Data-driven simulations and policy gradients for limit order books



Michael Giegrich
St Anne's College
University of Oxford

Thesis submitted for the degree of
DPhil in Mathematics

Michaelmas 2024

Acknowledgements

First and foremost, I would like to express my sincere and deep gratitude to my supervisor Professor Christoph Reisinger for his continued support and his genuine interest in my research and my well-being. I would also like thank my industry partner and co-author Dr. Roel Oomen for his insightful comments and suggestions, which enhanced the applicability of my research. I am grateful to my co-author Dr. Yufei Zhang for the profound and motivating discussions on stochastic control and other topics. Furthermore, I would like to acknowledge the helpful feedback I received on my research from Professor Álvaro Cartea, Professor Sam Cohen, Professor Mihai Cucuringu, Professor Ben Hambly, Dr. Leandro Sanchez Betancourt, Professor Renyuan Xu, and other researchers I had the pleasure to interact with. I am also grateful to my colleagues and friends at the Math Department, who shared this journey with me, in particular, my (unofficial) office mates Filippo de Angelis, Philipp Jettkant, and Aldaïr Petronilia. In my previous studies, I was fortunate enough to learn and receive mentorship from great mathematicians, in particular, Professor Antje Berndt, Professor Lukas Gonon, Tobias Leopold, Professor Enno Mammen, and Professor Josef Teichmann. I acknowledge the funding from Deutsche Bank and the Mathematics of Random Systems CDT.

I am deeply grateful to my family, especially to my parents, Gisela and Jürgen, for helping me find my way. Furthermore, I thank my friends, in particular Carlo, Clemens, Martin, and Wasilios, for enriching my life. Finally, I could not have undertaken this journey without my wife, María, and her constant support, love, understanding, and encouragement.

Abstract

Over the last decades, central limit order books (LOBs) became a main method for asset execution in exchanges around the world for a wide variety of asset classes (e.g. [39, 85, 213]). Their practical relevance and inherent complex strategic interactions made LOBs a primary objective of study for academia and industry. The classical approach to modelling limit order books is based on identifying the most important features of the problem, encoding them in a parametric model and calibrating this model to market data. However, parametric modelling can introduce biases leading to a mismatch between actual market data and model output (e.g. [48]). Recent success in generative artificial intelligence (e.g. [75, 100]) and reinforcement learning (e.g. [93, 134, 194]) combined with the wide availability of LOB data motivates the usage of a new paradigm for decision making in LOBs, namely, fitting a generative, non-parametric model to the historical data and using reinforcement learning for optimization. In this thesis, we explore different aspects of this new paradigm.

Firstly, we propose a novel generative model, K-nearest neighbor resampling, for estimating the performance of a policy from historical data containing realized episodes of a decision process generated under a different policy. We provide statistical consistency results under weak conditions by generalizing a well-known result in non-parametric statistics on local averaging to include episodic data and counterfactual estimation. Compared to similar methods, our algorithm does not require optimization, can be efficiently implemented via tree-based nearest neighbor search and parallelization, and does not explicitly assume a parametric model for the environment's dynamics. Numerical experiments demonstrate the effectiveness of the algorithm compared to existing baselines in a variety of stochastic control settings.

Secondly, we show how K-nearest neighbor (K-NN) resampling can be applied to simulate LOB markets and how it can be used to evaluate and calibrate trading strategies. Using historical LOB data, we demonstrate that our simulation method is capable of recreating realistic LOB dynamics and that synthetic trading within the simulation leads to a market impact in line with the corresponding literature. In the context of a LOB with pro-rata type matching, we demonstrate how our algorithm can calibrate the size of limit orders for a liquidation strategy.

Thirdly, we move from generative modelling to optimization. In particular, we discuss applications of policy gradient methods for the optimal execution of an asset position via limit orders. We apply policy gradient methods in a parametric LOB model and a realistic generative adversarial neural network (GAN) LOB model. In the first case, we apply a zeroth-order gradient estimator and modify the algorithm to lower the variance in the estimate. In the second case, we alter a policy gradient method with a path-wise gradient estimator to overcome issues with the roughness of the loss-landscape. In both cases, we are able to learn effective trading strategies.

Finally, in LOBs, interactions may occur on the level of microseconds [39] and should, thus, be seen as practically time-continuous. We provide new insights on the usage of policy gradient algorithms in continuous time. In particular, we give sufficient conditions to prove the linear convergence of a policy gradient method in an exploratory finite-horizon linear quadratic control (LQC) problem in continuous time. We give numerical evidence for the linear convergence of the policy gradient method. Furthermore, considering policies with piece-wise constant parameters in time, we show numerically that rescaling the gradient for the policy gradient method by the time-discretization of the policy leads to a robust linear convergence over different action frequencies.

Contents

1 Introduction **7**

2 Background **11**
2.1 Limit order books . . . 11
2.2 The reinforcement learning environment and stochastic control . . . 12
2.2.1 Set-up . . . 13
2.2.2 Example: LQC problem . . . 14
2.2.3 Ergodic vs time-dependent control . . . 16
2.2.4 Reinforcement learning environment and time-continuous control problems . . . 16
2.3 Off-policy evaluation . . . 17
2.3.1 Problem formulation . . . 17
2.3.2 Algorithms for off-policy evaluation . . . 17
2.4 The policy gradient algorithm . . . 19
2.4.1 Parametrization of policy function and choice of features . . . 19
2.4.2 Gradient estimation: finite difference vs path-wise differentiation . . . 20
2.4.3 Gradient-based optimization algorithms . . . 21
2.4.4 The policy gradient algorithm . . . 22
2.4.5 The role of initialization and exploration . . . 23
2.4.6 Literature on policy gradient methods . . . 24
2.4.6.1 Historical remarks on policy gradient methods . . . 24
2.4.6.2 Theoretical results on policy gradient methods . . . 24
2.4.6.3 Policy gradient methods in finance and numerical mathematics . . . 25

3 K-Nearest-Neighbor Resampling for Off-Policy Evaluation in Stochastic Control **26**
3.1 Introduction . . . 26
3.2 Description of algorithm . . . 28
3.3 Consistency of K-nearest neighbor resampling . . . 31
3.3.1 Local averaging regression estimators . . . 33
3.3.2 The K_n-NN path regression estimator . . . 34
3.4 Experiments . . . 35
3.5 Proofs . . . 37
3.5.1 Proof of Theorem 2 . . . 37
3.5.2 Proof of Theorem 3 . . . 38
3.5.3 Consistency of the KNNR estimator . . . 42
3.6 Conclusion . . . 44

4 Limit Order Book Simulation and Trade Evaluation with K-Nearest-Neighbor Resampling 45
4.1 Introduction . . . 45
4.2 Limit order books . . . 47
4.2.1 Set-up . . . 47
4.2.2 Assumptions on limit order book dynamics . . . 49
4.2.3 Trading agent in the limit order book . . . 50
4.2.4 Interaction with the limit order book . . . 50
4.2.5 Limit order execution mechanisms . . . 51
4.3 Algorithm . . . 52
4.3.1 K-NN resampling for LOB simulation with a trading agent . . . 53
4.3.2 K-NN resampling for trading strategy evaluation . . . 54
4.4 Data set . . . 54
4.5 Simulation results . . . 55
4.5.1 Resampling statistics . . . 57
4.5.2 Single transitions . . . 57
4.5.3 Dynamic comparison . . . 60
4.5.4 Benchmark comparison . . . 62
4.5.5 Market order impact . . . 63
4.5.6 Limit order impact . . . 64
4.6 Policy evaluation . . . 65
4.7 State space extension . . . 67
4.8 Further research directions . . . 68

5 Policy Gradient Methods for Limit Order Books 69
5.1 Introduction . . . 69
5.2 Policy gradients for a parametric LOB model . . . 70
5.2.1 The parametric LOB model . . . 70
5.2.2 Discretized version of the parametric LOB model . . . 72
5.2.3 Algorithm . . . 72
5.2.3.1 Changes to the original algorithm . . . 74
5.2.3.2 Changes to the sampling oracle . . . 74
5.2.4 Numerical results . . . 75
5.2.4.1 Variance in gradient estimation and its impact . . . 76
5.2.4.2 Optimization procedure . . . 79
5.3 Policy gradients for GAN LOB model . . . 81
5.3.1 Execution problem in GAN LOB model . . . 82
5.3.2 Gumbel-LOB GAN . . . 84
5.3.3 Algorithmic considerations for policy gradients . . . 86
5.3.4 Results . . . 89
5.4 Conclusion . . . 91

6 Policy Gradient Methods for Continuous-time Linear Quadratic Control 93
6.1 Introduction . . . 93
6.1.1 Related work . . . 94
6.1.2 Notation . . . 95
6.2 Regularised stochastic LQ control problems with indefinite costs . . . 95
6.3 Optimisation over policy parameters and loss landscape analysis . . . 97
6.3.1 Parametrization and parametrized optimisation . . . 97
6.3.2 Loss landscape analysis . . . 98

6.4 Additional results . . . 99
6.4.1 Noncoerciveness of costs . . . 99
6.4.2 Geometry-aware policy gradient method . . . 100
6.4.3 Convergence analysis . . . 100
6.4.4 Mesh-independent linear convergence with discrete-time policies . . . 100
6.5 Numerical experiments . . . 102
6.5.1 Experiment details . . . 103
6.6 Proofs: Analysis of optimisation landscape . . . 105

7 Summary and outlook **111**

A Appendix: K-Nearest-Neighbor Resampling for Off-Policy Evaluation in Stochastic Control **126**
A.1 Theoretical considerations . . . 126
A.1.1 Environment . . . 126
A.1.2 Proof of Propositions 7 . . . 127
A.2 Proof of Proposition 2 . . . 131
A.3 Implementation . . . 140
A.3.1 Extended KNNR algorithm . . . 140
A.3.2 MFMC algorithm . . . 141
A.3.3 Complexity analysis . . . 141
A.4 Experiments . . . 142
A.4.1 Linear quadratic regulator . . . 142
A.4.2 Optimal execution in limit order book markets . . . 143
A.4.3 Online stochastic bin packing . . . 144
A.4.4 Baselines . . . 145

B Appendix: Limit Order Book Simulation and Trade Evaluation with K-Nearest-Neighbor Resampling **146**
B.1 Benchmarks . . . 146
B.1.1 Naive sampling . . . 146
B.1.2 CGAN . . . 147

Chapter 1

Introduction

Over the last decades, central limit order books (LOBs) became a main method for asset execution in exchanges around the world for a wide variety of asset classes (e.g. [39, 85, 213]). This order driven execution mechanism is based on the direct interaction of market participants ranging from institutional investors over banks to high-frequency trading hedge funds. These market participants may vary in their objectives and their level of sophistication. Aggregating this multitude of interactions leads to emerging dynamics usefully described by stochasticity in the observed trading data. This forces an individual market participant to make risky trading decisions under uncertainty over the price or the volume they may obtain for their trades[1]. Furthermore, the trader's decisions may impact the decision of other market participants who change their strategy in response. This mixture of randomness, strategic interactions, and practical relevance made LOBs a primary objective of study for academia and industry (e.g. [39, 85, 213]).

The classical approach to problems such as optimal execution or market making in limit order books is based on identifying the most important features of the problem, encoding them in a parametric model, and calibrating this model to market data. For decision making, these parametric models may then be solved using analytical or numerical approaches (e.g. [39, 85, 213]). However, parametric modelling can introduce biases leading to a mismatch between actual market data and model output (e.g. [48]). Basing decisions on a misspecified model can potentially have significant consequences on the outcomes for a trader.

Recent success in generative artificial intelligence (e.g. [75, 100]) and reinforcement learning (e.g. [93, 134, 194]) combined with the wide availability of LOB data motivates the usage of a new paradigm for decision making in LOBs. Namely, instead of describing the dynamics of the LOB using a parametric model, a generative, non-parametric model is fitted to the available data. Allowing for interactions with this generative model provides a trading simulator which a trader may use to evaluate trading strategies. Furthermore, the availability of a realistic simulator allows the usage of optimization techniques from reinforcement learning to search for effective trading strategies. This thesis provides novel approaches and algorithms to further the applicability of this new paradigm to decision making in LOBs and discusses some of the main challenges.

Chapter 2 contains background material for the following chapters. In particular, it gives more details on limit order books and discusses an environment for reinforcement learning problems. As an important benchmark, a simple linear quadratic regulator problem is introduced. Then the problem of off-policy evaluation is considered and some popular algorithms are presented. Finally, we introduce policy gradient methods as a way to solve the optimization problem in reinforcement learning and discuss different aspects for their implementation.

