inventory penalty. Second, another significant difference in average reward over the standard deviation ($\overline{r}/\sigma_r$), which is around 1 for the SPM and 10–100 for the MC model. Third, the heatmaps of the strategies in the SPM setting are very smooth in comparison to those of the MC model, indicating that the Q-learning probably has not fully converged in the latter setting, which can be seen in Figure 3.17. Here one should note that training was run significantly longer in the SPM setting, 15,000,000 episodes versus 1,600,000 / 40,000 episodes (SH / LH). Fourth, the average rewards over the eight different runs differ significantly in both the SH-Q and the LH-Q setting, indicating that the Q-learning is less stable in the MC model. Additional proof of this can also be seen in that the mean strategies in the MC model perform much worse than the individual runs, in contrast to the mean strategy being the best in the SPM setting.

The benchmarking strategies confirm the overall difference in difficulty in performing profitable market making in the two models. While the constant and the random strategies are able to make a profit in the SPM, they instead make a loss in the MC model. This suggests that the more sophisticated the market model is, the more intricate the market making strategy has to be.

The Short-Horizon versus the Long-Horizon Setting

Two settings of the MC model are considered in this thesis, the SH and the LH. They differ in the length of the trading window as well as the market maker's action frequency. Looking at the time-adjusted average reward, strategies in the LH setting outperform those in the SH setting with a factor of 1,000. From the visualizations of the learned strategies, it is also clear that the longer horizon allows for riskier strategies, which can be seen in the build-up of inventory per second being almost twice as high for the LH setting. Moreover, the longer episode means that the increased number of time steps dilutes the negative impact of the forced liquidation.

Focusing on the Q-learning settings, SH-Q and LH-Q, the agent will not learn to be especially risk-averse in the SH-Q setting since it will not build large inventories. This is clearly visible from the two strategies – in the SH-Q setting, there is no clear gradient, whereas, in the LH-Q setting, there is a clear gradient which indicates that the agent has learned to be risk-seeking. It could also be argued that the agent in the SH-Q setting is risk-averse since it relatively consistently chooses a large depth.

Furthermore, there is a significant difference in the convergence of the Q-learning. As already noted, the state-value of the first model converges toward the expected reward over an episode, whereas the state-value of the time-grouped model does not converge to it. Nonetheless, the Q-learning in the time-grouped model manages to learn an intuitive strategy, albeit a strategy that loses money.

Taking a look at the mean strategies for the Q-learning on the MC model one notices that they perform significantly worse than the individual runs, indicating that there may be many local minima for this problem, in contrast to the SPM. However, the mean strategy is the worst for the LH-Q setting, averaging a negative reward, which is probably due to its lack of the Markov property.

Looking at the optimal strategies at $(t, Q_t) = (0, 0)$ for all individual runs in the two models, we see much more agreement between the runs in the SH-Q setting than in the LH-Q setting. This demonstrates the fact that the Q-learning has not converged, and may explain why the

average strategy in the LH-Q setting significantly underperforms the individual runs whereas, for the SH-Q setting, the average run is only slightly worse-performing than the best individual run.

Continuing with the DDQN settings, SH-DDQN and LH-DDQN, one notices some similarities and some differences. First, the DDQN agents learn to be more risk-seeking in the LH-DDQN setting than in the SH-DDQN; however, in the SH-DDQN setting both the best run and the mean strategy on average skew towards a positive inventory (see Figure 3.40), indicating the knowledge of the LOB helps mitigate risks of skewing. Secondly, despite the agent now having access to the whole LOB, the length of the SH setting is too short to allow for decent profit-making. Thirdly, the mean DDQN strategies are not significantly worse than the individual runs (see Tables 3.17 and 3.20) like for Q-learning, once again indicating the importance of observing the whole LOB. Finally, like for the Q-learning, the DDQN does not seem to have fully converged as the optimal strategies at $(t, Q_t) = (0, 0)$ for all individual runs in the two settings seem to differ substantially (see Figure B.15 in Appendix B.16).

Another result worth commenting on is an asymmetry in the bid and ask depths in the starting state for all settings except SH-Q. There is no reasonable explanation other than pump-and-dump schemes (which will be discussed later) that we can think of that would explain this phenomenon from a financial/trading perspective. It is more logical to see the symmetrical behavior in the SPM and the SH-Q setting. Notably, this is not the result of averaging the strategies – the asymmetry is found in essentially all individual runs, as demonstrated in Table 3.11.

Tabular versus Deep Reinforcement Learning

Our second research question (**RQ2**) that we sought to answer was

> *Can deep reinforcement learning algorithms learn better performing market making strategies than tabular Q-learning?*

Simply looking at the tables and figures in Section 3.3, it is quite clear that DDQN manages to find strategies that outperform Q-learning in terms of average reward and risk-adjusted average reward ($\overline{r}/\sigma_r$). Also, most p-values are small enough in order to statistically support these claims. Furthermore, the variance between the different runs is smaller for DDQN than Q-learning, indicating that it is more stable. Additionally, DDQN converged more rapidly, both in terms of the number of episodes and training time.

However, the DDQN agents were given better conditions, no binning of time nor inventory was used, and the agents could observe the full LOB. While this may be considered "cheating," being able to use large state spaces is one of the strong points of deep reinforcement learning methods; the state space used for DDQN would have been infeasible for Q-learning.

Both Q-learning and DDQN were quite similar in how much fiddling was needed for learning to occur. While it involved model parameters (e.g., the size of the inventory grouping) and hyperparameters (e.g., learning rate and exploring starts) for Q-learning, it involved additional hyperparameters for DDQN (e.g., network architecture and reward scale), which warranted further tuning.

State Space

For all experiments involving Q-learning, the state space was essentially the same and consisted only of the elapsed time and the market maker's inventory. In one of the experiments, this was

slightly altered such that the time state space variable was binned instead, see Long-horizon setting for Q-learning. This quite uninformative state space was sufficient to infer market making strategies that outperformed the benchmarking strategies.

In all of our experiments, we included time in the state space. As we discussed in Section 2.3.2, this is not particularly common, and only 5/22 articles studied by [25] did so. It is possible that the added state space complexity arising from this has negatively impacted the learning of optimal strategies in our experiments and that it would have been better to include something potentially more informative such as the bid-ask spread. The primary reason for our inclusion of time was to compare the strategies in the Markov chain LOB setting with the SPM. An additional motivation is simply that it is sensible for a market maker to consider the remaining time of a trading session to suitably time the liquidation of inventory. We also considered and experimented with omitting time *per se* and using a binary variable to indicate if the trading session is in the last $x\%$ (e.g., 10%) of the session (however, we included no results from these experiments in this written thesis). This is quite comparable to the grouping of time that we attempted for the MC LOB model. Similar issues were run into for the binary coding. We provided a plausible explanation for the failure of time-binning in the section Long-Horizon setting for Q-learning. In essence, it boils down to the fact that the Markov property of the decision process is further violated when the time is binned (the observation space becomes only partially observable). To keep the Markov property and have full observability and thus guaranteed convergence for Q-learning, we would have to include the whole state (including the entire LOB and the precise inventory level); for tabular methods, this becomes infeasible. To omit the LOB and bin the inventory worked well despite only being partially observable. When the observability was decreased additionally by binning time, issues were run encountered.

Contrary to many published papers, we did not include any measures, including imbalances, volatility, RSI, etc., in the state space. For the Q-learning experiments, this was to keep the state space as small as possible, and in the DDQN experiments, we wanted to see if only the "raw" data were sufficient for the market maker to learn good strategies. One contribution of this thesis is that the agent in our DDQN experiments only observes "raw" data, such as the time, bid-ask spread, inventory, and LOB data. To the best of our knowledge, no other published paper only uses extensive LOB data (volume data at up to 20 price levels) without any constructed imbalance, flow measures, or similar. Sadighian uses quite extensive (but altered) LOB data in conjunction with various imbalances etc. [60, 61].

Action Space

The action spaces used throughout our experiments were practically identical and consisted only of the posted order depths on the bid and ask sides of the market maker. However, focusing only on the pricing aspects of market making and disregarding the order sizes, etc., reduces the action space significantly, increasing the algorithms' learning speed.

We planned to include a market order in the action space for our DDQN experiments (observable in Algorithm 2 where a code snippet dealing with MOs remains). However, we did not have the time to experiment with this. The action space we planned to use is defined as

$$\mathcal{A} = \mathcal{D} \times \mathcal{M} = \{1, \dots, \bar{d}\}^2 \times \{0, 1\}, \tag{77}$$

where $\mathcal{D}$ is the order depth action space and $\mathcal{M}$ is the market order action space. The market order action space has a binary coding where 1 indicates that the market maker places a market order with the volume defined as in Algorithm 2.

As previously discussed, some authors include an action to place a market order to reduce the inventory, e.g., [60, 64]. However, we choose to implement it somewhat differently in our model. Firstly, they treat it as an entirely separate action; that is, the market maker does not have the option to place a market order and immediately afterward post two-side quotes. That has to be done on two different time steps, which in our case would entail a latency of one second between those actions. Defining an entirely separate action that is to post a market order to clear the inventory and not post limit orders may result in the issue that the market maker – despite having no inventory – can place a market order with volume zero (i.e., to do nothing virtually) for an arbitrarily long time period. This is undesirable since the market maker should provide prices as long as possible.

Secondly, rather than liquidating the entire inventory in one stroke, the market maker is limited in our model regarding the order volume as hinted at in Algorithm 2. Our motivation for this is two-fold: price impact and to avoid the issue of an increased action space for specifying the MO volume.

Rewards

Initially, we planned to examine a range of reward functions, but due to a lack of time, this was unfortunately unattainable. Throughout all of our experiments, we used the reward

$$R_t = \Delta V_t - \phi Q_t^2, \tag{78}$$

where $\phi = 0$ for the experiments with the Markov chain LOB model. The two major advantages of this reward construction are the high frequency and interpretability of rewards. Most reward formulations found in published articles on reinforcement learning for market making are frequent, as seen in Section 2.3.2 (close to all reward setups yield frequent rewards). Regarding the interpretability, our formulation is essentially unrivaled, particularly when $\phi = 0$ since the reward then equals the actual profit. Whenever penalization terms are introduced to the reward function, intuitive interpretability is lost.

In the cases where $\phi = 0$, no explicit inclusion of risk aversion is included in the problem formulation which may explain the lack of apparent gradients in some experiments.

Of reward formulations with an explicit risk aversion, we were particularly keen to test the asymmetrically dampened PnL, albeit with a minor tweak. Instead of using the mid price, we would use the micro price, which is more likely to change following order book events, thus resulting in more non-zero rewards that enable faster learning. The asymmetrically dampened PnL with micro price would be computed as:

$$\Delta AsDPnL_t = \delta_t^- q_t^b - \delta_t^+ q_t^a + Q_t \Delta\mu(t) - [\eta Q_t \Delta\mu(t)]_+, \tag{79}$$

where all quantities are defined in Section 2.3.2.

Considering the huge importance of the reward formulation on the output strategies of market making, an entire thesis could have been dedicated to only exploring various reward functions and analyzing and contrasting them.

Pump-and-Dump Schemes

Arguably, some of the strategies found by RL agents in this thesis may look reminiscent of pump-and-dump schemes (or the flip side, *bear raids*). A pump-and-dump scheme is a manipulative method where an actor acquires a position in some asset and subsequently artificially inflates the asset's price. Traditionally, the price inflation tends to ensue by the spreading of positive misinformation ("pumping"), and once the price has increased sufficiently, the actor will sell the position off ("dumping"). In addition to this traditional form of pump-and-dump, variants exist. In high-frequency and algorithmic trading, the prices can be artificially boosted by an aggressive purchasing of the asset, after which the position is liquidated and desirably a profit has been scored. [1] As we saw in Figure 3.9, the market maker's quoted depths have a significant impact on the mid price process, enabling pump-and-dump behavior to be possible in our environments. Huberman and Stanzl (see [37]) and Gatheral (see [27]) have researched enabling conditions for HFT pump-and-dump schemes and the presence of (quasi)arbitrage in settings of varying market impact of traders.

For legal reasons, it would be desirable to restrict the abilities of RL agents to manipulate markets in the fashion described above (since market manipulation is illegal); however, this is probably easier said than done. First and foremost, it may be challenging to identify a trading pattern that constitutes market manipulation. Secondly, how would one practically go about preventing this behavior? Limiting what the agent can do? Adjusting the agent's reward such that acting in specific illicit ways will be penalized? Merely addressing and researching these issues may require extensive scrutiny in the future.

Implementation and Challenges Thereof

Ideally, we would like our work on this thesis to be utilized by SEB as promptly as possible. However, this is far from realistic due to many reasons.

Current work on reinforcement learning and market making is still in an exploratory and learning phase. Financial markets are ever-evolving and substantially more complex than a game of Pong or Go. The consequences of various mistakes can carry enormous consequences.