In Chapter 3, we propose a novel K-nearest neighbor resampling procedure for estimating the performance of a policy from historical data containing realized episodes of a decision process generated under a different policy. We provide statistical consistency results under weak conditions. In particular, we avoid the common

[1] In this context, uncertainty refers to the inability to perfectly predict outcomes due to imperfect information and risk describes the potential downsides of decisions under uncertainty.

assumption of identically and independently distributed transitions and rewards. Instead, our analysis allows for the sampling of entire episodes, as is common practice in most applications. To establish the consistency in this setting, we generalize Stone's Theorem, a well-known result in non-parametric statistics on local averaging, to include episodic data and the counterfactual estimation underlying off-policy evaluation (OPE). By focusing on feedback policies that depend deterministically on the current state in environments with continuous state-action spaces and system-inherent stochasticity effected by chosen actions, and relying on trajectory simulation similar to Monte Carlo methods, the proposed method is particularly well suited for stochastic control environments. Compared to other OPE methods, our algorithm does not require optimization, can be efficiently implemented via tree-based nearest neighbor search and parallelization, and does not explicitly assume a parametric model for the environment's dynamics. Numerical experiments demonstrate the effectiveness of the algorithm compared to existing baselines in a variety of stochastic control settings, including a linear quadratic regulator, trade execution in limit order books, and online stochastic bin packing. This chapter contains the results from [69] and is currently under review.

Main contributions: Chapter 3

- We present a novel nearest-neighbor based resampling algorithm (K-NN resampling) for off-policy evaluation in stochastic environments (see Section 3.2). The algorithm can generate trajectories without exact knowledge of the observed policy and does not rely on estimating a parametric model for the dynamics. It is faster than comparable existing algorithms and parallelizable. Furthermore, we demonstrate the method's effectiveness on various benchmark problems and show competitive performance of the resampling algorithm compared to baselines (see Section 3.4).
- We prove that the resampling algorithm leads to a consistent estimator for episodic sampling, under only weak assumptions (Section 3.3). For model-based OPE methods, this is to our knowledge the first consistency result under episodic sampling for non-deterministic environments with continuous state-action spaces.
- To prove consistency of the resampling algorithm, we extend a well known statistical result known as Stone's Theorem (see [87], Theorem 4.1), which gives conditions for the consistency of locally weighted regressions. Our generalization consists of including the episodic setting and the counterfactual element of OPE. This result can be found in Theorem 2 in Section 3.3.

In Chapter 4, we show how K-nearest neighbor (K-NN) resampling, the off-policy evaluation method proposed in Chapter 3, can be applied to simulate LOB markets and how it can be used to evaluate and calibrate trading strategies. Using historical LOB data, we demonstrate that our simulation method is capable of recreating realistic LOB dynamics and that synthetic trading within the simulation leads to a market impact in line with the corresponding literature. Compared to other statistical LOB simulation methods, our algorithm has theoretical convergence guarantees under general conditions, does not require optimization, is easy to implement, and computationally efficient. Furthermore, we show that in a benchmark comparison our method outperforms a deep learning-based algorithm for several key statistics. In the context of a LOB with pro-rata type matching, we demonstrate how our algorithm can calibrate the size of limit orders for a liquidation strategy. Finally, we describe how K-NN resampling can be modified for choices of higher dimensional state spaces. This chapter contains the results from [70] and is currently under review.

Main contributions: Chapter 4

- We introduce a framework for how K-NN resampling can be adapted for simulating LOBs and how to incorporate trades within these simulations (see Section 4.3). Furthermore, we describe how K-NN resampling can be used for evaluating trading strategies with both limit and market orders.

- We provide empirical evidence that K-NN resampling is capable of realistically simulating LOB dynamics with and without trade interventions (see Section 4.5). In particular, we show that our method can outperform a strong deep learning-based benchmark on key statistics.
- We demonstrate how K-NN resampling can be used to evaluate and choose trading strategies in a LOB with a pro-rata type matching mechanism (see Section 4.6). Specifically, we show how to calibrate the quantity of limit orders with our resampling algorithm.
- We illustrate how K-NN resampling can effectively be used in a higher dimensional state space and how the algorithm can be combined with dimension reduction techniques (see Section 4.7).

In Chapter 5, we move from creating realistic environments for evaluating trading strategies using historical data to the problem of learning the trading strategies. The chapter discusses applications of policy gradient methods for the optimal execution of an asset position via limit orders. We consider two different problem settings. First, we apply a policy gradient method to a parametric model which admits a closed-form solution. The main challenge in this setting is that the objective is not easily differentiable with respect to the parametrization of the policy as it controls the intensity of a jump process. To overcome this issue, we consider a zeroth-order approximation for the gradient and introduce several numerical modifications to standard policy gradient methods to limit the variance in the gradient estimator. With these modifications, we are able to learn a well-performing strategy for this problem. Second, we consider an optimal execution problem where LOB dynamics are given by a generative adversarial neural network (GAN) trained on real LOB data. In comparison to the first problem setting, the GAN simulator provides a more realistic view. Furthermore, by the choices for implementation of the GAN, the gradient of the objective with respect to the policy parameters can be easily estimated by using automatic differentiation on simulated trajectories. A main challenge in this setting is the roughness of the loss-landscape leading to a large number of suboptimal local minima. We overcome the issue of getting stuck in a bad local minima by applying a quasi-Newton method in the gradient optimization scheme and applying some of the numerical modifications explored in the first problem setting. With these modifications, we are capable of learning effective strategies for a realistic LOB dynamic that does not admit a closed form solution and by using a policy gradient method. The content of this chapter is currently unpublished and under preparation for submission.

Main contributions: Chapter 5

- For a parametric LOB model, we describe how a policy gradient method with zeroth-order approximation of the gradient can be modified for learning an execution strategy using limit orders. We show the effectiveness of the modified policy gradient algorithm in numerical experiments (see Section 5.2). To our knowledge, this constitutes the first application of a policy gradient method for optimal execution with limit orders in a paramateric LOB model.
- We propose a modified GAN-based LOB simulator that allows placing and evaluating of limit orders, a feature not previously included in GAN-based LOB simulators. Furthermore, the GAN simulator is differentiable with respect to the trader's actions. We train this novel architecture on LOB data and provide empirical evidence for the simulator's effectiveness. These results can be found in Section 5.3.1 and Section 5.3.2.
- We introduce a modified policy gradient algorithm using path-wise gradient estimates for learning execution strategies with limit orders in a GAN-based LOB simulator (see Section 5.3.3) and show the algorithms effectiveness in experiments where closed form solutions are not obtainable (see Section 5.3.4). To our knowledge, combining a differentiable GAN-based simulator for a complex dynamical system and a policy gradient method with path-wise gradient estimator is a novel approach to solving reinforcement learning problems with only observational data.

In LOBs, interactions may occur on the level of microseconds [39] and should thus be seen as practically time-continuous. Chapter 6 provides new insights on the usage of policy gradient algorithms in continuous time and emphasizes modifications to standard policy gradient algorithms to improve robustness with respect to action frequency. The chapter investigates the loss-landscape of an exploratory finite-horizon linear quadratic control (LQC) problem in continuous time. In particular, the analyzed model extends to a stochastic LQC with indefinite costs and may additionally include entropy regularization in the control objective. We start by stating the LQC problem as an optimization over time-dependent Gaussian policies where the mean is linear in the current state of the system and the covariance state-independent. Using Pontryagin optimality principle, we identify the gradient of the objective in the direction of the parameters. With this characterisation of the gradient we derive a non-uniform Łojasiewicz domination and a non-uniform smoothness condition for the cost functional. These conditions are sufficient to prove the linear convergence of a geometry-aware policy gradient method and are required as uniform conditions from the discrete-time analogue no longer hold. We provide numerical evidence for the linear convergence of the policy gradient method. Furthermore, considering policies with piece-wise constant parameters in time, we show numerically that rescaling the gradient for the policy gradient method by the time-discretization of the policy leads to a robust linear convergence over different action frequencies. This confirms a theoretical result from [71]. The presented content is the contribution from the author to the published article [71]. Additional results obtained by co-authors are included without proof for context and completeness. The relevant sections are appropriately highlighted.

Main contributions: Chapter 6

- We formulate a time-continuous, exploratory and stochastic LQC problem as an optimization problem over Gaussian policies parametrized by time-dependent functions.
- We use Pontryagin optimality principle to describe the gradient of the cost function in the direction of the policy parameters and using this characterization we show that the control objective satisfies a non-uniform Łojasiewicz condition (Proposition 4) and smoothness condition (Proposition 5). These conditions can be used to prove the linear convergence of a geometry-aware policy gradient descent method. In contrast to previous results in discrete time, our results require the usage of tools from stochastic analysis and PDEs.
- We conduct numerical experiments for an exploratory LQC problem arising from mean-variance portfolio construction. We provide evidence for the linear convergence of a geometry-aware policy gradient method and show that proper rescaling of gradients leads to robustness of the learning algorithm with respect to action frequency. These numerical results confirm theoretical results obtained by our co-authors.

Chapter 7 summarizes the thesis and provides additional future directions for research.

Chapter 2

Background

This chapter introduces the background for some of the main concepts and algorithms explored in this thesis. We start by introducing limit order books (LOBs) and a general overview of applications of reinforcement learning to LOBs. We then present a dynamical systems formulation for Markov decision processes and discuss several characteristics of such control problems observed in LOB related tasks. We then introduce the concept of off-policy evaluation and discuss some standard algorithms. We conclude by considering policy gradient methods and discuss several related numerical topics.

2.1 Limit order books

Over the last few decades LOBs became the main trading mechanism on exchanges around the world for a wide variety of assets (e.g. NYSE [42], NASDAQ [102], CME [129], EUREX [56]). Central limit order books mostly replaced a brokerage based trading system. According to [160], in these markets, a customer submitted a trading order to a brokerage firm. This brokerage firm had traders on the floors of physical exchanges and these floor traders approached so-called specialists who maintained order books on certain assets for execution. For executing a trade the specialists received a commission. Furthermore, they may have gained additional profit from trading on their own account. With the emergence of electronic exchanges in the 1980s [85], central limit order books operated by the exchanges themselves were introduced, e.g. the Paris Bourse in the early 1990s [28]. In the early 2000s, the security exchange commission (SEC) investigated multiple specialists operating on the NYSE and found evidence that the specialist firms prioritized their own orders before client orders. This lead to a penalty on specialist firms totalling $240 million [190] and coincided with a push for centralization of trading activity by regulators [124]. By 2012, central limit order books matched trades in more than half of the worlds financial markets [77] where fully electronic execution allows for the direct interaction of market participants on the scale of microseconds [39].

The basic functioning of a LOB depends on two order types – limit orders and market orders. A limit order constitutes a commitment by a trader to buy or sell a certain quantity of an asset for a certain price from the moment the order is placed. Limit orders are either bid or ask orders where a bid order describes the trader's willingness to buy while an ask order corresponds to their willingness to sell. The prices for which a limit order can be placed are usually pre-determined by the trading venue and are usually grouped by equidistant tick sizes. The limit order is valid until either the entire quantity is executed or until the order is cancelled by the trader. A limit order is either executed against a market order or against an incoming limit order that crosses the spread. A market order either sells or buys a certain quantity of an asset immediately as the order is placed. Market orders are usually executed against active limit orders. The orders are recorded and executed in a central book usually maintained by the exchanges. This book is referred to as the central limit order book. Chapter 4 contains a more detailed explanation of the functioning of LOBs and their dynamics. Furthermore, the chapter provides

references for different modelling approaches for LOBs presented in the literature (see [109] for a comprehensive overview).

In the broad literature on LOBs from the viewpoint of an individual trader, two tasks emerged as the dominant research focus. Namely, the optimal execution of assets and the problem of market making. In optimal asset execution, a trader tries to sell (or buy) a certain amount of an asset using limit and/or market orders. For asset execution with market orders, the main objective for optimization is usually to limit the price impact triggered by placing orders (e.g. [3]) and/or the capture of trading signals (e.g. [133]). For optimal execution with limit orders, the main objective is concerned with managing the uncertainty of the limit order being filled (e.g. [17, 82]). The market making problem focusses on a trader who may take positions on either side of the order book and by doing so tries to profit of the trading of other market participants by providing liquidity to the market. In these kinds of problems, the market maker may try to optimize the spread they quote or the position they take in the book while taking into account possible inventory risks and trading signals (e.g. [9, 41, 81, 84]). In Chapter 5, we present a model for optimal execution with limit orders from [39] and we refer to [39, 85, 213] for more details on optimal execution and market making problems.