For example, consider the five-minute Flash Crash in Nordic equity markets on May 2nd, 2022, where supposedly a "fat finger" mistake by an individual trader at Citigroup's London trading desk wiped out as much as €300bn across European markets at one point (the OMX Stockholm 30 Index tumbled by as much as 8%!) [67]. It is not entirely unimaginable that a trained RL agent could make equally devastating mistakes in the absence of proper fail-safes if, for instance, the RL agent observes a previously unvisited state and the deep Q-network does not generalize desirably. Other weaknesses of reinforcement learning algorithms also impede the likelihood of a prompt adoption of the technology in the industry.

Relating to the game of Go again, the reader interested in reinforcement learning probably recalls the Google DeepMind Challenge Match where DeepMind's RL-based agent AlphaGo played five games against the world champion, Lee Sedol. AlphaGo won games one, two, three, and five. In the fourth game, Sedol played move 78, for which AlphaGo seemed entirely unprepared. AlphaGo determined that the chance any human player would have played it was less than one in 10,000 why AlphaGo may have been unprepared. At that time, AlphaGo had an estimated probability of 79% of winning. However, AlphaGo's subsequent 15 moves or so consisted of repetitious mistakes, after which AlphaGo was unable to recover (despite strong gameplay) and

subsequently resigned. [3] Go enthusiast David Ormerod attributes the downfall of AlphaGo to a weakness of Monte Carlo tree searches. He wrote the following about game four and AlphaGo's undoing:

> *"We've seen these kinds of moves before, played by other Go AIs. They seem to be a common failure pattern of AIs which rely on Monte Carlo Tree Search to choose moves. AlphaGo is greatly superior to that previous generation of AIs, but it still relies on Monte Carlo for some aspects of its analysis. My theory is that when there's an excellent move, like White 78, which just barely works, it becomes easier for the computer to overlook it. This is because sometimes there's only one precise move order which makes a tesuji work, and all other variations fail. Unless every line of play is considered, it becomes more likely that the computer (and humans for that matter) will overlook it. This means that approximation techniques used by the computer (which work really well most of the time) will see many variations that lead to a win, missing the needle in the haystack that leads to a near certain loss."* [57]

Unquestionably, before the risk of similar catastrophic failures has been eliminated, it is implausible that RL agents will be put into production to act as market makers entirely autonomously.

Another illustrative example of why reinforcement learning may not be mature for actual trading is Norwegian daytraders Mr. Larsen and Mr. Veiby, who in 2010 were convicted for market manipulation. The men learned to predict how Timber Hill's computer algorithm – which was trading in various illiquid Norwegian stocks – would respond to various trades. They utilized their insights to manipulate the algorithm to score a profit – about NOK 250k and NOK 160k for the two men. Mr. Larsen said

> *"I did not set out to trick the robot [...] But after acting against it a few times, you see how it behaves. The computer was fairly obvious."* [74]

Similarly, some blame the Flash Crash of 2010 on high-frequency and algorithmic traders. Navinder Singh Sarao, the Brit responsible for the crash that for a while wiped out almost USD 1tn in value, was sentenced to three years in prison for *spoofing*. Spoofing is a type of market manipulation where one places many large orders and then quickly cancels or changes them. High-frequency trading (HFT) algorithms might pick up on this as a genuine change in demand and adjust their prices accordingly; this may result in either swiftly rising or falling prices allowing the properly positioned "spoofer" to secure a hefty profit. [75] To suitably safeguard algorithmic traders (e.g., RL agents) against such tactics is another obstacle that must be overcome before RL agents enter the financial markets as market makers.

The message of these two anecdotes is that even if the reinforcement learning agent behaves optimally in a testing environment or even in real markets, there are no guarantees that its strategy may not be observed and exploited by other market participants on the hunt for a profit. Instead, a reinforcement learning agent should ideally be able to identify such situations and adapt accordingly; however, as highlighted throughout this thesis, it takes a tremendous amount of experience for an agent to identify and learn (new) optimal behaviors. These issues remain major challenges.

Also, to avoid the uncertainties of black boxes, it might be wiser to start with non-deep RL methods for simpler tasks. Indeed, Dixon and Halperin write about reinforcement learning for financial applications:

> *"Our general recommendation is to avoid hastily applying deep reinforcement learning ahead of first using 'shallow' reinforcement learning — establishing, at a minimum, a benchmark for more advanced techniques such as deep Q-learning."* [21].

On the other hand, Jim Simons and Renaissance Technologies have for ages employed various black box methods for investing in equities, futures, FX, and commodities while maintaining an entirely unrivaled investing track record. Renaissance Technologies' much-celebrated Medallion fund scored an average annual return of 66.1% before fees between 2002 and 2018 (more than USD 100bn in total trading profits). [81] It might be that unleashing the powers of deep reinforcement learning – perhaps at a smaller scale initially – and deploying RL agents in financial markets is in fact the best way forward. It is widely known and agreed upon that deep neural networks can find patterns and signals superiorly compared to other methods. A possible path ahead is learning from mistakes and adjusting the models accordingly in an iterative manner. Additionally, proper limits and safeguards that restrict, e.g., the trading volumes (and thus potential damages) for black box methods is another possible remedy that would enable such methods to be used sooner. Furthermore, trading algorithms that are more interpretable than black boxes might also be more easily decipherable for other market participants than more advanced algorithms and can thus evade the issues of exploiting algorithmic traders discussed above.

We conclude this section with a brief discussion close to where we started it. We mentioned that financial markets are substantially more complex than a game of Go or Pong, which is true to perhaps an incomprehensible degree. Financial markets are typically non-stationary and second-order chaos systems, meaning that the systems themselves respond to predictions made about the system. Second-order chaos systems are significantly less predictable than first-order chaos systems (e.g., weather). Greg Jensen, Co-CIO of Bridgewater Associates, discussed similar aspects in Bloomberg's Odd Lots podcast:

> *"And to your point on what machine learning can help with and what it can't, at least in the current situation, is that when the data that you can plug into a machine learning model is representative of the data in the future, it can be very helpful. You have to have a lot of it and you have to have, but it has to be representative of the data in the future. What's so interesting about economies and markets is it never works that way because even just the existence of machine learning itself changes the future. So the future data points aren't going to be like the past data points because machine learning exists. And this is a game in which the players are affected by the tools. It's not like physics. It's not something physical where it doesn't matter if you're watching it. It matters completely that people are using machine learning techniques, make machine learning techniques themselves dangerous." [2]*

Whenever a financial actor finds an edge in markets, the edge will eventually fade away as the actor's trading will impact the market, and counterparties will adapt to the actor's new behavior. Additionally, other actors may observe, identify, and exploit the trading edge, leading the edge to disappear more quickly. Consequently, the system has changed in response to predictions about it, and the initial predictions of the system (or optimal behaviors in the system) become unusable. In situations where edges are quickly disappearing, large winnings may be reaped by actors that quickly can find new edges by, e.g., updating a sandbox environment of the new market conditions and employing an RL agent therein to find new edges or optimal behaviors. When new strategies have been found, the agent may be put to use in real markets again. Hultin et al. have developed a generative LOB model based on recurrent neural networks that can replicate

real order book data characteristics [41]. Notably, despite successes in creating realistic models of reality, a major challenge will remain in translating and converting an RL agent trained in a simulated environment into a real environment. In order for the agent to adapt to the real world, it has to be given a chance to train in it, something that most likely will result in short-term losses. Nevertheless, this is a risk one must be willing to take in order to improve current methods.

4.2 FUTURE WORK

Unfortunately, we did not have the time (primarily due to the long training times of the RL algorithms) to perform all the experiments we wanted. There are many such experiments. Here, we will briefly discuss a few areas that we consider particularly interesting and would have liked to explore further.

Distributional Reinforcement Learning

Distributional reinforcement learning is an area within RL that aims to learn a quantile function for the state-action return distribution rather than only its expectation (i.e., the value of Q). Dabney et al. introduced the notion of *implicit quantile networks* (IQN) in 2018, which they describe as *"a generally applicable, flexible, and state-of-the-art distributional variant of DQN."* The authors use quantile regression to learn the quantile function. IQN enables an agent to learn a large class of risk-sensitive policies, e.g., risk measures including expected shortfall (ES). [20] One reason it may be interesting to combine risk-sensitive RL and market making is the financial and risk-averse nature of market making – it may be of much greater interest for a market maker to appropriately limit the tail risk than focusing only on the mean reward. This is aligned with the currently prevailing paradigm in financial risk management; more and more focus is placed on limiting the tail risk through e.g., ES.[1] The Python package `PFRL` supports IQN and could, without much effort, be implemented in our LOB and learning environments. No published works combine distributional reinforcement learning and market making, so there is still plenty to explore here.[2]

Infinite Horizon

We have formulated the problem of market making as an *episodic task*; however, it would be interesting to frame it as a *continuing task*, i.e., a single episode of infinite length. The episodic setting may be preferable if one views the process of market making during a trading day (or some other finite unit of time) as a task, but in reality, a market maker may engage in the process indefinitely. To the best of our knowledge, formulating market making as a continuing task has not been attempted in any published papers. It is possible that market making is better formulated as an episodic task, which would explain why very few, perhaps no one, frame it in such a way. Nonetheless, we believe that it would be an interesting experiment. In addition to merely reframing the problem, it would be interesting to add some extra stochasticity every nth time step to simulate how the LOB and price process may jump when the market is closed. Adding a variable to the state space that indicates the time left until "market closure" would give the agent an indicator of when and/or if it would be a good idea to reduce the inventory to scale down the risk. Additionally, it would be interesting to simulate "a weekend" every fifth "trading session" where the movements can be even larger. We guess that the agent would learn to try to

[1] The Fundamental Review of the Trading Book (also known by its abbreviation, FRTB), is a set of financial guidelines that, among other things, promotes the use of the more tail-sensitive measure expected shortfall over value-at-risk which is worse at capturing tail risk.

[2] A search on Google Scholar with the query `"distributional reinforcement learning" + "market making"` yields three results per the writing of this, neither of which focuses on distributional reinforcement learning specifically in the setting of market making.

liquidate as much inventory as possible before a weekend, mainly if a risk-sensitive policy is being learned, reducing the downside.

Competitive Multi-Agent Setting

Some previous works employ multi-agent reinforcement learning in market making where competing market making agents interact in an environment, e.g., [26, 58, 65]. We think that in addition to this, it would be interesting to study a multi-agent setting not only with competing market makers but also with RL agents with entirely different objectives, such as in the traditional order execution problem. Such a setting would be able to better simulate real-world markets where participants typically do not share objectives more closely; however, this would entail a great challenge in choosing the agents such that their aggregate replicates reality. A market making agent in such a setting would more likely learn to defend against predatory trading tactics, something the agents in this thesis have no experience with due to the static modeling of the market.

5 CONCLUSION

The process of simultaneously and continuously providing buy and sell prices in a financial asset – market making – is rather complicated to optimize. Traditionally, attempts to optimize market making have prevailingly been of an analytical character and usually involve stochastic differential equations. In this thesis, reinforcement learning has been applied to infer optimal market making strategies, a relatively uncharted and novel research area. Moreover, since most published articles in the field are notably opaque concerning most aspects, extra effort has been put into sharing methods, parameters, and results.

The main goal of this thesis has been to answer our two research questions (**RQ1** and **RQ2**). The first question regarded comparing analytical and reinforcement learning strategies for market making while the second question tackled comparing tabular and deep reinforcement learning methods in the context of market making.

In order to explore the topic and answer the research questions, we first used a simple probabilistic model of a limit order book to compare analytical and RL-derived strategies. Subsequently, we used a Markov chain model of a limit order book to train an agent to make markets optimally using tabular Q-learning and deep reinforcement learning with double deep Q-learning. Finally, we analyzed, compared, and discussed the results and strategies.

Related to **RQ1**, we have found that reinforcement learning can be used to outperform analytically derived strategies. We have also concluded that reinforcement learning may be particularly useful for more advanced models and in actual markets as the techniques do not require one to make naive and simplistic model assumptions that are not representative of reality.

Next, answering **RQ2**, we have observed that deep reinforcement learning methods generally can outperform non-deep methods with market making in regards to most aspects. This finding is not particularly surprising, as we significantly extended the state space in the experiments where deep methods were employed. Furthermore, tabular methods (Q-learning in our case) do not generalize in any manner – a significant drawback. However, it should be noted that deep methods (DDQN in our case) are susceptible to substantial needs for (hyper)parameter-tuning. Finding appropriate parameter values is a very time-consuming procedure.

We also addressed implementation aspects of reinforcement learning in market making and concluded that the current techniques are insufficient and far from being put into production as entirely autonomous trading vehicles, albeit very promising.

Lastly, we have proposed some exciting extensions and directions for future work in this research field. We discuss implementing distributional reinforcement learning methods, reformulating market making as an infinite horizon task, and employing multi-agent algorithms to allow multiple market makers to compete with each other and other RL-trained financial actors.

REFERENCES

1. Aldridge, I. *High-frequency trading: a practical guide to algorithmic strategies and trading systems* (John Wiley & Sons, 2013).
2. Alloway, T. & Weisenthal, J. *Transcript: Bridgewater's Greg Jensen on Why Markets Have Further to Fall* May 2022. https://www.bloomberg.com/news/articles/2022-05-23/transcript-bridgewater-s-greg-jensen-on-why-markets-have-further-to-fall.
3. *AlphaGo - The Movie* YouTube. Mar. 2020. https://www.youtube.com/watch?v=WXuK6gekU1Y.
4. Anschel, O., Baram, N. & Shimkin, N. *Averaged-dqn: Variance reduction and stabilization for deep reinforcement learning* in *International conference on machine learning* (2017), 176–185.
5. Arulkumaran, K., Deisenroth, M. P., Brundage, M. & Bharath, A. A. Deep reinforcement learning: A brief survey. *IEEE Signal Processing Magazine* **34,** 26–38 (2017).
6. Avellaneda, M. & Stoikov, S. High-frequency trading in a limit order book. *Quantitative Finance* **8,** 217–224 (2008).
7. Baldacci, B., Manziuk, I., Mastrolia, T. & Rosenbaum, M. Market making and incentives design in the presence of a dark pool: a deep reinforcement learning approach. *arXiv preprint arXiv:1912.01129* (2019).
8. Battalio, R., Jennings, R. & McDonald, B. Deviations from time priority on the NYSE. *Journal of Financial Markets* **53,** 100567 (2021).
9. Blom, G. *Sannolikhetsteori och statistikteori med tillämpningar* (Studentlitteratur, 1980).
10. Brockman, G. *et al.* Openai gym. *arXiv preprint arXiv:1606.01540* (2016).
11. Cartea, Á., Jaimungal, S. & Penalva, J. *Algorithmic and high-frequency trading* (Cambridge University Press, 2015).
12. Chan, N. T. & Shelton, C. An electronic market-maker (2001).
13. Church, G. & Cliff, D. *A simulator for studying automated block trading on a coupled dark/lit financial exchange with reputation tracking* in *European Modeling & Simulation Symposium* (2019), 284–293.
14. Collins, R. A. CARA Utility and Gain Aversion (2006).
15. Cont, R., Kukanov, A. & Stoikov, S. The price impact of order book events. *Journal of financial econometrics* **12,** 47–88 (2014).

16. Cont, R., Stoikov, S. & Talreja, R. A stochastic model for order book dynamics. *Operations research* **58,** 549–563 (2010).

17. Council of European Union. *Council Directive (EU) no 65/2014* https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:32014L0065. 2014.

18. Council of European Union. *Commission Delegated Regulation (EU) 2017/575* https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:32017R0578&from=MT. 2017.

19. Crisafi, M. A. & Macrina, A. Simultaneous trading in 'Lit'and dark pools. *International Journal of Theoretical and Applied Finance* **19,** 1650055 (2016).

20. Dabney, W., Ostrovski, G., Silver, D. & Munos, R. *Implicit quantile networks for distributional reinforcement learning* in *International conference on machine learning* (2018), 1096–1105.

21. Dixon, M. F. & Halperin, I. The four horsemen of machine learning in finance. *Available at SSRN 3453564* (2019).

22. Friedman, D. *The double auction market: institutions, theories, and evidence* (Routledge, 2018).

23. Fujita, Y., Nagarajan, P., Kataoka, T. & Ishikawa, T. ChainerRL: A Deep Reinforcement Learning Library. *Journal of Machine Learning Research* **22,** 1–14. http://jmlr.org/papers/v22/20-376.html (2021).

24. Ganesh, S. *et al.* Reinforcement learning for market making in a multi-agent dealer market. *arXiv preprint arXiv:1911.05892* (2019).

25. Gašperov, B., Begušić, S., Posedel Šimović, P. & Kostanjčar, Z. Reinforcement Learning Approaches to Optimal Market Making. *Mathematics* **9,** 2689 (2021).

26. Gašperov, B. & Kostanjčar, Z. Market making with signals through deep reinforcement learning. *IEEE Access* **9,** 61611–61622 (2021).

27. Gatheral, J. No-dynamic-arbitrage and market impact. *Quantitative finance* **10,** 749–759 (2010).

28. Glosten, L. R. & Milgrom, P. R. Bid, ask and transaction prices in a specialist market with heterogeneously informed traders. *Journal of financial economics* **14,** 71–100 (1985).

29. Gould, M. D. *et al.* Limit order books. *Quantitative Finance* **13,** 1709–1742 (2013).

30. Guéant, O. Optimal market making. *Applied Mathematical Finance* **24,** 112–154 (2017).

31. Haider, A., Hawe, G., Wang, H. & Scotney, B. Gaussian based non-linear function approximation for reinforcement learning. *SN Computer Science* **2,** 1–12 (2021).

32. Haider, A., Wang, H., Scotney, B. & Hawe, G. *Effect of market spread over reinforcement learning based market maker* in *International Conference on Machine Learning, Optimization, and Data Science* (2019), 143–153.

33. Hart, A. G., Olding, K. R., Cox, A. M., Isupova, O. & Dawes, J. H. Echo State Networks for Reinforcement Learning. *arXiv preprint arXiv:2102.06258* (2021).

34. Hasselt, H. Double Q-learning. *Advances in neural information processing systems* **23** (2010).

35. Henderson, P. *et al. Deep reinforcement learning that matters* in *Proceedings of the AAAI conference on artificial intelligence* **32** (2018).

36. Ho, T. & Stoll, H. R. Optimal dealer pricing under transactions and return uncertainty. *Journal of Financial economics* **9,** 47–73 (1981).

37. Huberman, G. & Stanzl, W. Price manipulation and quasi-arbitrage. *Econometrica* **72,** 1247–1275 (2004).

38. Hult, H. *SF2957 Statistical Machine Learning, Lecture Notes* (Royal Institute of Technology, 2018).

39. Hult, H. & Kiessling, J. Algorithmic trading with Markov chains. https://www.researchgate.net/publication/268032734_ALGORITHMIC_TRADING_WITH_MARKOV_CHAINS (2010).

40. Hult, H., Lindskog, F., Hammarlid, O. & Rehn, C. J. *Risk and portfolio analysis: Principles and methods* (Springer Science & Business Media, 2012).

41. Hultin, H., Hult, H., Proutiere, A., Samama, S. & Tarighati, A. A generative model of a limit order book using recurrent neural networks (2021).

42. Ioffe, S. & Szegedy, C. *Batch normalization: Accelerating deep network training by reducing internal covariate shift* in *International conference on machine learning* (2015), 448–456.

43. Izenman, A. J. Modern multivariate statistical techniques. *Regression, classification and manifold learning* **10,** 978– (2008).

44. Kessous, A. D. *The risk and reward of more dark pool trading* Dec. 2021. https://www.nasdaq.com/articles/the-risk-and-reward-of-more-dark-pool-trading.

45. Kim, A. J., Shelton, C. R. & Poggio, T. Modeling stock order flows and learning market-making from data (2002).

46. Kingma, D. P. & Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

47. Kushner, H. & Yin, G. in *Stochastic Modelling and Applied Probability* (Springer-Verlag NY, 2003).

48. Lim, Y.-S. & Gorse, D. *Reinforcement learning for high-frequency market making* in *ESANN 2018-Proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning* (2018), 521–526.

49. Liu, C. *Deep Reinforcement Learning and Electronic Market Making* PhD thesis (Imperial College London London, UK, 2020).

50. Liu, Q., Chung, A., Szepesvári, C. & Jin, C. When Is Partially Observable Reinforcement Learning Not Scary? *arXiv preprint arXiv:2204.08967* (2022).

51. Lokhacheva, K., Parfenov, D. & Bolodurina, I. Reinforcement Learning Approach for Market-Maker Problem Solution. *International Session on Factors of Regional Extensive Development (FRED 2019),* 256–260 (2020).

52. Majeed, S. J. & Hutter, M. *On Q-learning Convergence for Non-Markov Decision Processes.* in *IJCAI* **18** (2018), 2546–2552.

53. Mani, M., Phelps, S. & Parsons, S. *Applications of Reinforcement Learning in Automated Market-Making* in *Proceedings of the GAIW: Games, Agents and Incentives Workshops, Montreal, Canada* (2019), 13–14.

54. Minsky, M. Steps toward artificial intelligence. *Proceedings of the IRE* **49,** 8–30 (1961).

55. Mnih, V. *et al.* Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602* (2013).

56. Navarro, D. *Learning statistics with R* (2015).

57. Ormeron, D. Lee Sedol defeats AlphaGo in masterful comeback - game 4. *Go Game Guru.* https://web.archive.org/web/20161116082508/https://gogameguru.com/lee-sedol-defeats-alphago-masterful-comeback-game-4/ (Mar. 2016).

58. Patel, Y. Optimizing market making using multi-agent reinforcement learning. *arXiv preprint arXiv:1812.10252* (2018).

59. Peer, O., Tessler, C., Merlis, N. & Meir, R. *Ensemble bootstrapping for Q-Learning* in *International Conference on Machine Learning* (2021), 8454–8463.

60. Sadighian, J. Deep reinforcement learning in cryptocurrency market making. *arXiv preprint arXiv:1911.08647* (2019).

61. Sadighian, J. Extending Deep Reinforcement Learning Frameworks in Cryptocurrency Market Making. *arXiv preprint arXiv:2004.06985* (2020).

62. Selser, M., Kreiner, J. & Maurette, M. Optimal Market Making by Reinforcement Learning. *arXiv preprint arXiv:2104.04036* (2021).

63. Silver, D. *et al.* Mastering the game of Go with deep neural networks and tree search. *Nature* **529,** 484–489 (2016).

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

65. Spooner, T. & Savani, R. Robust market making via adversarial reinforcement learning. *arXiv preprint arXiv:2003.01820* (2020).

66. Stoikov, S. & Sağlam, M. Option market making under inventory risk. *Review of Derivatives Research* **12,** 55–79 (2009).

67. Surane, J. & Barnert, J.-P. Citi Trader Made Error Behind Flash Crash in Europe Stocks. *Bloomberg.* https://www.bloomberg.com/news/articles/2022-05-02/citi-s-london-trading-desk-behind-rare-european-flash-crash (May 2022).

68. Sutton, R. S. & Barto, A. G. *Reinforcement learning: An introduction* (MIT press, 2018).

69. Tversky, A. A critique of expected utility theory: Descriptive and normative considerations, 163–173 (1975).

70. Tversky, A. & Kahneman, D. in *Behavioral decision making* 25–41 (Springer, 1985).

71. Tversky, A. & Kahneman, D. Loss aversion in riskless choice: A reference-dependent model. *The quarterly journal of economics* **106,** 1039–1061 (1991).

72. Van Hasselt, H., Guez, A. & Silver, D. *Deep reinforcement learning with double q-learning* in *Proceedings of the AAAI conference on artificial intelligence* **30** (2016).

73. Wang, Y. *Electronic Market Making on Large Tick Assets* PhD thesis (The Chinese University of Hong Kong (Hong Kong), 2019).

74. Ward, A. Norwegian traders charged with manipulation. *Financial Times.* https://www.ft.com/content/edcaf806-abbb-11df-9f02-00144feabdc0 (Aug. 2010).

75. Verity, A. & Lawrie, E. Hound of hounslow: Who is Navinder Sarao, the 'flash crash trader'? *BBC News.* https://www.bbc.com/news/explainers-51265169 (Jan. 2020).

76. Wiering, M. A. & Van Hasselt, H. Ensemble algorithms in reinforcement learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* **38,** 930–936 (2008).

77. Wu, P., Rambaldi, M., Muzy, J.-F. & Bacry, E. Queue-reactive Hawkes models for the order flow. *arXiv preprint arXiv:1901.08938* (2019).

78. Xu, K., Gould, M. D. & Howison, S. D. Multi-level order-flow imbalance in a limit order book. *Market Microstructure and Liquidity* **4,** 1950011 (2018).

79. Zhang, G. & Chen, Y. Reinforcement learning for optimal market making with the presence of rebate. *Available at SSRN 3646753* (2020).

80. Zhong, Y., Bergstrom, Y. & Ward, A. *Data-Driven Market-Making via Model-Free Learning.* in *IJCAI* (2020), 4461–4468.