Furthermore, there has been an increased interest in approaching these kinds of trading problems using reinforcement learning techniques, where the general idea is that a trading strategy is learned by direct interaction with the environment. We only give a limited number of examples from the existing literature and refer to the reviews [43, 89] for a more complete discussion. Popular approaches include Q-learning-based methods (e.g. [2, 99, 159, 161, 199, 227]), policy-based methods (e.g. [88, 227]), and actor-critic methods (e.g. [66, 222, 227]). As problem settings, researchers considered both optimal execution (e.g. [88, 99, 159, 161, 222, 227]) and market making (e.g. [2, 66, 199]). Within these frameworks, both market and limit orders can potentially be used. For market dynamics, often a simple replay of historical data is used (e.g. [2, 66, 159, 161, 199, 222, 227]). In this case, the implicit assumption is made that the order placed by the reinforcement learning agent does not impact the future state of the market (cf. Chapter 4). Other approaches consider (calibrated) parametric market models, e.g. [88, 99]. Other important design choices are the state space which describes the possible states of the underlying dynamical system and the action space which determines the possible ways to interact with the system (see Section 2.2). The state space may include trader's characteristics such as current inventory or cash, problem characteristics such as current time for finite horizon problems, market characteristics such as order book imbalances, previous market returns or the current spread. The action space is often chosen as either the order size for market orders or the quoted spread for limit orders[1]. For Q-learning-based methods the actions space is usually discretized on a finite interval in order to simplify the maximization of the Q-function. Common optimization targets are implementation shortfall (e.g. [66, 99, 159, 222]) and profit-and-loss (e.g. [2, 161, 199, 227]). Such a target may be combined with inventory constraints, e.g. [2, 66].

2.2 The reinforcement learning environment and stochastic control

The environment for reinforcement learning is typically described by a Markov Decision Process (MDP). Since most standard texts on reinforcement learning such as [203] are more geared towards engineering applications, they often lack in generality and specificity to simplify readability. Instead, we will introduce a variation of a discrete-time stochastic optimal control model given by [23] that is better suited to our needs. More specifically, we will introduce a finite horizon model with path- and time-dependent rewards. The dynamics of the system are given by a random dynamical system [2]. This model is particularly relevant for financial applications where the success of a strategy is judged at a final point in time and may be dependent on decisions at an earlier time (i.e. the optimal execution problem). We will then give an example in form of the linear quadratic control problem (LQC) which is of interest due to its relative simplicity while maintaining a certain level of generality.

[1] Note however that the quoted spread may not be the appropriate action for markets trading at the touch (i.e. only limit orders at the best bid/ask level are executed). For an example see Chapter 4 and Chapter 5.

[2] For a comparison between a Markov-kernel formulation and a random dynamical systems formulation for the dynamics consider Appendix A.1.

Then, we will shortly discuss the difference between an ergodic model and a time-dependent episodic setting. Finally, we will comment on the relationship between a discrete and a continuous time framework.

2.2.1 Set-up

We begin by recalling the definition of a stochastic kernel:

Definition 1 (Cf. Definition 7.12, [23]). *Let $X \subseteq \mathbb{R}^m$ and $Y \subseteq \mathbb{R}^n$ with $m, n \in \mathbb{N}$. A stochastic kernel $q(dy|x)$ on Y given X is a collection of probability measures parametrized by $x \in X$. Furthermore, equip X with a σ-algebra $\mathcal{F}$, and let $\mathcal{B}_{P(Y)}$ denote the Borel σ-algebra induced by the weak topology on the space of probability measures $P(Y)$ on Y. If $q(dy|\cdot) : X \to P(Y)$ is $\mathcal{F}$-$\mathcal{B}_{P(Y)}$-measurable, we say q is $\mathcal{F}$-measurable.*

Remark 1. *For a sequence $v = (v_k)_{k=0}^N$ with $N \in \mathbb{N}$ or $N = \infty$ and an $l \in \mathbb{N}$, v_l denotes the lth element of the sequence while $v^l = (v_k)_{k=0}^l$ denotes the sequence up to and including the lth element in this Chapter.*

Definition 2. *[Cf. Definition 8.1, [23]] A finite horizon stochastic optimal control model with path dependent rewards is an eight-tuple $\{S, C, U, W, p, f, r, T\}$ where*

- *$S \subseteq \mathbb{R}^m$ is the state space,*
- *$C \subseteq \mathbb{R}^n$ is the control or action space,*
- *a function U from S to the set of non-empty cuboids on C is the control constraint,*
- *$W \subseteq \mathbb{R}^d$ is the disturbance space,*
- *the Borel-measurable stochastic kernel $p(dw|x, u)$ on W given $S \times C$ is the disturbance kernel,*
- *Borel-measurable $f : S \times C \times W \to S$ is the system function or the environment and*
- *$r = (r_t)_{t=0}^{T-1}$ is a family of integrable reward functions $r_t : S^t \times C^t \times W^t \to \mathbb{R}$ defined on paths,*
- *$T \in \mathbb{N}$ is the time-horizon.*

All sets are assumed to be non-empty. The letters x, u and w are used to denote an element of S, U and W, respectively.

The system function f can be thought of describing a random dynamical system of the form

$$x_{t+1} = f(x_t, u_t, w_t) \text{ for } t = 0, ..., T-1$$

starting from some initial state x_0. The initial state may be drawn randomly from a distribution ν and the noise term w_t is random with distribution given by $p(dw|x_t, u_t)$. Informally speaking, given this dynamical system our goal is to find a sequence u^T where each $u_t \in U(x_t)$ is chosen dependent on (u^{t-1}, x^t) such that

$$\mathbb{E}_p[\sum_{t=0}^{T-1} r_t(x^t, u^t, w^t)]$$

is maximised[3]. Furthermore, we choose the reinforcement learning convention of maximising rewards rather than the control convention of minimizing costs. Both formulations are of cause equivalent and both will be used if convenient. Before stating the optimization problem in its entirety, we will allow for stochastic choice of actions by introducing the concept of policy.

[3]Note that the statement is made precise in Definition 2.

Definition 3. *[Cf. Definition 8.2, [23]] A policy in the context of Definition 2 is a sequence* $\pi = (\mu_0, \mu_1, ..., \mu_{T-1})$ *such that, for each* t, $\mu_t(du_t|x^t, u^{t-1})$ *is a measurable stochastic kernel on* C *parametrized on* S^t, C^t *satisfying*

$$\mu_t(U(x_t)|x^t, u^{t-1}) = 1$$

for all paths (x^t, u^{t-1})*. If all* μ_t *are measurable, we say that* π *is measurable. If, for each* t, μ_t *is parametrized only by* x_t*, then* π *is a Markov policy. If, for each* t *and all paths* x^t, u^{t-1}, μ_t *is a delta measure,* π *is a deterministic policy. In this case, by a slight abuse of notation, we identify* μ_t *for all* $t = 0, ..., T-1$ *with the function mapping from the path histories to the point of concentration in the control space. The set of all policies is denoted by* Π' *and the set of all deterministic Markov policies by* Π*.*

Remark 2. *If we use* Π' *or* Π*, we assume these sets to be non-empty. A sufficient condition for this assumption is that the graph set of the function* U *is analytic (cf. Proposition 7.49, [23]).*

These considerations allow us to formulate the optimisation problem:

Definition 4. *[Cf. Definition 8.3, [23]] Consider the model from Definition 2 and a policy* π*. For a* $t \leq T$*, the value function of the policy* π *with history* $y = (x^{t-1}, u^{t-1})$ *is given by*

$$J_t(y; \pi) = \mathbb{E}_{p,\pi}[\sum_{k=t}^{T-1} r_k(x^k, u^k, w^k)|y]$$

The optimal value function over the policy set Π' *is given by*

$$J_t^*(y) = \sup_{\pi \in \Pi'} \mathbb{E}_{p,\pi}[\sum_{k=t}^{T-1} r_k(x^k, u^k, w^k)|y]$$

where $t \leq T$ *and* $y = (x^{t-1}, u^{t-1})$*.*

Remark 3. *In this definition, we follow the reinforcement learning naming convention by differentiating between the value function and the optimal value function.*

2.2.2 Example: LQC problem

In this section, we introduce a specific control problem, namely a version of the linear quadratic control (LQC) problem. This framework is of interest in control theory since it allows a wide array of theoretical results while requiring non-trivial control behaviour. Furthermore, the LQC is believed to approximate control environments with more complex system dynamics and value functions. Based on these properties, the LQC gained a significant interest as a benchmark problem in reinforcement learning as it allows to evaluate an algorithm in a well-understood environment and permits provable statements on some of the algorithm's behaviour (see e.g. Chapter 6). In this section, we will introduce a stochastic and episodic LQC framework and its solution for the discrete-time case. After that, we will comment on the approximation capabilities of the LQC.

We will mostly follow the exposition given in [21] and will use a cost rather than a reward formulation. The first defining property of a LQC problem are the linear system functions

$$x_{t+1} = A_t x_t + B_t u_t + w_t \text{ for } t = 0, 1, ..., T-1,$$

where $x_t \in \mathbb{R}^m$, $u_t \in \mathbb{R}^n$, and w_t is an iid. random vector in $\mathbb{R}^m$ with zero mean and finite second moments independent of x_t and u_t. Moreover, A_t is a $\mathbb{R}^{m \times m}$ matrix and B_t is a $\mathbb{R}^{m \times n}$ matrix. The second defining property is the quadratic cost function

$$\mathbb{E}\left[x_T' Q_T x_T + \sum_{t=0}^{T-1} (x_t' Q_t x_t + u_t' R_t u_t)\right]$$

where the apostrophe denotes the transpose, $Q_t \in \mathbb{R}^{m\times m}$ is symmetric positive semidefinite and $R_t \in \mathbb{R}^{n\times n}$ symmetric positive definite for all t.

Using the dynamic programming equation and setting the derivative of the term within the minimum operator to zero one can derive the optimal policy recursively. The optimal policy is linear in the state and is given by

$$\mu_t^*(x_t) = L_t x_t,$$

where L_t is the *gain matrix* given by

$$L_t = -(B_t' K_{t+1} B_t + R_t)^{-1} B_t' K_{t+1} A_t.$$

The matrix K_t is the symmetric positive semidefinite matrix given recursively by

$$K_N = Q_N$$

and

$$K_t = A_t'(K_{t+1} - K_{t+1}B_t(B_t' K_{t+1} B_t + R_t)^{-1} B_t' K_{t+1})A_t + Q_t.$$

The second equation is called a *discrete-time Riccati equation.*

The interest in LQC problems can be partly explained by the fact that they can be understood as the Taylor approximation of a wider class of non-linear stochastic control problems. Below you can find a remark on this interpretation of an LQC problem based on a discussion in [201] and adapted to our framework[4].

Remark 4. *Consider a finite horizon stochastic optimal control model $\{S, C, U, W, p, f, r, T\}$ as defined above. Additionally assume that f and $r = (r_t)_{t=0}^T$ are differentiable, and that the state and control pair stay sufficiently close to a reference value (x_t^*, u_t^*, w_t^*) for every t. Without loss of generality, we choose $(x_t^*, u_t^*, w_t^*) = (0,0,0)$. Then by Taylor expansion, we have*

$$\begin{aligned} f(x,u,w) =& f(0,0,0) + \partial_x f(0,0,0)x + \partial_u f(0,0,0)u + \partial_w f(0,0,0)w + \cdots, \\ r_t(x,u) =& r_t(0,0) + \partial_x r_t(0,0)x + \partial_u r_t(0,0)u + \frac{1}{2}\langle \partial_{xx} r_t(0,0)x, x\rangle \\ &+ \langle \partial_{xu} r_t(0,0)x, u\rangle + \frac{1}{2}\langle \partial_{uu} r_t(0,0)u, u\rangle + \cdots, \end{aligned}$$

where "$\cdots$" signifies higher order terms. Truncating the higher order terms and denoting

$$\begin{array}{ll} A = \partial_x f(0,0,0), & B = \partial_u f(0,0,0), \\ C = \partial_w f(0,0,0), & Q_t = \partial_{xx} r_t(0,0), \\ S_t = \partial_{xu} r_t(0,0), & R_t = \partial_{xu} r_t(0,0), \\ q_t = \partial_x r_t(0,0)^\top, & \rho(s) = \partial_u r_t(0,0)^\top, \\ c = f(0,0,0), & \end{array}$$

we get the state dynamics

$$x_{t+1} = Ax_t + Bu_t + Cw_t + c$$

and the cost functional

$$J_t(y;\pi) = \mathbb{E}_{p,\pi}[\sum_{k=t}^{T-1} x_t' Q_t x_t + x_t' S_t u_t + u_t' R_t u_t + 2q_t x_t + 2\rho_t u_t | y].$$

This constitutes a LQC problem.

[4]While this remark provides a heuristic explanation of the approximating quality of the LQC problem, we are not aware of any formal result linking non-linear-quadratic stochastic control problems with the related linear-quadratic approximation.

2.2.3 Ergodic vs time-dependent control

Note that the formulation in Section 2.2.1 allows for changing rewards over time depending not only on Markovian information (i.e. current state and control choice) but also on the history of realizations. We choose this formulation to highlight the importance of time-dependence and to include important financial applications such as the optimal execution of assets with market orders over a fixed time-horizon. In the seminal paper [3], for instance, the optimal policy solely relies on the initial asset endowment and the relative time in the time horizon. This behaviour is partially explained by their choice of objective function. Namely, in their model the objective is given as minimizing a function depending on non-linear transformations of the state paths (e.g. the expected implementation shortfall plus its variance weighted by a parameter) over the control paths. Reinforcement learning formulations of this problem (e.g. [88]) require a reward that is only granted at the end of the trading period and it depends on a non-linear transformation of the state paths. In particular, this non-linear transformation of the paths does not allow an online period-by-period attribution of rewards and, thus, in all periods except the last no reward occurs. Hence, the reward function changes over time and is path dependent. Similar modelling choices in finance can, for example, be found in the context of option hedging. We face this particular challenge in Chapter 5.

By enlarging the state space appropriately (i.e. including time and paths), one can see that our modified formulation is equivalent to the formulation in [23]. As mentioned above, one reason to choose this formulation is to highlight the time-dependence of a control, which should inform the selection of a learning algorithm. For example, it might be beneficial to estimate parameters for each time period rather than share parameters across the time horizon. The time-dependence in finite-time horizon problems is contrast to the ergodic behaviour of infinite-time horizon control settings where policies are stationary in time (see e.g. [23]). Note that finite-time horizon problems with non-stationary optimal policies are considered in Chapter 5 and Chapter 6.

2.2.4 Reinforcement learning environment and time-continuous control problems

In financial applications, control problems are commonly formulated in continuous time. In part, this might be explained by the mathematical tools available in continuous time. From a practical perspective, this modelling choice often seems justified, especially since the rise of low latency trading environments. For reinforcement learning algorithms, however, we in general require a discrete version as for example formulated above. A priori, it seems unclear how discrete and continuous control problems are related. The topic has been originally studied in the context of numerical methods for solving continuous control problems and has been applied to study some reinforcement learning algorithms. We provide additional context in Chapter 6, where we consider a policy gradient method for a time-continuous control problem.

In [127] and [126], the author establishes that a wide variety of stochastic control problems can be approximated by Markov chains in a weak sense. Based on this finding, they establish point-wise convergence of the optimal value function of the Markov chains to the optimal value function of the original problem. These results have for example been used in [155] to establish convergence of the value function for a model-based reinforcement learning algorithm to its continuous version and in [165] for a multigrid-based method.

More recently, a continuous formulation of the reinforcement learning problem was considered in [215]. This framework was applied in [216] to continuous-time mean-variance portfolio selection. The theoretical considerations in this paper focus mostly on the continuous setting and do not systematically study the discretization occurring in practical implementation. For policy evaluation, a problem closely related to control, [112] show that any convergent time-discretized algorithm converges to its continuous-time counterpart as the mesh-size diminishes.

Other literature that is potentially relevant to better understand discretizations of continuous systems is concerned with piecewise constant policies in the context of approximating value functions for controlled degenerate diffusion processes [125]. Here, an optimal discrete policy is evaluated in a continuous environment and it is shown that the optimal value function over discretized policies converges to the optimal value function over all admissible policies. Moreover, a convergence rate is established.

2.3 Off-policy evaluation

After introducing an environment for reinforcement learning, we would like to highlight an additional assumption often made implicitly by algorithms used to search for good policies. Namely, these algorithms assume that while the dynamics of the system may be unknown, the algorithm can directly interact with the system and, for example, try new policies directly. From a causal perspective, this is equivalent to having a controlled experimental environment. For practical applications such as in finance this is often not the case as experimentation with the actual environment may prove prohibitively costly. For example, in optimal asset execution, a new and untested strategy may lead to large losses. However, observational data of the environment where a different strategy was used might be available (as is the case for LOBs). Off-policy evaluation, a subfield of reinforcement learning, is concerned with evaluating policies with data gathered under different policies (i.e. observational data). This section will give an introduction to the problems considered in off-policy evaluation and provide a short overview on the most popular algorithms used in this context. The discussion is intended to give additional background for Chapter 3 and Chapter 4. Additionally, some of the methods introduced here are used as benchmarks in Chapter 3. The problem formulation and the dichotomy for off-policy evaluation algorithms is based on the account given in [214].

2.3.1 Problem formulation

To introduce the off-policy evaluation problem, we consider a control problem $\{S, C, U, W, p, f, r, T\}$ (see Definition 2) where we assume without loss of generality (see Section 2.2.3) that the reward is given by a function $r : S \times C \times W \to \mathbb{R}$ rather then a family of functions for each point in time. For simplicity, we write for the reward $r_t = r(x_t, u_t, w_t)$. We are given a behavioural policy π_b and a target policy π_e where for notational simplicity we assume the policy is Markov and the same for each t. Furthermore, we have access to a data set $\mathcal{D} = \{\tau^i\}_{i=0}^N$ of independent trajectories $\tau^i = (x_0^i, u_0^i, r_0^i, \ldots, x_{T-1}^i, u_{T-1}^i, r_{T-1}^i)$ generated with the behavioural policy π_b. With this data set, the goal is to estimate the (expected) value function

$$\mathbb{E}[J_0(x_0, u_0; \pi_e)]$$

for the policy π_e where the expectation is take over the distribution of initial state of the system x_0 and the initial action u_0 is distributed according to $\pi_e(x_0)$.

2.3.2 Algorithms for off-policy evaluation

According to a slightly modified dichotomy from that in [214], one can identify three different types of off-policy evaluation algorithms: inverse propensity scoring (IPS) methods, model-free[5] estimation of the value function and model-based estimation of the value function. Furthermore, there exist hybrid methods combining these approaches. For the sake of brevity we will focus on non-hybrid methods and refer to [214] for a discussion of the most used hybrid approaches.

IPS methods IPS methods are based on the notion of importance sampling, a well-known approach in Monte Carlo methods (e.g. [73]). For this exposition, we assume that for every $x \in X$, the policies π_b, π_e admit densities p_b, p_e and that π_e is absolutely continuous with respect to π_b (i.e. $p_e(u \mid x) > 0$ implies $p_b(u \mid x) > 0$). The basic idea for IPS methods is to reweigh the returns of the samples in the data set with the likelihood ratio between the target policy and the behaviour policy $(p_e(u \mid x)/p_b(u \mid x))$. For an episodic setting, this weighting of the returns can be either done over entire trajectories or for each decision. For that purpose, we define the

[5]Note that we use the term "model-free" in the sense of reinforcement learning where "model-free" usually refers to the usage of dynamic programming principles for estimation of value functions (see e.g. [203]).

cumulative importance weight $\rho^i_{j:j'}$ as

$$\rho^i_{j:j'} = \prod_{t=j}^{\min(j',T-1)} \frac{p_e\left(u_t^i \mid x_t^i\right)}{p_b\left(u_t^i \mid x_t^i\right)} \tag{2.1}$$

where $\rho = 1$ if $j > j'$ and i refers to the i-th sample in $\mathcal{D}$. The trajectory-wise estimator for the value function is then given by

$$\sum_{i=1}^{N} \frac{\rho^i_{0:T-1}}{N} \sum_{t=0}^{T-1} r_t^i$$

and the per-decision estimator by

$$\sum_{i=1}^{N} \sum_{t=0}^{T-1} \frac{\rho^i_{0:t}}{N} r_t^i.$$

While this method provides a simple approach to off-policy evaluation, in practice, it often suffers from instability if there is a large mismatch between π_e and π_b. To overcome this issue, researchers proposed different sample-weighing schemes (instead of $1/N$) to improve robustness (see e.g. [54, 114]). While these approaches improve the robustness of IPS methods, they still cannot be applied directly to situations where for example π_e is a Dirac measure and π_b is not. In Chapter 3, we highlight some of the challenges faced by IPS methods for off-policy evaluation in the context of stochastic control.

Model-free estimation Model-free estimation methods, in the sense of reinforcement learning, usually rely on a dynamic programming principle (see e.g. [203]). Here, we will present one particular method that proved effective in Chapter 3, Fitted-Q-Evaluation (FQE). FQE uses an estimator based on the action-value function (or Q function) which is the value function depending on the current state and action, i.e.

$$Q_t(x, u) = \mathbb{E}[J_t(x_t, u_t; \pi_e) | x_t = x, u_t = u] \tag{2.2}$$

for all $t = 0, \ldots, T-1$ where $x \in S$ and $u \in U(x)$. Given an estimator for the Q-function $\widehat{Q_0}$ with parameters θ_0^*, the estimation target can be retrieved by integrating over the target policy, i.e.

$$\frac{1}{N} \sum_{i=1}^{N} \int_{u \in C} \widehat{Q_0}\left(x_0^i, u; \theta_0^*\right) p_e(u \mid x_t^i) du. \tag{2.3}$$

The regression function $\widehat{Q_0}$ can be any type of appropriate regressor such as neural networks, linear regressions, or random forests. For the episodic case, the fitting of the regressor is done backwards in time: For every $t = 0, \ldots, T-1$,

$$\widehat{Q_t}(\theta_t^*) = \min_{\theta} \frac{1}{N} \sum_{i=1}^{N} \left(\widehat{Q}_t\left(x_t^i, u_t^i; \theta\right) - y_t^i\right)^2$$

where

$$y_t^i = r_t^i + \int_{u \in C} \widehat{Q_{t+1}}(x_{t+1}^i, u; \theta_{t+1}^*) p_e(u \mid x_{t+1}^i) du$$

and $\widehat{Q_T} = 0$.

Model-based estimation Model-based estimation tries to fit a model for the underlying control problem's dynamic using the observational data. Given such a trained model, the estimate for the value function under the target policy is then obtained by either sampling the model using the target policy or finding a closed form solution dependent on the target policy. A model can be parametric and in that case the fitting of the model corresponds to the classical calibration problem. To give an example, in a setting with a finite state and action space, the calibration problem to consider could be the fitting of transition matrices. Another approach is to fit a non-parametric model for the problem's dynamic. This approach seems to be particularly appealing due to the recent advances in effectiveness for generative artificial intelligence. One example for the non-parametric approach is fitting a conditional GAN for the problem's dynamics and approximating the value function via a Monte Carlo estimator of GAN-generated trajectories under the target policy. Another non-parametric approach is the method described in Chapter 3.

2.4 The policy gradient algorithm

Over the last decades a multitude of different algorithms were proposed to solve reinforcement learning problems as described in Section 2.2.1. In this thesis, we consider policy gradient methods, a family of algorithms that proved to be effective for learning in stochastic control settings (see e.g. [116, 122, 187, 188, 204]). In this section, we introduce some variants of policy gradient algorithms and some algorithmic/numerical considerations within this type of algorithm. In particular, we discuss the choice of parametrisation for the policy, the estimation of the gradient, and the optimization of parameters via a gradient stepping algorithm. Based on the previous discussion, we then present some classical policy gradient algorithms. Finally, we will make some historical remarks on the development of the algorithm and discuss theoretical results and applications in the context of financial and numerical mathematics.

2.4.1 Parametrization of policy function and choice of features

The goal of parametrizing the policy is to restrict the policy search onto a finite dimensional parameter space rather than an infinite dimensional function space. This principle of reducing the search space by parametrization is a fundamental tool in statistics. This motivates that model selection for policy functions should follow similar considerations as described by statistical learning theory. The theory suggests three error sources that should be considered in model selection: Approximation, estimation, and optimization error (e.g. [212]).

Generally, the approximation error is defined as the difference between the minimized risk functional over a restricted model class and the minimized risk functional. It, thus, measures how rich a certain model class is with respect to its approximative quality. The estimation error is usually given as the difference between the risk functional evaluated at the empirical risk minimizer and the minimized risk functional over the restricted model class. Finally, for the optimization loss, one compares the empirical risk given by the output of an optimization algorithm with the minimal empirical risk in expectation.