81. Zuckerman, G. *The man who solved the market: how Jim Simons launched the quant revolution* (Penguin, 2019).

A

DERIVATIONS

A.1 DERIVATION OF ANALYTICALLY OPTIMAL SOLUTION OF SIMPLE PROBABILISTIC MODEL

The performance criterion is defined as

$$H^{\delta}(t,x,S,q) = \mathbb{E}_{t,x,S,q}\left[X_T + Q_T(S_T - \alpha Q_T) - \phi \int_t^T (Q_u)^2 du\right], \tag{80}$$

where the final term represents the running inventory term. S_t is the mid price, X_t is the MM's cash process, Q_t the MM's inventory process, $\phi \geq 0$ is the parameter of the running inventory penalty and $\alpha \geq 0$ is the fee of taking liquidity. For the full definition of the SPM, see Section 2.2.3.1. The market maker's (optimal) value function is thus

$$H(t,x,S,q) = \sup_{\delta^{\pm} \in \mathcal{A}} H^{\delta}(t,x,S,q). \tag{81}$$

Here, $\mathcal{A}$ is the set of admissible strategies. By results in analysis of stochastic differential equations, it holds that the value function solves the following dynamic programming equation.

$$\begin{aligned} 0 =& \partial_t H + \frac{1}{2}\sigma^2 \partial_{SS} H - \phi q^2 \\ &+ \lambda^+ \sup_{\delta^+} \Big\{ \exp\{-\kappa^+ \delta^+\} \big(H(t, x + (S + \delta^+), q - 1, S\big) - H \Big\} \mathbb{1}_{q > \underline{q}} \\ &+ \lambda^- \sup_{\delta^-} \Big\{ \exp\{-\kappa^- \delta^-\} \big(H(t, x - (S - \delta^-), q + 1, S\big) - H \Big\} \mathbb{1}_{q < \overline{q}} \end{aligned} \tag{82}$$

where the terminal condition is

$$H(T,x,S,q) = x + q(S - \alpha q). \tag{83}$$

Cartea et al. make the following ansatz for H:

$$H(t,x,q,S) = x + qS + h(t,q), \tag{84}$$

with which an optimal solution can be obtained. Setting $\overline{q} > 0$ and $\underline{q} < 0$ as the maximum absolute long and short inventory volumes allowed, we get

$$\delta^{+,*}(t,q) = \begin{cases} \frac{1}{\kappa^+} - h(t,q-1) + h(t,q), & \text{if } q \neq \underline{q} \\ +\infty, & \text{if } q = \underline{q} \end{cases} \tag{85}$$

$$\delta^{-,*}(t,q) = \begin{cases} \frac{1}{\kappa^-} - h(t,q-1) + h(t,q), & \text{if } q \neq \overline{q} \\ +\infty, & \text{if } q = \overline{q} \end{cases} \tag{86}$$

If we let the fill probabilities of buy and sell LOs be the same ($\lambda^+ = \lambda^-$), and $\kappa_+ = \kappa_- = \kappa$, then

$$h(t,q) = \frac{1}{\kappa}\log\omega(t,q), \tag{87}$$

with

$$\boldsymbol{\omega}(t) = [\omega(t,\underline{q}), \omega(t,\underline{q}+1), \ldots, \omega(t,\overline{q})]^\top$$

Finally, when the $(\overline{q}-\underline{q}+1)$-square matrix $\boldsymbol{A}$ is defined as

$$\boldsymbol{A}_{i,q} = \begin{cases} -\phi\kappa q^2, & \text{if } i = q, \\ \lambda^+ e^{-1}, & \text{if } i = q-1, \\ \lambda^- e^{-1}, & \text{if } i = q+1, \\ 0, & \text{otherwise,} \end{cases} \tag{88}$$

the solution is given by

$$\boldsymbol{\omega}(t) = e^{\boldsymbol{A}(T-t)}\boldsymbol{z}, \tag{89}$$

where $\boldsymbol{z}$ is a $(\overline{q}-\underline{q}+1)$-vector where the components are given by $z_j = e^{-\alpha\kappa j^2}$, $j = \underline{q}, \ldots, \overline{q}$. This derivation is entirely based on [11].

B

ADDITIONAL RESULTS

This appendix contains plots, graphs, and tables, that are not central to the results but nonetheless provide value to some readers.

B.1 HYPERPARAMETER SCHEMES OF SIMPLE PROBABILISTIC MODEL

The schemes for the hyperparameters – the exploration rate of the ϵ-greedy policy, the exploration rate of the exploring starts, and the learning rate – used for Q-learning in the SPM are graphically illustrated in Figure B.1.

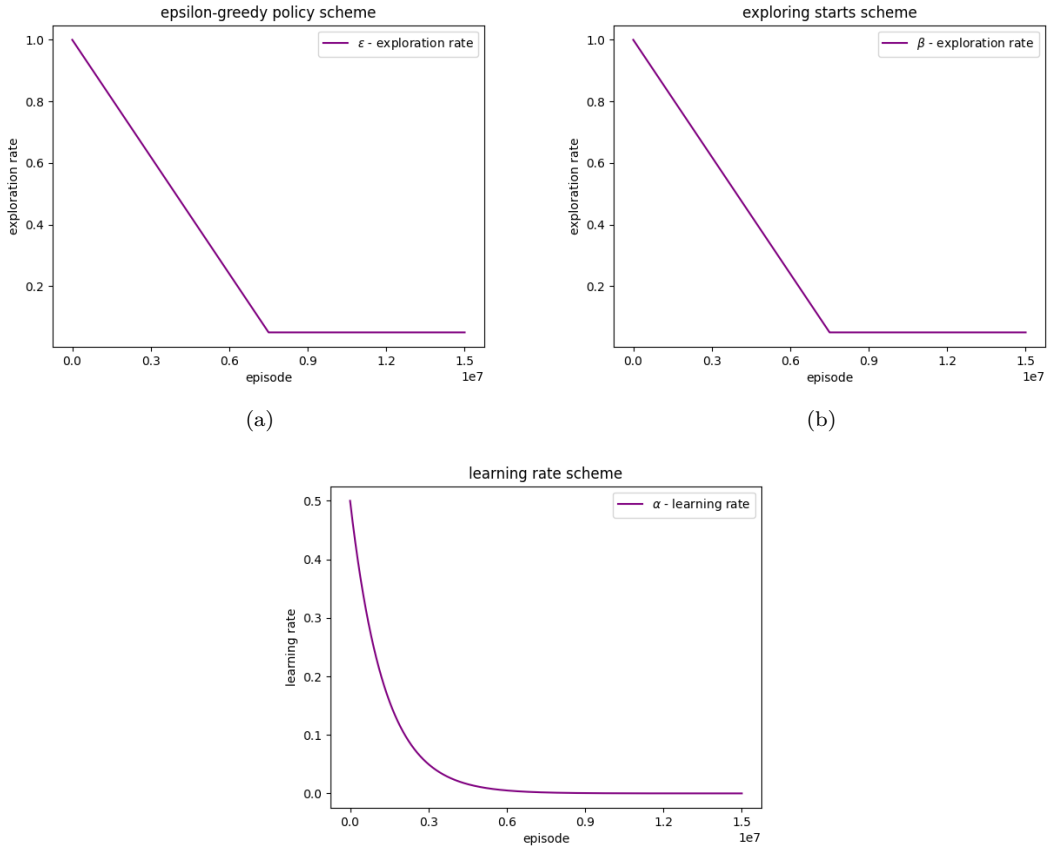
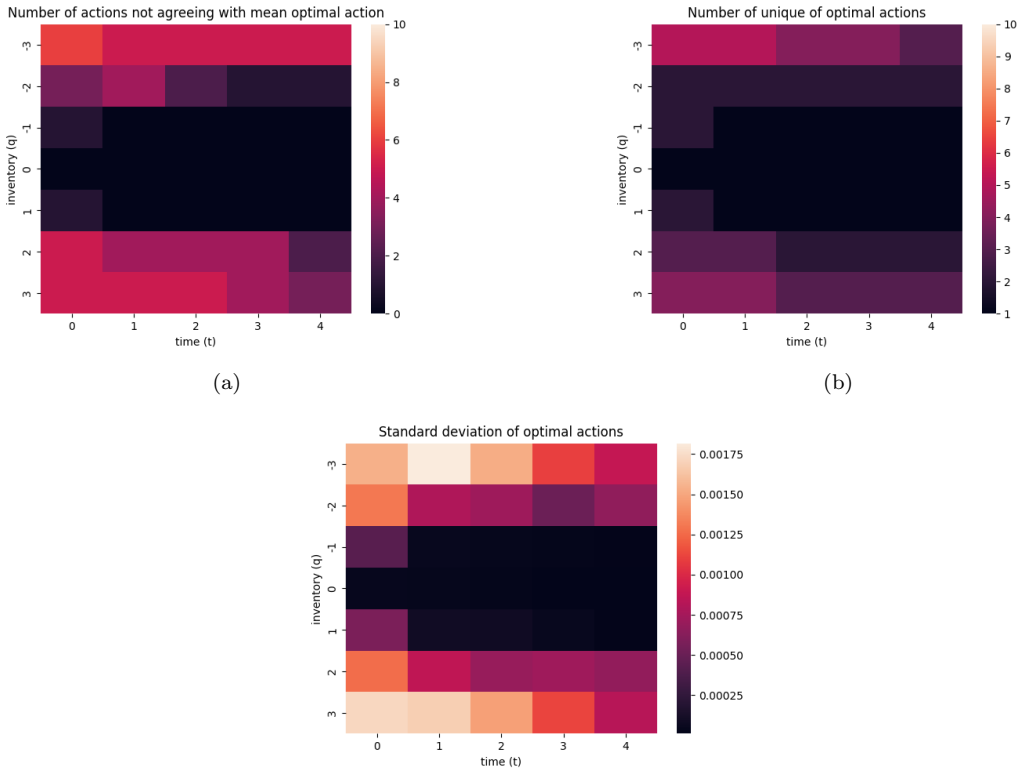


Figure B.1: The schemes of **(a)** the ϵ-greedy policy, **(b)** the exploring starts, and **(c)** the learning rate.

B.2 STABILITY OF Q-LEARNING STRATEGIES ON SIMPLE PROBABILISTIC MODEL

To look further into the stability of the strategies, we have three graphs in Figure B.2 that are useful. Figure B.2a shows the number of runs not agreeing with the optimal actions of the mean strategy and Figure B.2b shows the number of unique optimal actions for every state over all the runs. These two plots have a very similar pattern, showing great stability for the state $(t, Q_t) = (0, 0)$ as well as states closer to the end of the episode with lower absolute volume. The same pattern can be seen in Figure B.2c which shows the standard deviation of the state-values of the different runs based on the averaged Q-table. This pattern is not surprising; at the end of the episode, fewer rewards are to be received (since the episode is over soon), which will make the convergence faster. The initial state being more stable is due to the decreasing rate of exploration of exploring starts – it will be visited the most during training. One should also note that one "half" of the action – the bid price (or ask price) – at $Q_t = \overline{q}$ ($Q_t = \underline{q}$) is very unstable in nature since the chance of a trade being executed on that side of the order book is very rare (it has to perfectly even out with an opposite order).



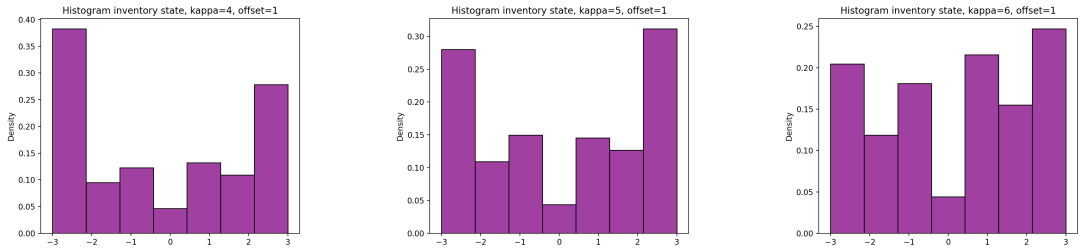
(a) (b)

(c)

Figure B.2:
(a) the number of actions not agreeing with the mean strategy
(b) the number of unique optimal actions
(c) the average standard deviation of the action-value of the optimal action according to the mean strategy.

B.3 FINDING A SUITABLE INVENTORY-GROUPING

An empirical investigation of the running inventory of a market maker employing the fixed offset strategy with $\delta^{\pm} = 1, 2$ supports our choice of $\kappa = 5$ when $T = 1000$, see the distribution in Figure B.3. Recall that κ was used to define the inventory state space variable; it represented the size of the inventory buckets in the LH-Q setting. Considering the various distributions in the figure, $\kappa = 5$ seems like a good choice as the aggregated distribution (across the different order depths) seems quite even, much more so than for the other values of κ. It should be noted that this choice of $\kappa = 5$ is fairly arbitrary and that it may negatively affect the performance of the learning of an optimal strategy. The time $T = 1000$ and the number of episodes is 100 (which explains the asymmetry/variability observable in the figure). For the model experiments where $T = 25$, we adjusted the value of κ downwards slightly to $\kappa = 2$. The choice was based on a similar analysis to the above.



(a) $\kappa = 4$, $\delta^{\pm} = 1$ (b) $\kappa = 5$, $\delta^{\pm} = 1$ (c) $\kappa = 6$, $\delta^{\pm} = 1$

(d) $\kappa = 4$, $\delta^{\pm} = 2$ (e) $\kappa = 5$, $\delta^{\pm} = 2$ (f) $\kappa = 6$, $\delta^{\pm} = 2$

Figure B.3: Distribution of inventory state space variable Q_t for select values of κ and $\delta^{\pm} = 1, 2$. Here, $T = 1000$.