The choice of a parametrization often impacts all of these errors and creates trade-offs between them. For example, in a simple regression case, a neural network can usually approximate more complex functions compared to a linear model with the same features. On the other hand, the optimization of a linear model tends to be simpler than for a neural network. However, by smartly extending the feature space (e.g. with a polynomial basis) one might be able to overcome the approximation deficits of the linear model. In the context of policy gradient methods, we expect similar trade-offs to exist and also that choosing a proper feature space can have a similar impact. However, we are currently not aware on any results in this domain. Common choices of parametrizations for deterministic policy functions are linear models (i.e. [88]) or neural networks (i.e. [92]). For randomized policies, it is common to look at parametrized families of densities (i.e. [25]).

As a final note, if we have a time-continuous reinforcement learning problem, one also needs to consider the choice of policy function in-between the time steps. In this thesis, we only consider piecewise constant time

interpolation. However, other choices for interpolation can be used as well. A more in-depth discussion of the role of discretization in time can be found in Chapter 6.

2.4.2 Gradient estimation: finite difference vs path-wise differentiation

An integral part of the policy gradient method is estimating the gradient of the value function in the direction of the parameters by sampling the environment. The problem of estimating gradients by simulation is well-known in the literature on Monte Carlo methods as sensitivity estimation. We will introduce two methods as described in [73]: Finite-differences and path-wise approximation. We will first consider the path-wise approximation before pointing out some limits of the procedure that may necessitate a finite difference scheme.

Consider a collection of random variables $\{Y(\theta), \theta \in \mathbb{R}^m\}$ defined on a probability space $(\Omega, \mathcal{F}, \mathbb{P})$. Fixing an $\omega \in \Omega$ and assuming $Y(\theta)$ is differentiable almost everywhere, we can define a path-wise gradient as $\nabla_\theta Y(\theta, \omega)$. In practice, $Y(\theta, \omega)$ is seen as the sample of a simulation algorithm at parameter θ and random number stream ω and $\nabla_\theta Y(\theta, \omega)$ the corresponding gradient. For our purposes, we are interested in when a path-wise estimator is unbiased for estimating the gradient of the expected value, i.e. when $\nabla_\theta \mathbb{E}[Y(\theta, \omega)] = \mathbb{E}[\nabla_\theta Y(\theta, \omega)]$. This question reduces to when the order of differentiation and integration can be exchanged and is equivalent to the uniform integrability of the difference quotients. [73] also offer a more practical approach to check this condition via additional continuity requirements on $Y(\theta)$. Finally, one should note that the availability of automatic differentiation toolboxes allows an efficient implementation of the algorithm with limited effort and favourable computational costs if using backward-mode differentiation [72]. In particular, it makes it possible to apply path-wise gradient estimation to complex policy parametrizations (e.g. [33]) or for complex model-dynamics (e.g. Chapter 5).

While path-wise gradient estimators are appealing, the method may suffer several issues. First, the method cannot be applied if the functional form of $Y(\theta)$ is unknown and, second, if the piecewise derivative does not exist. Furthermore, a problem with the exchangeability of limits can occur if the measure used for the expectation is parameter dependent and a parameter-dependent density function is not available. This example may occur for non-deterministic policies or in control mechanisms as described in the first part of Chapter 5. In these cases, we need to rely on finite difference estimators or estimators with finite difference component.

The idea of a finite difference estimator is to approximate derivatives via a difference quotient. We introduce it in the form of a central-difference estimator as it is known to reduce the bias in the approximation [73]. In the special case for θ where $m = 1$, let $\bar{Y}_n(\theta)$ denote the mean of n samples evaluated at θ. Then, the finite difference estimator is given by

$$\hat{\Delta}(n, h) = \frac{\bar{Y}_n(\theta + h) - \bar{Y}_n(\theta - h)}{2h}$$

where $h > 0$ is the step size. If $\alpha(\theta) = \mathbb{E}[Y(\theta)]$ is sufficiently smooth in θ, the bias of $\hat{\Delta}$ is of order $o(h)$. However, looking at the variance of $\hat{\Delta}$, one can see that in general the variance increases in decreasing h. Hence, there is a trade-off between bias and variance in choosing the step size h. One can show that with an optimal choice of increment h_n the convergence in a root means squared error is of order $O(n^{-\frac{1}{3}})$. If one reuses the same random seed for $\bar{Y}_n(\theta + h)$ and $\bar{Y}_n(\theta - h)$, one may improve the rate of convergence to $O(n^{-\frac{2}{5}})$. Note that these convergence rates are for general $Y(\theta)$. For continuous $Y(\theta)$, you can even achieve a rate of $O(n^{-\frac{1}{2}})$ reusing the same seed.

So far we have not commented on gradient estimation via finite differences, i.e. the case of $\theta = (\theta_1, ..., \theta_m) \in \mathbb{R}^m$. A straightforward approach is to compute estimates for partial derivatives. However, for a central difference estimator this requires the estimation of $\alpha(\theta)$ at $2m + 1$ values of θ and for a forward difference method at $m + 1$ values of θ[6]. Another method is to consider gradient estimation as a linear regression problem where the changes in parameters $h = (h_1, ..., h_m)$ are regressed onto the changes in Y denoted by ΔY. Thus, one would like to fit a

[6]Note that for a path-wise gradient estimator, estimation is only required at one value of θ.

model of the form

$$\Delta Y = \sum_{i=1}^{m} \beta_i h_i + \epsilon,$$

where β_i is the estimate for the partial derivative in the i-direction and ϵ is a residual error. Furthermore, another gradient estimator based on [61], is given by

$$\frac{1}{N} \sum_{k=1}^{N} \frac{\bar{Y}_n(\theta + hu_k) - \bar{Y}_n(\theta)}{h} u_k$$

where $N, n \in \mathbb{N}$, $h > 0$ and u_k is uniformly distributed on the m-dimensional unit sphere for each $k = 1, \ldots, N$. In the context of policy gradient methods, such a estimator was used in [88]. Finally, [158] proposes similar approximators based on Gaussian perturbations.

2.4.3 Gradient-based optimization algorithms

Gradient descent/ascent algorithms are well known methods for optimization of continuous functions (for a review, see e.g. [30]). These algorithms gained additional interest by their ability to fit machine learning algorithms to data. Gradient-based optimization algorithms are also a fundamental building block for policy gradient methods.

For a differentiable function $F : \mathbb{R}^m \to \mathbb{R}$ the goal of a gradient descent method is to find the minimum $\min_\theta F(\theta)$ by following the path of steepest descent starting from an initial point $\theta_0 \in \mathbb{R}^m$. This is done by following gradient steps of the form

$$\theta_{n+1} = \theta_n - \gamma_n \nabla F(\theta_n),$$

where ∇F is the gradient of F and $(\gamma_n)_{n=0}^{\infty}$ is a series of step sizes. Under certain conditions on $(\gamma_n)_{n=0}^{\infty}$ and F, it can be shown that the algorithm converges to a local minima or a saddle point. Under stronger assumptions even convergence to the global minimum can be established. A gradient ascent method is analogous with changed sign in the update step and a minimum is replaced by a maximum.

The simple gradient stepping scheme described above has multiple variations. Popular choices are methods with adaptive step sizes or inclusion of momentum terms. We will focus on two important modifications, namely, stochastic gradient methods and zeroth-order descent algorithms.

The stochastic gradient descent algorithm can be seen as a stochastic approximation of the gradient descent algorithm in the sense of [181]. In essence, we would like to optimize a function of the form $\mathbb{E}[F(X;\theta)]$, where X is a random variable and optimization is over θ. If the exact form of $\nabla_\theta \mathbb{E}[F(X;\theta)]$ is unknown, we will rely on a gradient estimate $\hat{\nabla}_\theta \mathbb{E}[F(X;\theta)]$. Possible candidates for estimates have been introduced in the previous section. The method is particularly popular for fitting machine learning models to data using a simple path-wise gradient estimator based on subsampling the available data. A benefit is that with large data sets using subsamples can lead to speed ups in training. Additionally, in the context of training neural networks and other non-convex optimization problems, it is believed that the introduction of noise due to subsampling the data has a regularizing effect helping to avoid saddle-points and bad local minima. However, in the context of the policy gradient problem studied in the Chapter 5, we need to introduce variance reduction techniques for gradient estimation to improve convergence behaviour of the algorithm. This points to a potential trade-off between the regularizing effect of variance and precision in gradient estimation.

In the special case where a finite difference method is used for gradient estimation, the descent algorithm is a zeroth-order optimization. This means that only function evaluations are used for optimization. A comprehensive monograph is [47]. The usage of zeroth-order optimization methods goes back to [120]. More recently, [158] have analysed a zeroth-order algorithm based on a smoothed gradient estimator using a Gaussian perturbation. In a convex setting, they find that such a zeroth-order method usually only needs at most d iterations more compared to standard gradient methods, where d is the dimension of the search space. For the non-convex case, complexity bounds are derived for the convergence of the algorithm to stationary points of the smoothed function.

2.4.4 The policy gradient algorithm

With the preparation from above, we can state the policy gradient algorithm. For that purpose, consider a reinforcement learning environment as described in the previous section on a probability space $(\Omega, \mathcal{F}, \mathbb{P})$ and a set of parametrized deterministic Markov policies $\pi(\theta) = (\mu(\theta_t))_{t=0}^{T-1}$ with parameters $\theta = (\theta_t)_{t=0}^{T-1} \in \Theta$ where Θ is an appropriate parameter space. The policies are assumed to be differentiable in the parameter. We will consider two different cases for the algorithm. In the first case, we assume that the functional forms of $(r_t)_{t=0}^{T}$ and f are known (or at least their derivatives with respect to the control variable) and that the functions are differentiable in the control and state variable for almost every $\omega \in \Omega$. In this case, we can state the method using a path-wise gradient estimator. In the second case, we assume that we are not able to calculate path-wise gradients and need to rely on a gradient estimate based on perturbed policies.

We state the first version of the policy gradient algorithm in Algorithm 1. Note that in this version we require knowledge of f and r where in a slight abuse of notation we highlight the dependence on realisations of random variables by including $\omega \in \Omega$. Given these considerations, we start the algorithm by initializing a parametrized family of policies. This policy is then updated by a step in the direction of a path-wise gradient estimator. The path-wise gradient estimator is established by sampling the control dynamics a certain number of times. We let the algorithm terminate after a fixed number of steps. Alternatively, one could also use some convergence criterium.

Algorithm 1 Policy gradient method with path-wise gradient estimator

Input: Stochastic control problem $\{S, C, U, W, p, f, r, T\}$ on $(\Omega, \mathcal{F}, \mathbb{P})$ where f and r are known and differentiable; Number of trajectories N_{traj}; Number of gradient steps N_{grad}; Gradient step-sizes $(\gamma_k)_{k=1}^{N_{\text{grad}}}$

Initialize: Parametrized policy $\pi(\theta^0) \leftarrow (\mu(\theta_t^0))_{t=0}^{T-1}$

for k in 1 to N_{grad} **do**

for l in 1 to N_{traj} **do**

Draw initial state $x_0^{l,k} = x_0(\omega_{l,k})$ where $\omega_{l,k} \in \Omega$

for t in 0 to $T-1$ **do**

Get $u_t^l = \mu(\theta_t^k)(x_t^{l,k})$; $x_{t+1}^{l,k} = f(x_t^{l,k}, u_t^l, \omega_{l,k})$; $r_t^{l,k} = r_t(x_t^{l,k}, u_t^l, \omega_{l,k})$

end for

Set

$$\nabla_\theta r^{l,k}(\theta^k) \leftarrow \nabla_\theta \left(\sum_{t=0}^{T-1} r_t^{l,k} \right)(\theta^k)$$

end for

Set $\widehat{\nabla_\theta r_k} \leftarrow \frac{1}{N_{\text{traj}}} \sum_{l=1}^{N_{\text{traj}}} \nabla_\theta r^{l,k}(\theta^k)$

Update

$$\theta^{k+1} \leftarrow \theta^k + \gamma_k \widehat{\nabla_\theta r_k}$$

end for

In the second case described above, rather than utilizing knowledge of the functional forms of f and r, we assume access only to a black-box simulator for the control problem, where for realizations of a chosen policy only the realized rewards and states are observed. Thus, we will need to rely on a zeroth-order estimator of the policy gradient. In Algorithm 2, we will use a gradient estimator based on perturbations with uniform samples of the unit-sphere (see Section 2.4.2). In the context of policy gradient methods, such an approach has been used in [88]. Note, that this is only one possible implementation of the policy gradient method with a perturbed gradient estimator. Other gradient estimators such as in [158] could be used as well. Note that estimating the gradient using a zeroth-order estimator does not require f and r to be differentiable nor continuous almost everywhere in the control variable, e.g. [73] and Chapter 5. However, lacking continuity may change the number

of samples required for estimation as explained in the previous section.

Algorithm 2 Policy gradient method with perturbed gradient estimator

Input: Sampling oracle $\mathcal{O}$ for the stochastic control problem $\{S, C, U, W, p, f, r, T\}$; Sampler $\mathcal{U}$ for sampling random vectors distributed uniformly over the unit sphere; Scaling parameter h; Number of trajectories N_{traj}; Number of gradient steps N_{grad}; Gradient step-sizes $(\gamma_k)_{k=0}^{K}$

Initialize: Parametrized policy $\pi(\theta^0) \leftarrow (\mu(\theta_t^0))_{t=0}^{T-1}$

for k in 1 to N_{grad} **do**
  for l in 1 to N_{traj} **do**
    for t in 0 to $T-1$ **do**
      Sample $\xi_t^{l,k}$ from $\mathcal{U}$ with the same dimensions as θ_t^k
      Denote by $\widehat{\rho_t^{l,k,+}}, \widehat{\rho_t^{l,k}}$ the cumulative rewards along the trajectories under the parametrizations

$$(\theta_0, \dots, \theta_t + h\xi_t^{l,k}, \dots, \theta_{T-1})$$

      and

$$(\theta_0, \dots, \theta_t, \dots, \theta_{T-1}),$$

      respectively, using the sampling oracle $\mathcal{O}$ starting from the initial state $x_0^{l,k}$.
      Set

$$\widehat{\nabla_{\theta_t^k} \rho_t^{l,k}} \leftarrow \frac{\widehat{\rho_t^{l,k,+}} - \widehat{\rho_t^{l,k}}}{h} \xi_t^{l,k}$$

    end for
  end for
  For every $t = 0, \dots, T-1$, set

$$\widehat{\nabla_{\theta_t^k} \rho_t^k} \leftarrow \frac{1}{N_{\text{traj}}} \sum_{l=1}^{N_{\text{traj}}} \widehat{\nabla_{\theta_t^k} \rho_t^{l,k}}$$

  and update

$$\theta_t^{k+1} \leftarrow \theta_t^k + \gamma_k B^{-1} \widehat{\nabla_{\theta_t^k} \rho_t^k}$$

end for

2.4.5 The role of initialization and exploration

It is well known in non-convex optimization tasks that initialization can have a strong impact on optimization outcomes (i.e. [74] in the context of neural network training). The same issues can be observed in many applications of policy gradient methods and one might encounter problems like vanishing gradients or convergence to bad local minima for poor choices of initialization. This in particular means that the choice of initialization might impact the policies that are explored and which are ignored.

One can also take a slightly more positive view on this behaviour: If some prior knowledge on what is a good policy exist, one can encode this knowledge via initialization. This can be achieved by either directly hard-coding the policy parameters or by supervised learning via a given policy function that is known to perform well (cf. imitation learning [108]). This avoids searching for policies in badly performing neighbourhoods and restricts the attention to satisfactory solutions. Hence, one can see initialization as a regularization via an appropriate prior. This can be seen in contrast to learning methods based on approximating the value function. In most problems it is unclear a priori what a good value function would look like while it seems more plausible to have

an idea of a good policy based for example on a previously-used heuristic.

Another difference between value-based methods and policy gradient methods is that policy gradient methods make gradual improvements in policy space rather than in value space. What we mean by this statement is that with a value-based method the policy can potentially alter strongly after each updating step while for the policy gradient method the rate of change in policy can be directly controlled. This control can be exerted via the choice of learning rate and leads to gradual adjustments in policy rather than large jumps.

A major challenge of the policy gradient method is that even if the learned policy is optimal, the performance of the parametrized policies in suboptimal state areas can be catastrophic. In other words, the learned policy might lack robustness. This issue was discussed in a slightly different context in [180], and can be avoided by taking into account additional information of the problem.

2.4.6 Literature on policy gradient methods

Due to the large body of literature on policy gradient methods, we focus on results most relevant to our research. We start by providing a historic perspective on the usage of policy gradient methods before highlighting some theoretical results and discussing recent applications in numerical mathematics and finance.

2.4.6.1 Historical remarks on policy gradient methods

The first application known to the authors occurred in [178], where the policy gradient method was applied to an inverted pendulum problem. In [14], the approach was used for an application to adaptive flight control systems. Interestingly, the author suggests a guided search approach where the experiments conducted in the environment are guided by prior knowledge of the system. In [78], the author argues that a continuous-time random search algorithm converges to the global minimum in the problem posed by [14]. Note that the control problems described in the earlier papers have deterministic systems.

Closely related to policy gradient methods is the field of stochastic approximation going back to [181] and [120]. Stochastic approximation theory provides a framework to understand recursive algorithms with noise and can be seen as a generalization of not only policy gradient methods but also value-based reinforcement approaches as can be seen for example in [128]. Direct investigations of zeroth-order descent methods can be found in [198] and its applications to control with policy gradient methods and neural network parametrization in [197].

Furthermore, policy gradient methods for control problems have been explored by the evolutionary algorithms community going back to at least [189]. A review of the evolutionary perspective can be found in [24] while a modern treatment is given in [184]. This paper summarizes the evolutionary algorithms for finding optimal policies in the following way: “At every iteration (generation), a population of parameter vectors (genotypes) is perturbed (mutated) and their objective function value (fitness) is evaluated. The highest scoring parameter vectors are then recombined to form the population for the next generation, and this procedure is iterated until the objective is fully optimized." ([184] p.2). Policy gradient methods clearly can be viewed within the scope of the described procedure, although the fitness aspect is often neglected in standard gradient policy methods. [184] provides further evidence on good performance of the method in modern standard reinforcement benchmark problems.

2.4.6.2 Theoretical results on policy gradient methods

The papers reviewed up until now did not have any noteworthy theoretical results, but rather focused on algorithmic issues and experimental performance. Recently, more theory-based results have been established. In the context of bandits, [61] show that regret grows sub-linearly by a factor of $O(\sqrt{n})$ where n is the number of optimization rounds. In [57], the authors show global convergence of the policy gradient method in an ergodic linear quadratic regulator setting and [88] generalizes this result to an episodic setting. [168] shows the convergence using proximal descent algorithms. These papers rely on a so-called gradient domination result

meaning that the difference in cost between the current control and the optimal control is dominated by the norm of the gradient of the cost at the current policy. If the cost function is smooth, this is enough to establish convergence to the global minimum [172]. In the LQC setting, this is not generally the case and requires a notion of almost smoothness that combined with the stability of the state covariance matrix yields the result. This first gives performance bounds if the gradient is observed exactly and can then be extended to the approximate case by showing that with enough samples the estimated gradient is sufficiently close to the actual gradient and that convergence occurs with a similar rate if the gradient has bounded perturbations[7]. Finally, [25] provides global convergence guarantees for more complex dynamics for the tabular and approximate case. However, their upper bounds become trivial if the action space is infinite since the Kullback–Leibler divergence between the learned policy and the optimal policy is upper bounded using the cardinality of the action set.

Since most of the results described in the previous paragraph are restricted to a LQC setting, we want to mention some related considerations. First, [179] and [144] argue that the LQC setting should be used as a regular benchmark for reinforcement learning algorithms, because of its well-known theoretical properties while still capturing a wide range of important features of general control problems. Furthermore, [144] shows competitive performance of a simple policy gradient method in the most common continuous action space benchmarks and the LQC setting.

2.4.6.3 Policy gradient methods in finance and numerical mathematics

Finally, we would like to highlight some developments with regards to policy gradient methods using path-wise gradient estimates where the policy is mainly parametrized by neural networks. This approach has recently gathered considerable attention within the mathematical finance and numerical analysis community. As a starting point, the paper by [92] explores the usage of deep neural networks for solving stochastic control problems in an applied context. For training, the authors used a policy gradient method with a path-wise gradient estimate (without explicitly stating the fact). Note that [197] had a similar objective but relying on zeroth-order estimations of the gradient. Based on the findings in [92], several papers using the same methodology and following the applied focus have developed. In numerical analysis, the method has been used to solve high-dimensional partial differential equations via simulating their related backward stochastic differential equation in [93]. In mathematical finance, the method has been applied to model-free derivative hedging [33] and to optimal execution of assets with market orders [132] among other applications.

[7] In Chapter 6, we establish gradient domination and almost smoothness for a time-continuous, episodic and stochastic setting.

Chapter 3

K-Nearest-Neighbor Resampling for Off-Policy Evaluation in Stochastic Control

3.1 Introduction

In reinforcement learning (RL), off-policy evaluation (OPE) deals with the problem of estimating the value of a target policy with observations generated from a different behavior policy. OPE methods are typically applied to sequential decision making problems where observational data is available but experimentation with the environment directly is not possible or costly. More broadly, OPE methods are a widely researched subject in RL (see, e.g., [64, 210, 214] for recent overviews), however, relatively little attention has been paid to stochastic environments where the stochasticity depends on the chosen actions and state and action spaces are continuous. For example, common benchmark problems are either deterministic or have finite state and/or action spaces (see, e.g., [64, 214]).

Notwithstanding this, stochastic control problems are precisely concerned with the setting where a decision process affects random transitions. Stochastic control is a field closely related to reinforcement learning and its methods have been applied to a wide range of high-stakes decision-making problems in diverse fields such as operations research [65, 147], economics [105, 118], electrical engineering [44, 153], autonomous driving [221] and finance [38, 186]. In the stochastic control literature, optimal policies are often represented as deterministic feedback policies (i.e., as deterministic functions of the current state) and, in the episodic case, are non-stationary due to the impact of a finite time-horizon.

Stochastic control environments pose a challenging setting for OPE methods. For example, classical methods like importance sampling (IS) [173] struggle with deterministic target policies in continuous action spaces due to the severe policy mismatch between the target and the behavior policy (see, e.g. [152], for more details on the difficulty of estimation in such a regime in general). This policy mismatch can lead to a high variance in the estimate (see, e.g., [5, 53, 142, 143, 174, 185]), which is exacerbated by the stochastic control environments' inherent stochasticity. Furthermore, IS requires knowledge of the distribution of the behavior policy. Learning this distribution, known as imitation learning, in a continuous action space of a policy functionally depending on a continuous and potentially high-dimensional state adds a further complication to this class of methods. Lastly, OPE methods not based on importance sampling often assume for their theoretical guarantees that the data sets used for value estimation contain only identically distributed and independent (iid) transitions (see, e.g., Fitted Q Evaluation [131], minimax V learning [58], model-free Monte Carlo [62, 63], DualDice [156]). However, for episodic control problems this is an unrealistic assumption, as previous interaction with the environment should

also be episodic and not rely on a potentially non-existent oracle that generates iid transitions.

In this chapter, we propose a novel procedure for off-policy evaluation based on a nearest-neighbor resampling algorithm. Assuming continuity in the rewards and the state transitions, our procedure exploits that similar state/action pairs (in a metric sense) are associated with similar rewards and state transitions. This addresses the counterfactual estimation problem of evaluating the performance of a target policy using a data set sampled under a different behavior policy without knowing or estimating the distribution of this policy. Similar to Monte Carlo methods, the resampling algorithm simulates trajectories and has a provably vanishing mean squared error (MSE) under episodic sampling. Compared to other OPE methods, our algorithm does not require optimization, can be efficiently implemented via tree-based nearest neighbor search and parallelization and does not explicitly assume a parametric model for the environment's dynamics. These properties make the proposed resampling algorithm particularly useful and directly applicable for stochastic control environments.

Table 3.1: In this table, we summarize statistical results on commonly used model-based OPE methods. We report whether the results take into account episodic sampling, whether they apply to continuous state/ action spaces, whether model-inherent stochasticity was considered, and whether statistical consistency was proved.

Method	Episodic	Cont. State	Cont. Action	Stochastic	Consistency
Mannor et al. 2007 [145]	No	No	No	Yes	No
Fonteneau et al. 2010 [62]	No	Yes	Yes	Yes	No
Liu et al. 2018 [140]	Yes	Yes	No	Yes	No
Gottesman et al. 2019 [76]	Yes	Yes	No[1]	No	Partially
Nachum et al. 2019 [156]	No	Yes	Yes	Yes	Partially
Yin & Wang 2020 [223]	Yes	No	No	Yes	Yes
Uehara & Sun 2021 [211]	No	Yes	Yes	Yes	Yes
Wang et al. 2024 [218]	Partially	No	No	Yes	Yes

Contributions

1. We introduce a novel nearest-neighbor based resampling algorithm for off-policy evaluation in stochastic control environments that can generate trajectories without exact knowledge of the behavior policy and does not rely on estimating a parametric model for the dynamics (see Section 3.2). It is faster than comparable existing algorithms and parallelizable.