B.4 STABILITY OF Q-LEARNING STRATEGIES IN THE SH-Q SETTING

Digging deeper into the stability of the Q-learning in the SH-Q setting, we created three plots presented in Figure B.4. Figure B.4a shows that none of the individual Q-learning strategies agree with the mean strategy on which action is optimal. Figure B.4b tells a somewhat similar story; however, some of the individual strategies agree on some of the states. Finally, the standard deviations in Figure B.4c are all larger than for the SPM (see Figure B.2c) but significantly smaller than for the time-grouped model (see Figure B.5c).

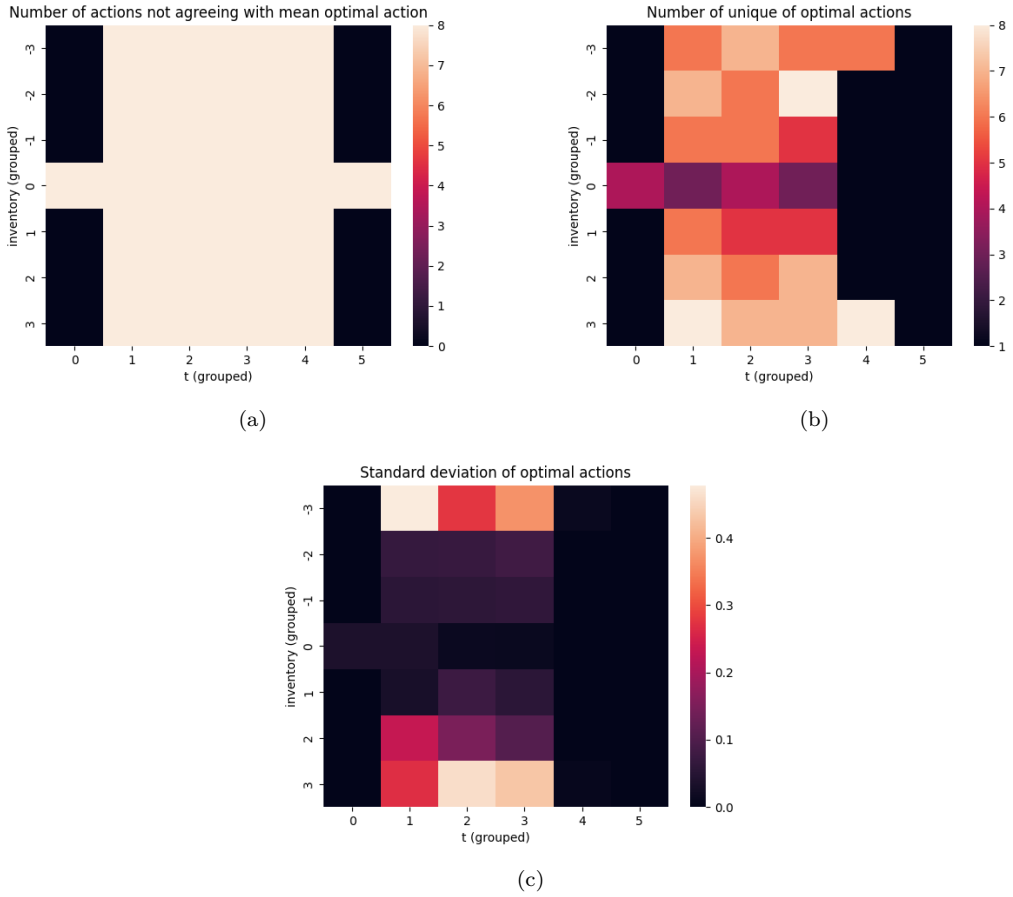


Figure B.4:
Stability of Q-learning Strategies in the SH-Q setting
(a) the number of actions not agreeing with the mean strategy
(b) the number of unique optimal actions
(c) the average standard deviation of the action-value of the optimal action according to the mean strategy.

B.5 STABILITY OF Q-LEARNING STRATEGIES IN THE LH-Q SETTING

Digging deeper into the stability of the Q-learning in the LH-Q setting, we created three plots presented in Figure B.5. Figure B.5a is very telling; none of the individual Q-learning strategies agree with the mean strategy on which action is optimal. Figure B.5b tells a similar story; the state with the least amount of unique actions has three unique actions, and the average is around five. Finally, the standard deviations in Figure B.5c are all in the thousands. We conclude that the training on this model has not converged.

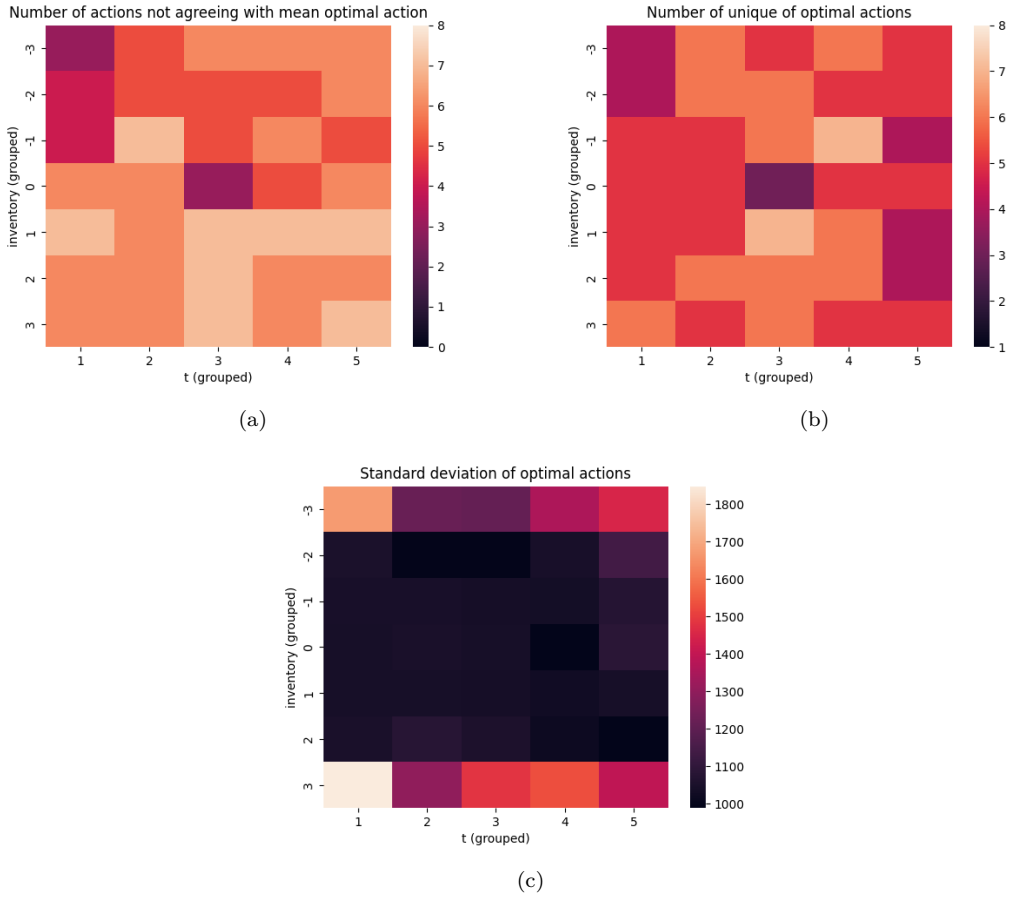


Figure B.5:
Stability of Q-learning Strategies in the LH-Q setting
(a) the number of actions not agreeing with the mean strategy
(b) the number of unique optimal actions
(c) the average standard deviation of the action-value of the optimal action according to the mean strategy.

B.6 TABLE OF REWARDS OF DDQN RUNS IN THE SH-DDQN NEUTRAL SETTING

The results from all eight individual runs in the SH-DDQN setting with neutral LOB resets are displayed in Table B.1. In the SH-Q setting (see Table 3.9), only one strategy was money-losing. Au contraire, five of the eight strategies obtained in the SH-DDQN setting with neutral LOB resets are not able to turn a profit on average. Remarkably though, the average strategies from Q-learning and DDQN are highly comparable, albeit with a somewhat higher standard deviation for the DDQN average strategy (compare Tables 3.10 and B.2).

Run	$\overline{r}$	σ_r	$\overline{r}$ **per action**	$\overline{r}$ **per second**	$v_*(0,0)$
1	-2.46×10^{-5}	1.16×10^{-2}	-4.92×10^{-6}	-9.84×10^{-7}	1.21×10^{-4}
2	$\mathbf{7.38 \times 10^{-5}}$	$\mathbf{5.27 \times 10^{-3}}$	$\mathbf{1.47 \times 10^{-5}}$	$\mathbf{2.95 \times 10^{-6}}$	5.50×10^{-5}
3	-1.33×10^{-4}	1.52×10^{-2}	-2.66×10^{-5}	-5.31×10^{-6}	1.11×10^{-4}
4	-4.40×10^{-5}	1.30×10^{-2}	-8.80×10^{-6}	-1.76×10^{-6}	9.21×10^{-5}
5	-4.20×10^{-6}	1.26×10^{-2}	-8.40×10^{-7}	-1.68×10^{-7}	$\mathbf{9.51 \times 10^{-5}}$
6	-1.30×10^{-5}	6.01×10^{-3}	-2.60×10^{-6}	-5.20×10^{-7}	6.76×10^{-5}
7	1.34×10^{-5}	1.57×10^{-2}	2.68×10^{-6}	5.36×10^{-7}	1.20×10^{-4}
8	2.66×10^{-5}	1.61×10^{-2}	5.32×10^{-6}	1.06×10^{-6}	1.80×10^{-4}

Table B.1: The average reward received when following the obtained strategies from DDQN in the SH-DDQN with neutral resets for 5×10^4 episodes. Bold values highlight the best result: the highest reward or the lowest standard deviation.

B.7 BOXPLOT OF REWARDS OF DDQN RUNS IN THE SH-DDQN NEUTRAL SETTING

Figure B.6 visualizes the results of the agents trained in the SH-DDQN setting with neutral LOB resets. The rewards received when following the eight different strategies for 5×10^4 episodes are shown in a boxplot.

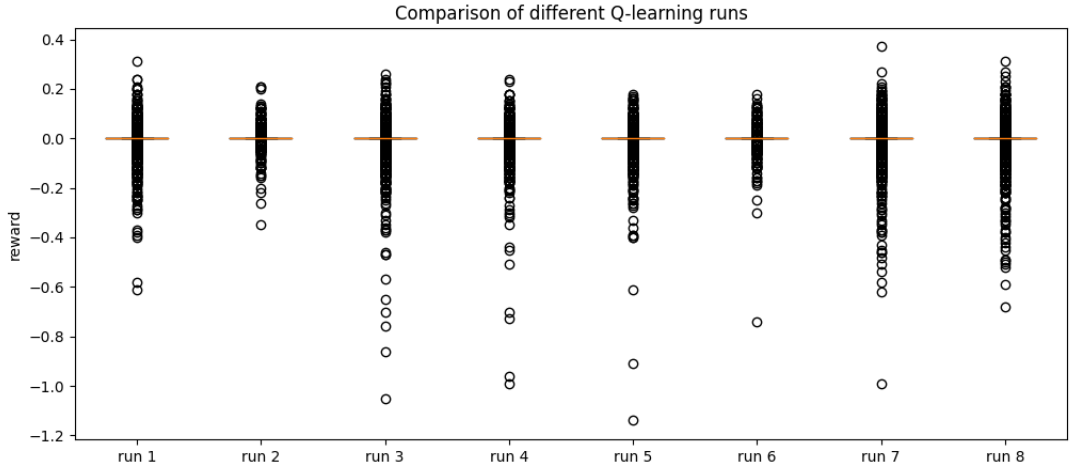


Figure B.6: Boxplot of the rewards of eight different DDQN runs in the SH-DDQN setting with neutral LOB when run for 5×10^4 episodes.

B.8 AVERAGE REWARD, STATE-VALUE, AND LOSS DURING TRAINING IN THE SH-DDQN NEUTRAL SETTING

Figure B.7 plots the average reward, the state-value at $(t, Q_t) = (0, 0)$, and the loss during training of the agents trained in the SH-DDQN setting with neutral LOB resets.

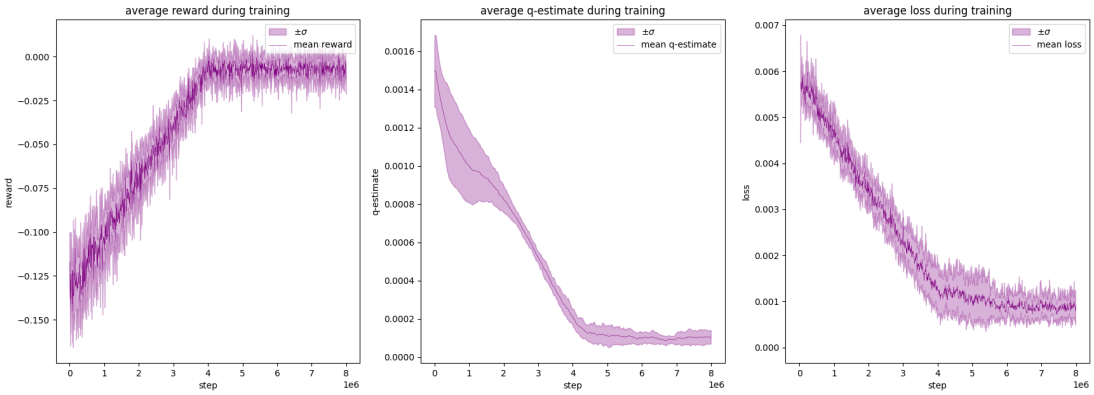
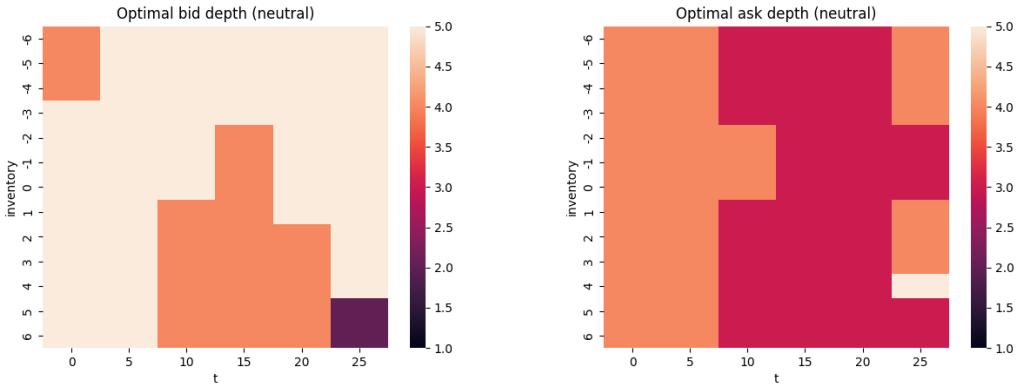


Figure B.7:
Training in the SH-DDQN setting with neutral LOB for 8×10^6 steps.
Left: the average reward during training with a shaded area showing ± one standard deviation.
Middle: the state-value at $(t, Q_t) = (0, 0)$ and $Q_t = 0$ during training with a shaded area showing ± one standard deviation.
Right: the average loss during training with a shaded area showing ± one standard deviation.

B.9 THE MEAN STRATEGY IN THE SH-DDQN NEUTRAL SETTING

If we study the mean strategy (in the SH-DDQN setting with neutral LOB resets) in finer detail, we find a not particularly comprehensible strategy. In Figure B.8, the mean strategy is presented. The bid depth is opposite to what is expected, showing a willingness to increase the inventory when it is already high. However, no particular gradient is shown for the ask depth.



(a) (b)

Figure B.8: These two panels are obtained by averaging the action-values from the eight agents trained in the SH-DDQN with neutral LOB. **(a)** shows a heat map of the optimal bid depths, and **(b)** shows a heat map of the optimal ask depths.

B.10 REWARDS OF THE DDQN AND BENCHMARK STRATEGIES IN THE SH-DDQN NEUTRAL SETTING

Table B.2 summarizes the results of the agents trained in the SH-DDQN setting with neutral LOB resets as well as the benchmarking strategies. From this, it is evident that the DDQN strategies severely outperform the benchmarking strategies, both in terms of average reward and standard deviation.

Strategy	$\overline{r}$	σ_r
DDQN		
- *best run*	$\mathbf{7.16 \times 10^{-5}}$	$\mathbf{6.45 \times 10^{-3}}$
- *mean strategy*	2.32×10^{-5}	9.34×10^{-3}
Benchmark		
- *constant*	-1.15×10^{-2}	6.95×10^{-2}
- *random*	-6.45×10^{-3}	4.21×10^{-2}

Table B.2: The average reward received when following different strategies in the SH-DDQN with neutral resets for 5×10^4 episodes. Bold values highlight the best result: the highest reward or the lowest standard deviation.

B.11 BOXPLOT OF DDQN AND BENCHMARKING STRATEGIES IN THE SH-DDQN NEUTRAL SETTING

Figure B.9 visualizes the results of the agents trained in the SH-DDQN setting with neutral LOB resets versus the benchmarking strategies. The rewards received when following the best single agent's strategy, the mean strategy, and the benchmarking strategies for 5×10^4 episodes are shown in a boxplot.

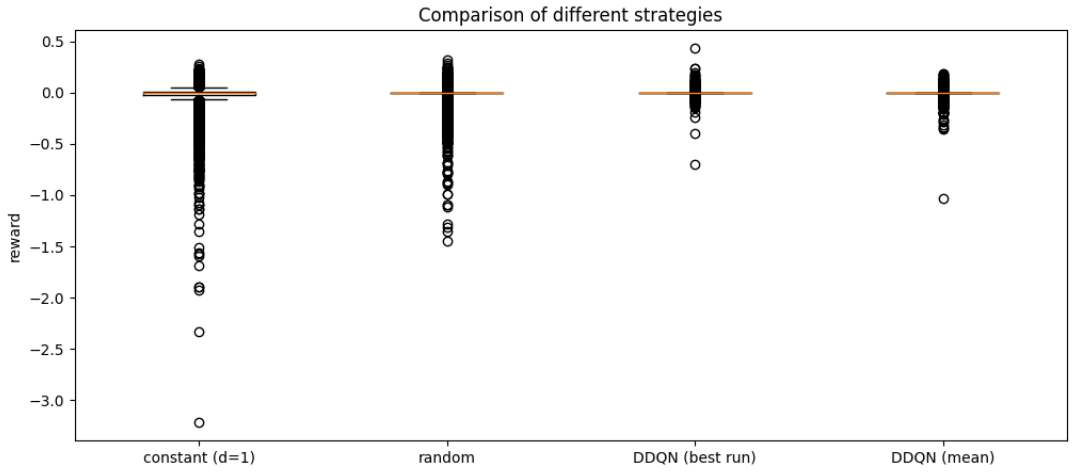


Figure B.9: Boxplot of the reward of different strategies in the SH-DDQN setting with neutral LOB when run for 5×10^4 episodes.

B.12 VISUALIZATION OF STRATEGIES IN THE SH-DDQN NEUTRAL SETTING

The resulting inventory, cash, and value processes of the mean strategy and the individual runs in the SH-DDQN setting with neutral LOB resets are illustrated in Figure B.10 and Figure B.11. Figure B.11 illustrates that there is no particular skewing that is beneficial for higher rewards, but most of the runs choose a short position.

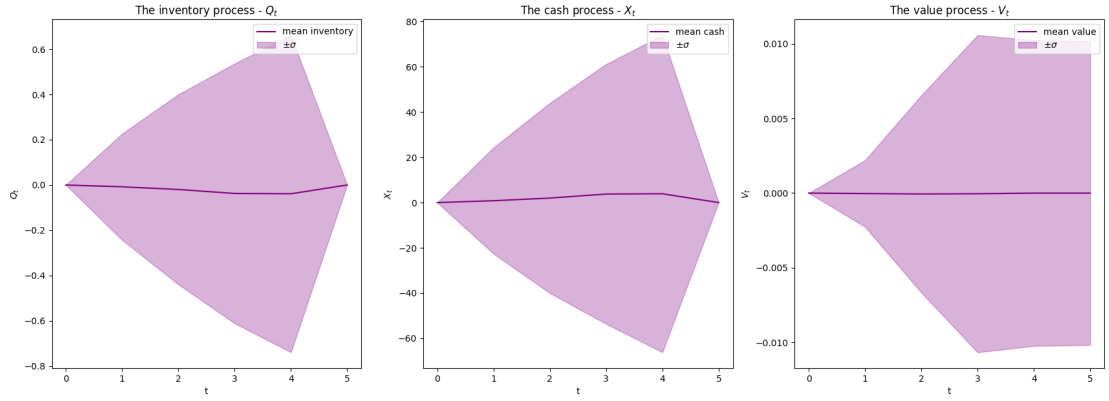


Figure B.10: The average of the inventory, cash process, and value processes when following the mean strategy obtained from training in the SH-DDQN setting with neutral LOB. Also showing ± one standard deviation in the shaded area. Evaluated for 10^4 episodes.

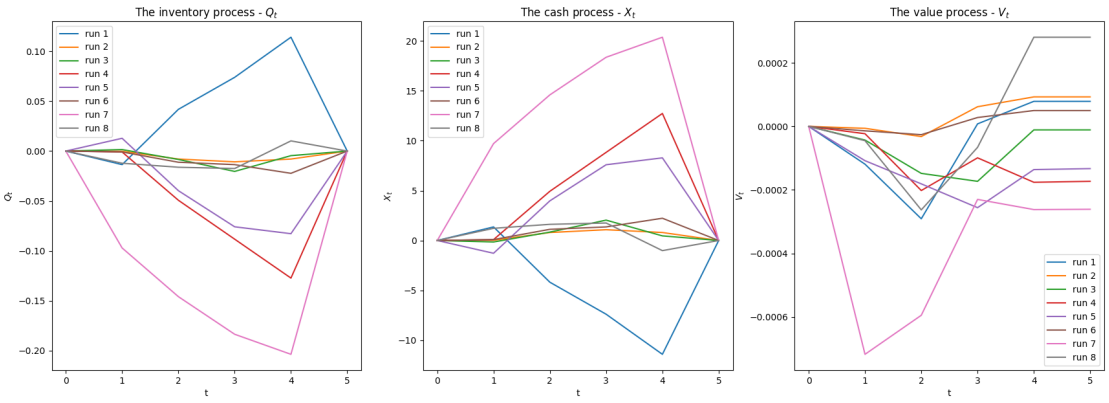


Figure B.11: The average of the inventory, cash process, and value processes for all eight individual runs obtained from training in the SH-DDQN setting with neutral LOB. Evaluated for 10^4 episodes.

B.13 STABILITY OF DDQN STRATEGIES IN THE SH-DDQN SETTING

Digging deeper into the stability of the DDQN in the SH-DDQN setting, we created three plots presented in Figure B.12. Figure B.12a shows that almost none of the individual DDQN strategies agree with the mean strategy on which action is optimal. Figure B.12b tells a somewhat similar story, however, there seems to be some more agreement. Finally, the standard deviations in Figure B.12c are all quite small, but significantly larger than the expected reward.

In all figures we also see a gradient in the direction of the time, showing that the agents agree more at the beginning of the episode.

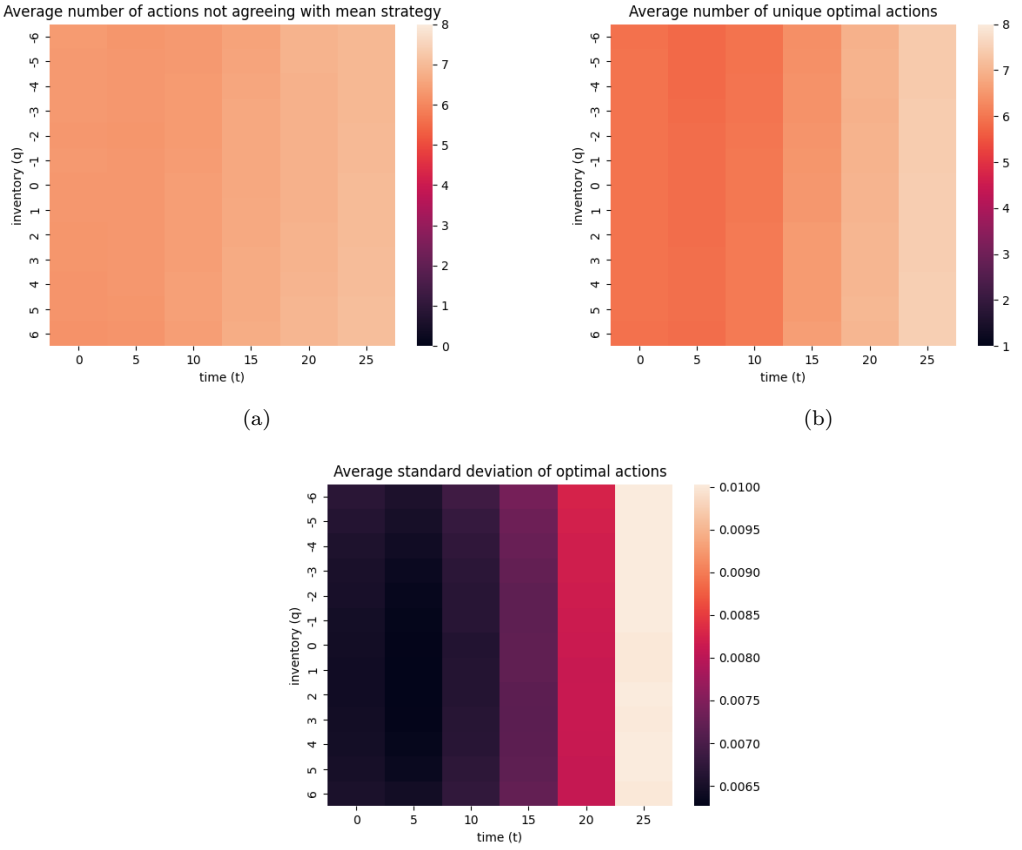


Figure B.12:
Stability of DDQN strategies in the SH-DDQN setting averaged over 1,000 random LOB states
(a) the average number of actions not agreeing with the mean strategy
(b) the average number of unique optimal actions
(c) the average standard deviation of the action-value of the optimal action according to the mean strategy.