2. We prove that the MSE of this method vanishes for episodic sampling, under only weak assumptions (Section 3.3). For model-based OPE methods, this is to our knowledge the first consistency result under episodic sampling for non-deterministic environments with continuous state-action spaces. An overview of statistical results for model-based OPE methods can be found in Table 3.1.

3. To prove consistency of the resampling algorithm, we extend a well known statistical result known as Stone's Theorem (see [87], Theorem 4.1), which gives conditions for the consistency of locally weighted regressions. Our generalization consists of including the episodic setting and the counterfactual element of OPE. This result can be found in Theorem 2 in Section 3.3.

4. We demonstrate the method's effectiveness on various stochastic control environments including a linear quadratic regulator, trade execution in limit order books, and online stochastic bin packing. These environments vary in their control mechanisms, sources of uncertainty, areas of application and dimensionality. We show competitive performance of the resampling algorithm compared to baselines (see Section 3.4).

Related Work Our method is most closely related to the model-free Monte Carlo (MFMC) algorithm proposed in [62, 63], where a nearest neighbor method is used for trajectory generation. Under fairly generic assumptions, this method leads to a computational complexity of $\mathcal{O}(n^2)$ in the number of observed sampling episodes n. In contrast, our method can utilize efficient tree-based nearest neighbor search algorithms, which leads to a more favourable computational complexity of $\mathcal{O}(K_n n \log(n))$ where the number of nearest neighbours under consideration is of the order $o(n)$ [2]. This translates into improvements in computational costs of an order of magnitude even for moderately-sized data sets (see Section 3.4). Furthermore, our algorithm allows for almost complete parallelization in the generation of trajectories. On the theory side, [62, 63] provide point-wise error bounds on the bias and variance depending on the distances of the nearest neighbors, while not studying convergence properties. In contrast, we establish our algorithm's consistency, which is global in the state domain. Crucially, their results only hold for data sets containing iid transitions, while our results allow for data sets sampled in episodes. Note that our analytic approach can be modified to establish consistency for the MFMC. A more detailed comparison between our resampling algorithm and MFMC can be found in the supplementary material.

In [76], a version of the MFMC algorithm was combined with a parametric model approach to improve performance for smaller data sets. Consistency is derived for a deterministic environment by using the stability of the deterministic dynamical system, and under the critical assumption that for any fixed action the distance to the nearest neighbor state vanishes as more data becomes available[3]. Crucially, they only consider nearest neighbor matching in the state space and require exact matches in the action space (Assumption 1 in [76]). A simple probabilistic argument shows, that this assumption excludes data generated by not fully atomic behavior policies on continuous action spaces and practically limits the results to discrete action spaces.

Finally, there is a broader literature on nearest neighbor search and other local weighting approaches in the context of learning for dynamical systems. In [130], the authors introduced a nearest neighbor resampling technique similar to ours for simulating hydrological time series. Their method has attracted considerable interest in hydrology, where several other papers apply it in this context (e.g.,[175, 177, 209]). This literature, however, does not consider interventions and counterfactual effects, and does not provide any theoretical analysis of the algorithm. Other examples of nearest neighbor methods in reinforcement learning include imitation learning [166], memory-based learning [7, 8, 12, 107, 146, 149], and data augmentation [136].

3.2 Description of algorithm

Notation For a metric space (S, d), $C(S, S)$ will denote the space of continuous functions with respect to d from S onto S. We will use the following multi-index notation: $i^k = (i_0, \ldots, i_k)$, $t^k = (t_0, \ldots, t_k)$, in particular $(i^k, t^k) = ((i_0, t_0), \ldots, (i_k, t_k))$. For any two multi-indexes i^k and j^l with $l \in \mathbb{N}$, denote $[i^k, j^l] = (i_0, \cdots, i_k, j_0 \cdots, j_l) \in \mathbb{N}^{k+l+2}$ and, for $l < k$, $i^{l:k} = (i_l, \cdots, i_k) \in \mathbb{N}^{k-l+1}$. We define the summation $\sum_{i^k, t^k}^{n,T} \coloneqq \sum_{i_1=1}^{n} \sum_{t_1=0}^{T-1} \cdots \sum_{i_k=1}^{n} \sum_{t_k=0}^{T-1}$, where $T, n \in \mathbb{N}$. The operator $|\cdot|$ applied to a set denotes its cardinality. If applied to a multindex, $|i^k|$ denotes the cardinality of the set $\{i_j\}_{j=0}^{k}$. $I.$ defines an indicator function. For any set S, $\mathcal{V}(S)$ denotes the uniform distribution on S.

Setting We are given a data set $\hat{\mathcal{D}} \coloneqq \{(X_i, U_i, R_i)\}_{i=1,\ldots,n}$, where the state process $X_i = (X_{i,t})_{t=0,\ldots,T}$, the action process $U_i = (U_{i,t})_{t=0,\ldots,T}$ and the reward process $R_i = (R_{i,t})_{t=0,\ldots,T}$ are stochastic processes on $\mathcal{X} \subset \mathbb{R}^{d_1}$, $\mathcal{U} \subset \mathbb{R}^{d_2}$ and $\mathbb{R}$, respectively, with $d_1, d_2 \in \mathbb{N}$. Here, $\mathcal{X}$ denotes the state space and $\mathcal{U}$ the action space. For each i, $((X_{i,t}, U_{i,t}, R_{i,t}))_{t=0,\ldots,T}$ is called a sample episode and we assume that $\hat{\mathcal{D}}$ contains n episodes,

[2] In our experiments, we choose K_n in the order of $\mathcal{O}(n^{\frac{1}{4}})$ based on empirical observations. A heuristic choice for the nearest neighbor growth rate is common (i.e., [130]) and regression theory suggests a rate smaller than 1/2 (i.e., Theorem 6.2 in [87] suggests the rate $\mathcal{O}(n^{\frac{2}{d+2}})$ where $d \geq 3$ is the dimension of the feature space).

[3] In statistics, this is commonly considered a statement that should be proven (e.g., [87], Lemma 6.1).

which are rolled out independently and with the same behavior policy. For all t, the behavior policy is given by the distribution of $U_{i,t}$ conditioned on $X_{i,t}$. Furthermore, denote the data set without the last period as $\mathcal{D} \coloneqq \hat{\mathcal{D}} \setminus \{(X_{i,T}, U_{i,T}, R_{i,T})\}_{i=1,\dots,n}$.[4]

Each episode is generated by a random dynamical system where we use a similar formulation as in [22, 23, 26]. Consider $S \coloneqq \mathcal{X} \times \mathcal{U}$ equipped with a metric d and let $\mathcal{S}$ be the Borel σ-algebra of S. Let $\Gamma \subset C(S,S)$, a set of uniformly equicontinuous functions, be endowed with a σ-algebra Σ such that $(\gamma, x) \to \gamma(x)$ is $(\Gamma \times S, \Sigma \otimes \mathcal{S})$-$(S, \mathcal{S})$ measurable and let $\mathbb{Q}$ be a probability measure on (Γ, Σ). On a probability space $(\Omega, \mathcal{F}, \mathbb{P})$, let $(\alpha_t)_{t=0}^{T-1}$ be a sequence of iid random functions from Γ distributed according to $\mathbb{Q}$. For an initial S-valued random variable (X_0, U_0), independent of the sequence $(\alpha_t)_{t=0}^{T-1}$, a sample episode is generated according to $(X_{t+1}, U_{t+1}) = \alpha_t(X_t, U_t)$ or equivalently by $(X_{t+1}, U_{t+1}) = \alpha_t \alpha_{t-1} \dots \alpha_0(X_0, U_0)$. We assume that the distribution of X_0 is known and we denote it by ν_0.

For the process $R = (R_t)_{t=0,\dots,T}$, we assume that there exists a sequence of iid real random variables $(\epsilon_t)_{t=0}^{T}$, independent of the system noise, distributed on $[-M, M]$ for $M \in \mathbb{R}$ with $\mathbb{E}[\epsilon_t] = 0$, and a continuous function $r : S \to [-M, M]$ such that $R_t = r(X_t, U_t) + \epsilon_t$. For sample i, denote with $\epsilon_{i,t}$ and $\alpha_{i,t}$ the corresponding random variable and random function of this sample. We call this dynamical system the sampling system. While we are not aware of any general result relating this set-up to Markov decision processes (MDP), we provide some discussion in the supplementary materials and note that the setting we consider here is more general compared than the settings considered in [62, 63, 76].

Furthermore, we define a second dynamical system, on the same space as the first system, as follows. Given a sequence of iid random functions $(\alpha_t)_{t=0}^{T-1}$ as above, consider a family of feedback policies $u = (u_t)_{t=0}^{T}$ with $u_t : \mathcal{X} \to \mathcal{U}$. The goal is to define a version of the dynamical system where the control strategy u is used instead of the behavior policy. For that purpose, define the sequence $(\alpha_t|_u)_{t=0}^{T-1}$ where $\alpha_t|_u = (\Pi(\alpha_t), u_t(\Pi(\alpha_t)))$ and $\Pi : \mathcal{X} \times \mathcal{U} \to \mathcal{X}$ with $\Pi(x,u) = x$ is the projection state-action pair on the state. For an initial S-valued random variable $(X_0, u_0(X_0)))$, where X_0 is the same as in the sampling system and is independent of the sequence $(\alpha_t|_u)_{t=0}^{T-1}$, an episode is generated by $(\widehat{X}_{t+1}, u_{t+1}(\widehat{X}_{t+1})) = \alpha_t|_u(\widehat{X}_t, \widehat{U}_t)$ or equivalently by $(\widehat{X}_{t+1}, u_{t+1}(\widehat{X}_{t+1})) = \alpha_t|_u \alpha_{t-1}|_u \dots \alpha_0|_u(X_0, u_0(X_0))$. Let $(\epsilon_t)_{t=0}^{T-1}$ be a sequence of iid real random variables as above, then define the stochastic process $R|_u = (R_t|_u)_{t=0,\dots,T}$ by $R_t|_u = r(\widehat{X}_t, u_t(\widehat{X}_t)) + \epsilon_t$.

We call this dynamical system the counterfactual system and for a given realisation of the noise $(X_0, (\alpha_t)_{t=0}^{T-1}, (\epsilon_t)_{t=0}^{T})$, we call the generated episode of the counterfactual system the counterfactual path. Note that for each episode generated by the sampling system a counterfactual path can be identified. Our algorithm has the target to consistently estimate the cumulative reward under the counterfactual system with target policy u:

$$m(u) \coloneqq \mathbb{E}\Big[\sum_{t=0}^{T} R_t|_u\Big]. \tag{3.1}$$

Remark 5. *A commonly made simplifying assumption in the OPE literature is that data consist of iid transitions $\{X_i, U_i, R_i, X_i'\}_{i=1}^n$, where X_i' denotes the subsequent state [62, 63, 94, 131]. Note that the samples in our setting are not assumed iid, but may exhibit an episodic dependence structure, which poses a considerable complication in the analysis of OPE methods not relying on importance weights. However, considering episodic data is often more realistic as in most practical situations data is gathered in episodes rather than from one-transition oracles.*

Remark 6. *Our algorithm and the corresponding consistency results allow for two generalizations of targets: First, we can allow the reward process to depend not only on the current state and action, but also the subsequent state, i.e., we can define the reward process as $\widetilde{R} = (\widetilde{R}_t)_{t=0,\dots,T}$ by a sequence $(\epsilon_t)_{t=0}^{T}$ as above and a continuous function $\widetilde{r} : \mathcal{X} \times \mathcal{U} \times \mathcal{X} \to [-M, M]$, such that $\widetilde{R}_t = \widetilde{r}(X_t, U_t, X_{t+1}) + \epsilon_t$. Second, we can consistently estimate the expectation of functions of counterfactual paths. More precisely, for a known continuous*

[4] While our algorithm and our theoretical results could be formulated slightly more data efficient if we use $\hat{\mathcal{D}}$ instead of $\mathcal{D}$, we omit this detail to simplify notation.

function $g : \mathcal{X}^T \to [-M, M]$, we can consistently estimate $\mathbb{E}[g|_u]$, where $(\widehat{X}_t)_{t=0}^T$ is a counterfactual path and $g|_u = g(\widehat{X}_0, \ldots, \widehat{X}_T)$ a random variable. This extension allows for example the consistent estimation of path-dependent risk measures using our algorithm. Since the two extensions can be treated analogously to the original rewards, both in the algorithm and the main theorem, we omit them for readability.