B.14 SELL-HEAVY SCENARIO FOR THE MEAN SH-DDQN STRATEGY

In order to provide a more intuitive understanding of the SH-DDQN strategies and how they change depending on the current LOB, an example is shown in Figure B.13. The two LOB states used for this are shown in Figure 3.30 – a "balanced" and a "sell-heavy" LOB. In this figure one sees that the agent on average decreases its bid depths and decreases its ask depths, indicating a decreased desire to decrease its inventory. For a more quantifiable analysis of the different LOB states, see Table 3.18 for some financial measures.

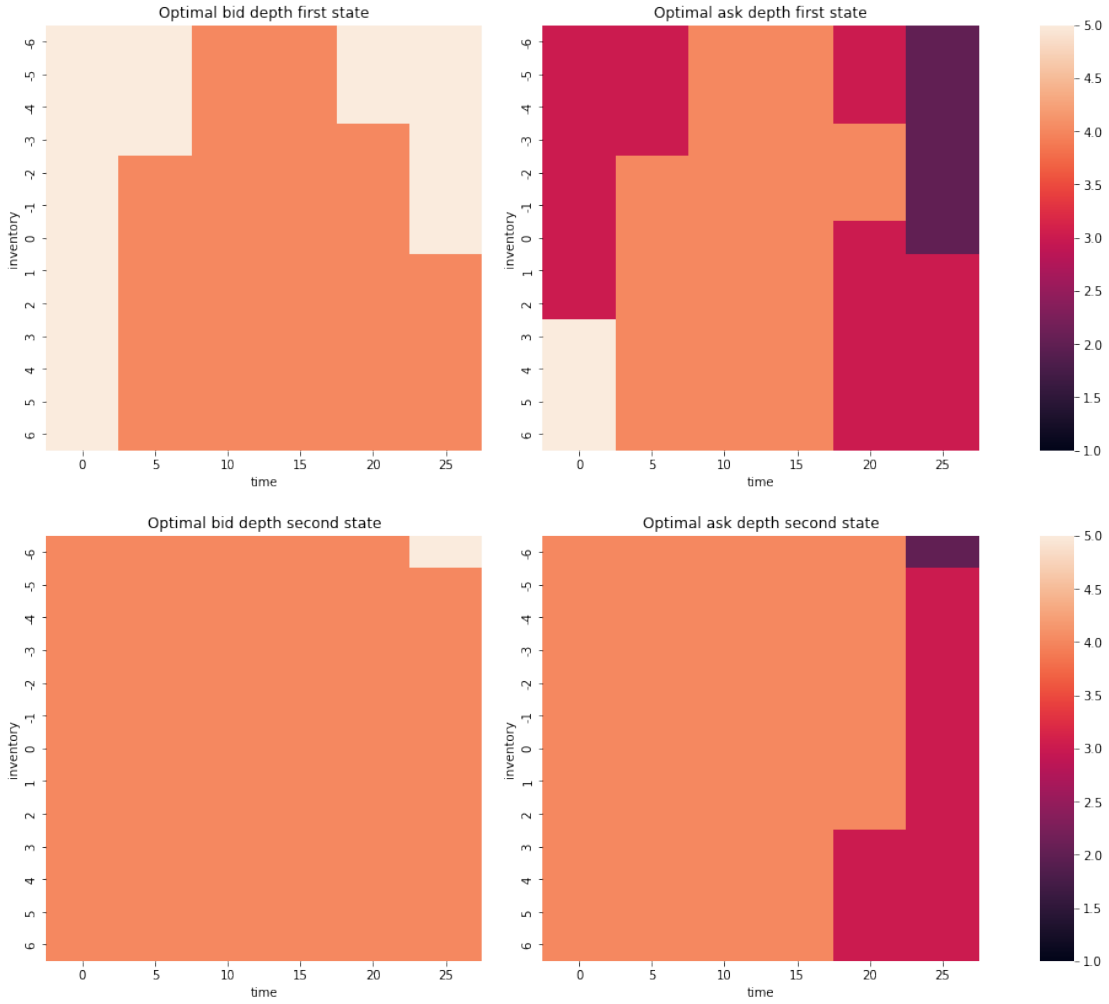


Figure B.13: Change from (**top**) neutral to (**bottom**) sell-heavy LOB in the SH-DDQN setting.

B.15 LARGE SPREAD SCENARIO FOR THE MEAN SH-DDQN STRATEGY

In order to provide a more intuitive understanding of the SH-DDQN strategies and how they change depending on the current LOB, an example is shown in Figure B.14. The two LOB states used for this are shown in Figure 3.30 – a "balanced" and a "large spread" LOB. In this figure one sees that the agent increases its bid and ask depths towards the end of the episode, indicating a reluctance to trade. For a more quantifiable analysis of the different LOB states, see Table 3.18 for some financial measures.

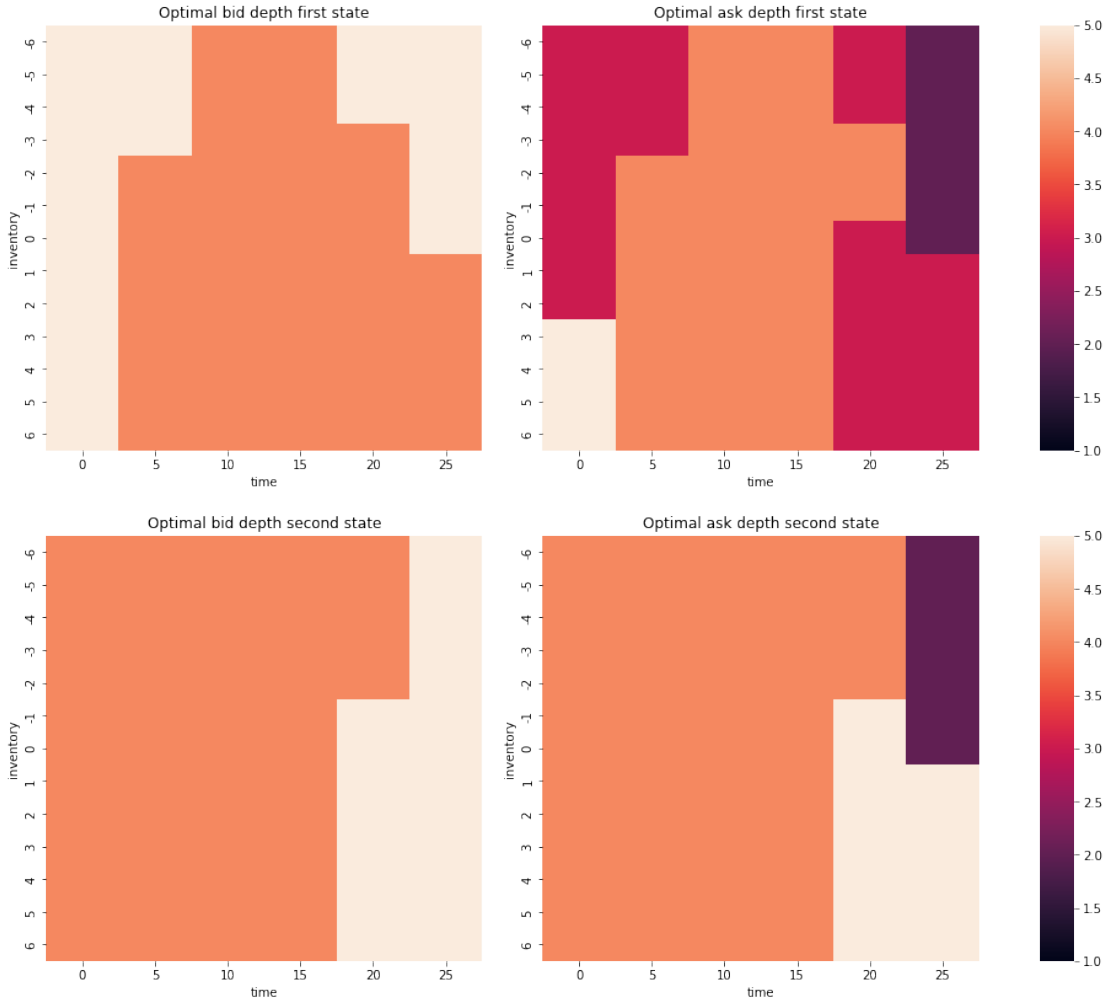
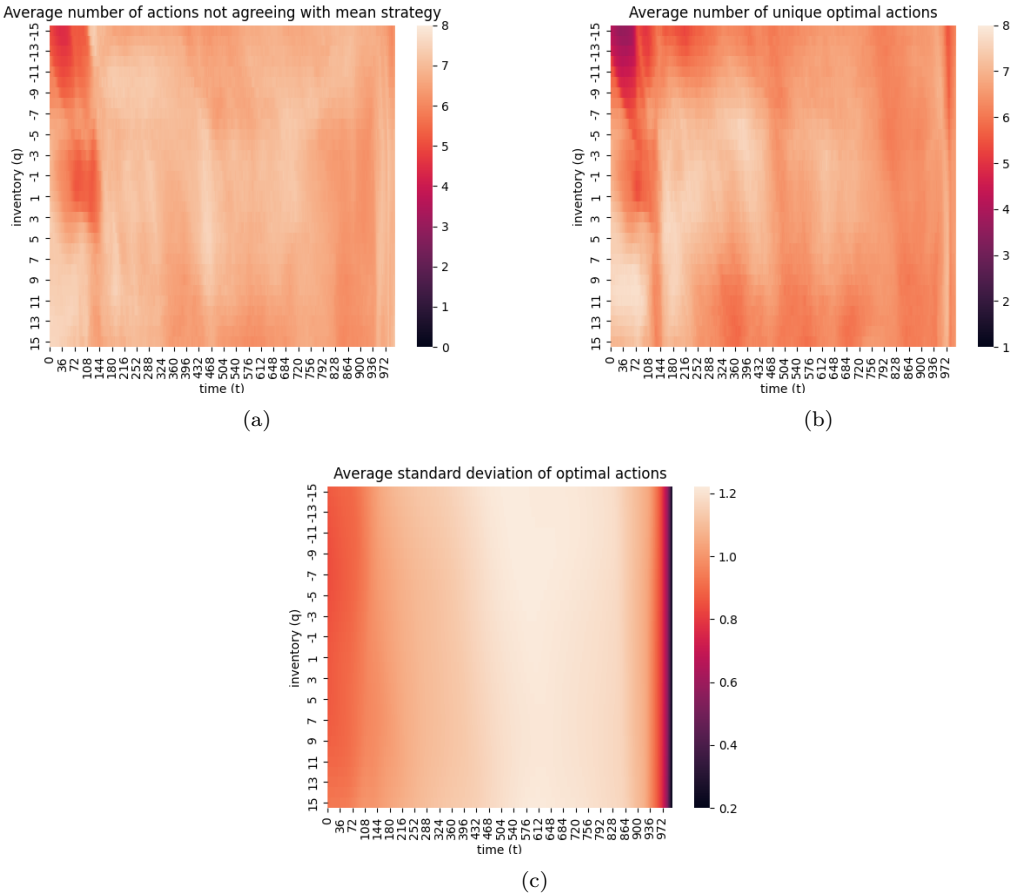


Figure B.14: Change from (**top**) neutral to (**bottom**) large spread LOB in the SH-DDQN setting.

B.16 STABILITY OF DDQN STRATEGIES IN THE LH-DDQN SETTING

Digging deeper into the stability of the DDQN in the LH-DDQN setting, we created three plots presented in Figure B.15. Figure B.15a shows that almost none of the individual DDQN strategies agree with the mean strategy on which action is optimal. Figure B.15b tells a somewhat similar story, however, there seems to be some more agreement. Finally, the standard deviations in Figure B.15c are all smaller than the expected reward, showing better signs of convergence than for the SH-DDQN setting in Figure B.12c.

In the standard deviation plot, Figure B.15c we also see a gradient in the direction of the time, similar to the SH-DDQN setting. However, here it is flipped, with the most certainty at the end of the episode.



(a) (b)

(c)

Figure B.15:
Stability of DDQN strategies in the LH-DDQN setting averaged over 100 random LOB states
(a) the average number of actions not agreeing with the mean strategy
(b) the average number of unique optimal actions
(c) the average standard deviation of the action-value of the optimal action according to the mean strategy.

B.17 SELL-HEAVY SCENARIO FOR THE MEAN LH-DDQN STRATEGY

Once again, an example of the mean strategy in two different states – a "balanced" and a "sell-heavy" LOB – is provided and shown in Figure B.16. The change in strategy between the two states is quite subtle; however, Figure B.17 shows the actual absolute difference between the strategies. From these figures, it is evident that the change in strategy is relatively small; most large changes occur at small absolute inventories. However, there seem to be some changes for small inventories on the bid size and for large inventories on the ask side, which in this case averages out to slightly higher willingness to buy. For a more quantifiable analysis of the different LOB states, see Table 3.18 for some financial measures.

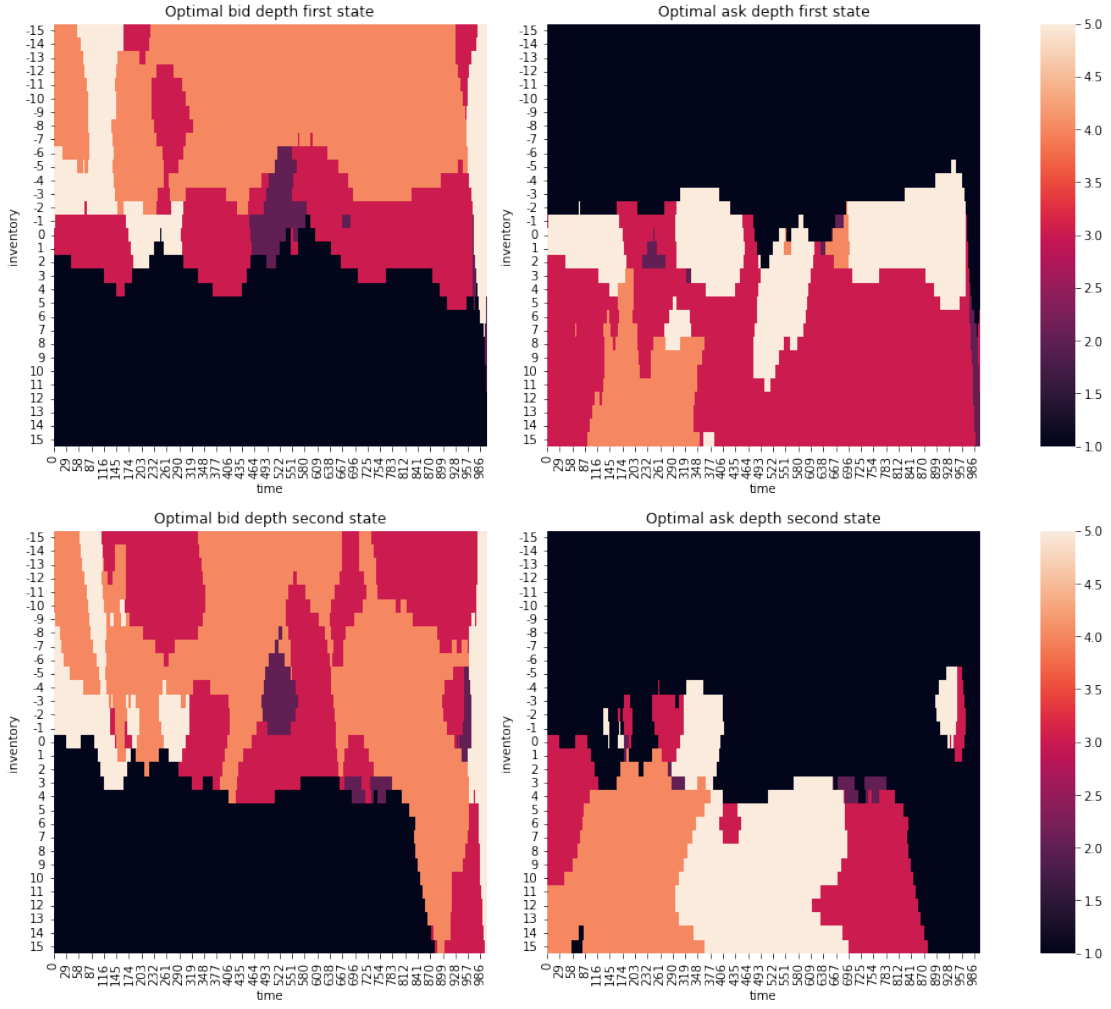


Figure B.16: Change from (**top**) neutral to (**bottom**) sell-heavy LOB in the LH-DDQN setting.

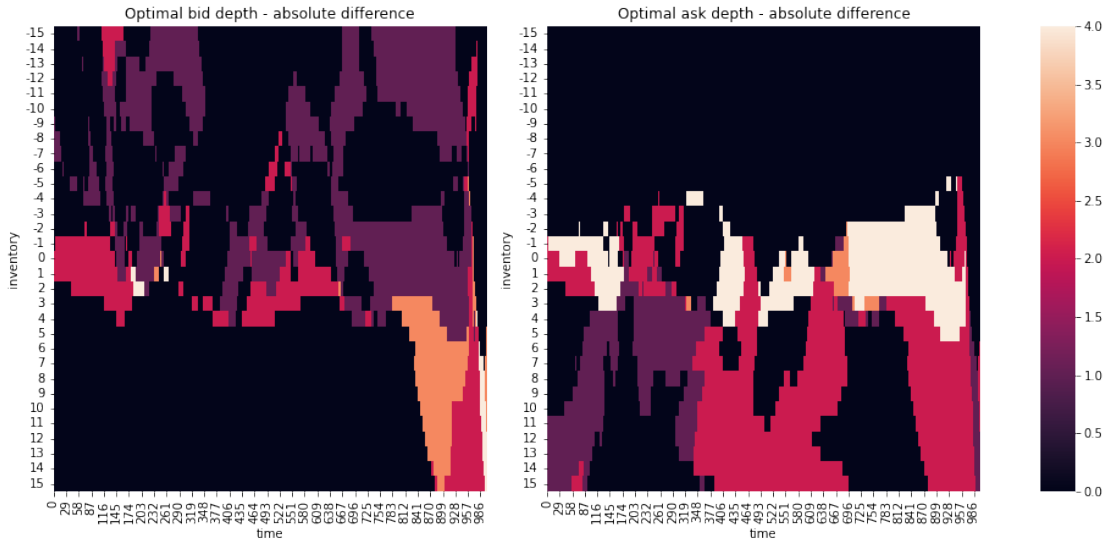


Figure B.17: The absolute difference in order depth between the two strategies based on the neutral and sell-heavy LOB.

B.18 LARGE SPREAD SCENARIO FOR THE MEAN LH-DDQN STRATEGY

Once again, an example of the mean strategy in two different states – a "balanced" and a "large spread" LOB – is provided and shown in Figure B.18. The change in strategy between the two states is quite subtle; however, Figure B.19 shows the actual absolute difference between the strategies. From these figures, it is evident that the change in strategy is relatively small; most large changes occur at small absolute inventories. However, there seem to be some changes for large inventories on the ask side. For a more quantifiable analysis of the different LOB states, see Table 3.18 for some financial measures.

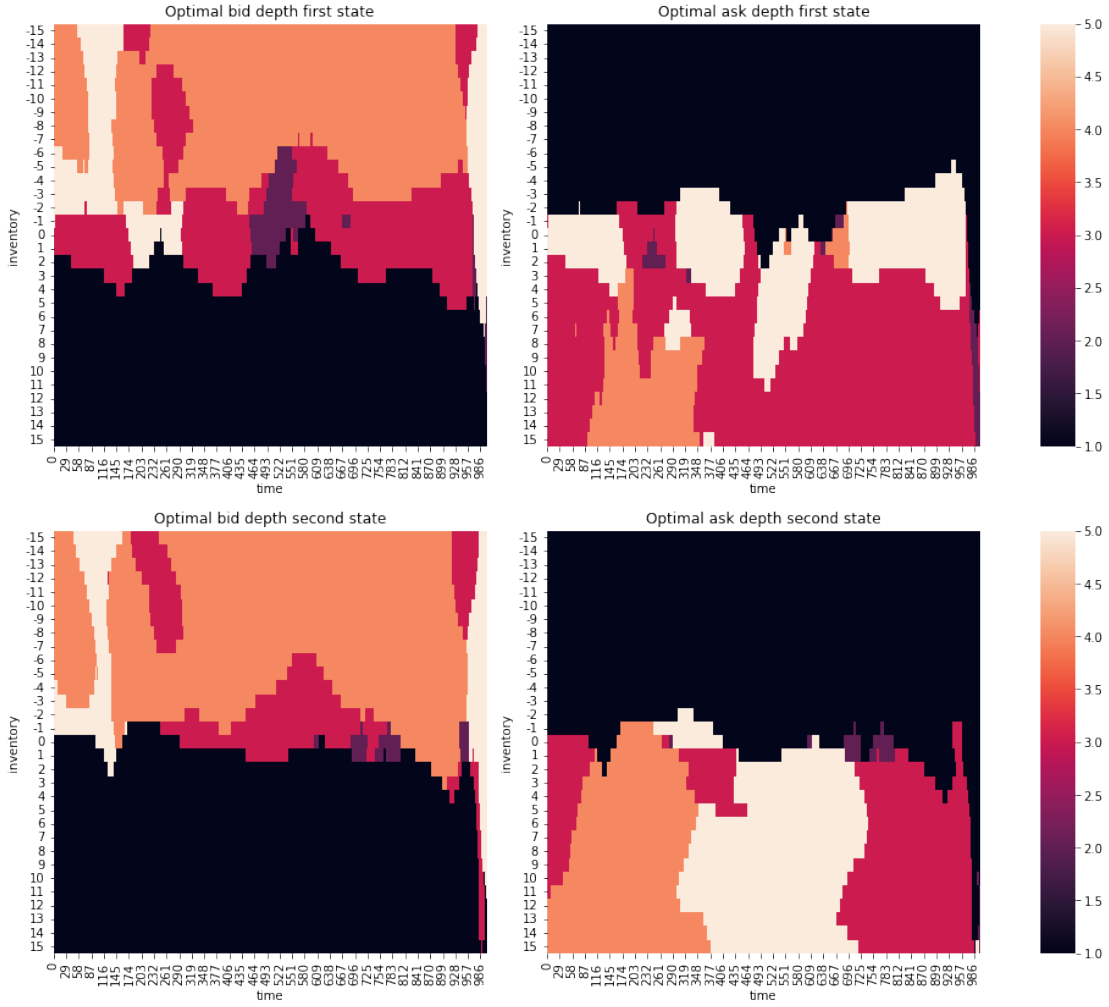


Figure B.18: Change from (**top**) neutral to (**bottom**) large spread LOB in the LH-DDQN setting.

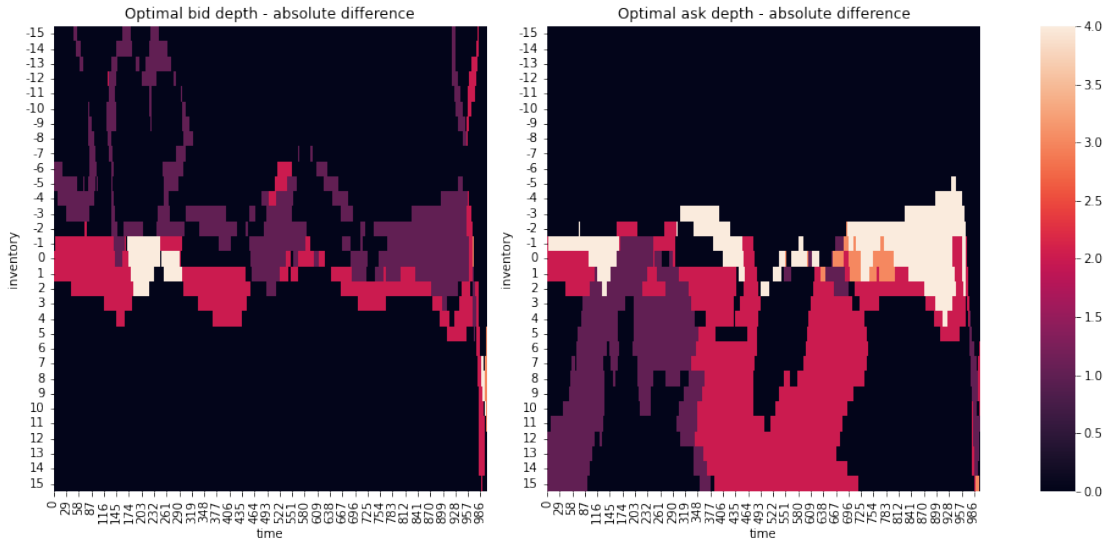


Figure B.19: The absolute difference in order depth between the two strategies based on the neutral and large spread LOB.