K-Nearest Neighbors and K-Nearest Neighbor Paths Given a tuple $(x_0, v_0) \in S$, the reordered version of a data set $\mathcal{D}$ is given by $\{(X_{(i,nT)}, U_{(i,nT)}, R_{(i,nT)})(x_0, v_0)\}_{i=1}^{nT}$, where the ordering is according to increasing values of $d\left((x_0, v_0), (X_{i,t}, U_{i,t})\right)$. For simplicity, we assume for this ranking and all the subsequent ones that no ties occur. For $K \in \mathbb{N}$ with $K < n$, $(X_{i,t}, U_{i,t})$ is a K-nearest neighbor of (x_0, v_0) in $\mathcal{D}$, if for the corresponding sample in the reordered data set $(X_{(j,nT)}, U_{(j,nT)})(x_0, v_0) = (X_{i,t}, U_{i,t})$ it holds that $j \leq K$. For both the reordered data sets and nearest neigbhors, we omit the dependence on (x_0, v_0) for readability, if it is clear from the context.

For $\tau = 0, \ldots, T$, define a K-nearest neighbor (K-NN) path for a given initial state $x_0 \in \mathcal{X}$, a feedback policy u, and data $\mathcal{D}$ by the following construction. First, reorder $\mathcal{D}$ according to increasing values in $d\left((x_0, u_0(x_0)), (X_{i,t}, U_{i,t})\right)$ and consider the reordered version $\{(X_{(i,Tn)}, U_{(i,Tn)}, R_{(i,Tn)})\}_{i=1}^{Tn}$. For $k_0 \leq K$, identify the k_0-nearest neighbor $(X_{(k_0,Tn)}, U_{(k_0,Tn)})$ with the original sample from the data set and denote it by $(X_{i_0,t_0}^{(k_0)}, U_{i_0,t_0}^{(k_0)})$. Next, consider $(X_{i_0,t_0+1}^{(k_0)}, u_1(X_{i_0,t_0+1}^{(k_0)}))$ and reorder the data set $\mathcal{D}$ according to increasing values in $d\big((X_{i_0,t_0+1}^{(k_0)}, u_1(X_{i_0,t_0+1}^{(k_0)})), (X_{i,t}, U_{i,t})\big)$ and denote it by $\{(X_{(i,Tn)}^{(k_0)}, U_{(i,Tn)}^{(k_0)}, r_{(i,Tn)}^{(k_0)})\}_{i=1}^{Tn}$. For $k_1 \leq K$, again identify $(X_{(k_1,Tn)}^{(k_0)}, U_{(k_1,Tn)}^{(k_0)})$ and denote it by $(X_{i_1,t_1}^{(k_0,k_1)}, U_{i_1,t_1}^{(k_0,k_1)})$. We repeat this procedure τ times where for any $q = 1, \ldots, \tau$, k^{q-1} and k_q we search for the k_q-nearest neighbor of $(X_{i_{q-1},t_{q-1}+1}^{k^{q-1}}, u_q(X_{i_{q-1},t_{q-1}+1}^{k^{q-1}}))$ in the set $\mathcal{D}$ and then identify the k_q-nearest neighbor with $(X_{i_q,t_q}^{k^q}, U_{i_q,t_q}^{k^q})$. For an initial state $x_0 \in \mathcal{X}$, $((X_{j_p,s_p}, U_{j_p,s_p}))_{p=0}^{\tau}$ is a K-NN path of length τ if there exists a sequence $k^\tau = (k_0, k_1, \ldots, k_\tau)$ with $k_q \leq K$ for all $q = 0, \ldots, \tau$ such that $((X_{j_p,s_p}, U_{j_p,s_p}))_{p=0}^{\tau} = ((X_{i_q,t_q}^{k^q}, U_{i_q,t_q}^{k^q})((x_0, u_0(x_0))))_{q=0}^{\tau}$. Equivalently, we say (j^τ, s^τ) is a K-NN path. Since we assume that each nearest neighbor attribution is unique, we can also uniquely identify each K-NN path by the sequence k^τ for any initial x_0. Thus, for any initial x_0 each K-NN path can be uniquely written as $(X_t^{k^\tau}, U_t^{k^\tau}, R_t^{k^\tau})_{t=0,\ldots,\tau}$. We provide a visualization of a K-NN path in Figure 3.1.

K-Nearest Neighbor Resampling for OPE Given the data set $\mathcal{D}$ and the feedback policy u, the goal of the K-nearest neighbor resampling algorithm is to estimate the expected cumulative reward $m(u)$. We achieve this by randomly sampling K-NN paths and averaging the cumulative rewards along the sampled K-NN paths. This procedure is explicitly formulated in Algorithm KNNR.[5]

The algorithm produces an estimate of the value by generating trajectories without assuming a parametric model. Instead, we exploit the fact that under sufficient regularity of the underlying stochastic system, state-action pairs that are close in the chosen metric are related to similar transitions and rewards. This also implies that the algorithm does not require optimization. Furthermore, it is directly applicable to problems with continuous state and action spaces and no detailed knowledge of the sampling policy is required. In particular, it can handle the distribution mismatch between random behavior policies and deterministic target policies and it allows for the evaluation of non-stationary policies. The algorithm also allows, with slight modifications, the estimation of path dependent quantities (see Remark 6)

The choice of the hyperparameters K and l, the number of resampled paths, allows for flexibility in the bias-variance trade-off of the estimation. Our theoretical results require $K_n \to \infty$ where $K_n = o(n)$ and that $l_n \to \infty$ as $n \to \infty$. As mentioned earlier, for K_n, standard regression theory suggests a dimension dependent rate of $\mathcal{O}(n^{\frac{2}{d+2}})$ where the dimension $d \geq 3$ to optimally balance variance and bias (Theorem 6.2 in [87]). In the regression case, this choice of K_n leads to minimax optimal convergence for the class of Lipschitz-continuous functions with a fixed Lipschitz constant (i.e., [87]). In practice, a heuristic choice of K is common (i.e., [130]).

[5] To simplify readability, some algorithmic details are omitted in Algorithm KNNR (e.g., tree-based nearest neighbor search). A more detailed version can be found in the supplementary materials.

Finally, the averaging over resampled paths makes the algorithm behave similarly to Monte Carlo methods. An example for this is the empirical observation in Section 3.4 that the standard deviation of the estimation vanishes approximately with the order $O(n^{-0.5})$, if l_n is of the order $O(n)$[6]. The Monte Carlo-like behavior makes the KNNR algorithm particularly well-suited for problems with inherent stochasticity.

Algorithm KNNR K-nearest neighbor resampling for OPE

Input: Data set $\mathcal{D}$; target policy u; nearest neighbors parameter K; metric d on S; number of resampled trajectories l

1: $R \leftarrow 0$
2: **for** j from 1 to l **do**
3: $r \leftarrow 0$
4: Sample an initial state $X_{j,0}^{\mathrm{NN}}$ from ν_0
5: **for** s from 0 to T **do**
6: Randomly choose $\mathcal{K}_s$ where $\mathcal{K}_s \sim \mathcal{V}(\{1,\dots,K\})$ and $\mathcal{K}^s \leftarrow (\mathcal{K}_0,\dots,\mathcal{K}_s)$
7: Find the $\mathcal{K}_s$-nearest neighbor of $(X_{j,s}^{\mathrm{NN}}, u_s(X_{j,s}^{\mathrm{NN}}))$ in $\mathcal{D}$ under d and denote them by $(X_{i_s,t_s}^{\mathcal{K}^s}, U_{i_s,t_s}^{\mathcal{K}^s})$ with the corresponding reward $R_{i_s,t_s}^{\mathcal{K}^s}$
8: Set $r \leftarrow r + R_{i_s,t_s}^{\mathcal{K}^s}$ and $X_{j,s+1}^{\mathrm{NN}} \leftarrow X_{i_s,t_s+1}^{\mathcal{K}^s}$ if $s < T$
9: **end for**
10: $R \leftarrow r + R$
11: **end for**

Output: $\frac{R}{l}$

Relying on nearest neighbor searches imposes some limitations on the algorithm. First, at evaluation, we are required to have the data set stored and the computational cost may be relatively high, depending on the size of the data set. As mentioned before, however, our algorithm does not require optimization and, in contrast to similar existing methods, allows for more efficient evaluation using tree-based nearest neighbor search and almost complete parallelization. Also, nearest neighbor search in high-dimensional spaces suffers from the curse of dimensionality. This problem can be potentially alleviated by using appropriate lower dimensional representations [166] and a suitable choice of metric. Furthermore, we need to assume that our decision problem has a Markov structure. This restriction can be potentially circumvented if a sufficient summary statistic of the paths can be identified. The formulation of the algorithm and the theoretical results require the target policies to be deterministic and the behavior policies stationary. These restrictions might be avoidable in future results. Finally, only weak regularity assumptions on transitions and rewards, and a sufficient coverage by the behavioral policy are required.

Remark 7. *On an algorithmic level, a crucial difference of Algorithm KNNR shown above to the MFMC method proposed by [62, 63] occurs in line 8. At this point of the algorithm, MFMC would remove the used sample from the data set while KNNR keeps the data set unchanged. This seemingly small difference leads to the favourable computational properties of the KNNR compared to the MFMC as specified in the introduction.*

3.3 Consistency of K-nearest neighbor resampling

In this section, we introduce our theoretical results on the consistency of the estimator implemented by Algorithm KNNR. This means we can guarantee under certain weak assumptions in the setting previously introduced that the mean-squared error related to the off-policy value estimation by K-nearest neighbor resampling goes to zero as the number of sampled episodes n increases. Note that this is in particular equivalent to the bias and variance

[6]For rates of order $o(n)$, the standard deviation decays slower in experiments.

of value vanishing as more data becomes available. To our knowledge, this is the first consistency result under episodic sampling for non-deterministic environments with continuous state-action spaces.

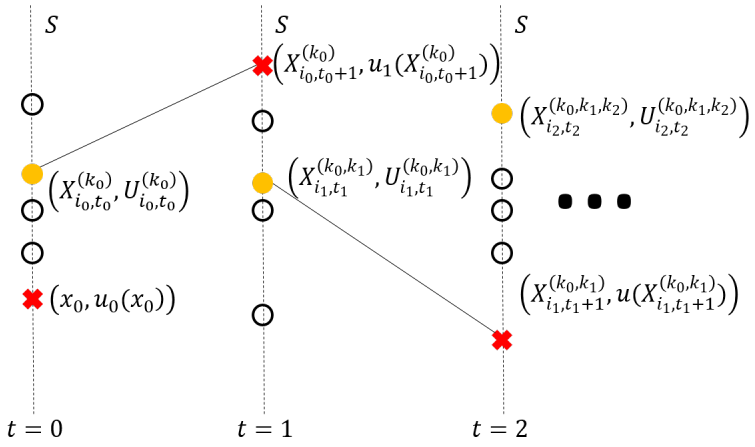


Figure 3.1: Visualization of the K-NN path with $k^3 = (3, 2, 4)$ where S is projected onto a line for each t and metric distances are assumed to be preserved in this one dimensional representation. The circles represent samples and the filled circle denotes the sample that is part of the K-NN path. The crosses represent the starting point on the first line and the transitions from samples on the K-NN path on the subsequent lines.

We will first provide the assumptions needed and introduce a regression notation for the algorithm's output, before stating our main result and providing the idea of the proof. A full proof is given in Section 3.5 with some detailed technical steps in the supplementary materials.

Assumption 1.

1. *The probability of a tie occurring in any nearest neighbor search is zero.*
2. *Let $(\pi_t)_{t=0}^T$ be a family of probability measures where π_t is the measure of (X_t, U_t) and let $\pi_t|_x$ denote the marginal distribution in the first component. Assume that for all t and for all $x \in supp(\pi_t|_x)$ there exists a $\tilde{t} < T-1$, such that $(x, u_t(x)) \in supp(\pi_{\tilde{t}})$.*

The first assumption is common in the analysis of nearest neighbor-based methods (e.g., [87], Chapter 6) and can usually be replaced by introducing a tie-breaking mechanism. We omit treating this case to simplify argumentation and note that this assumption is satisfied for any K-NN search where the matching variables have non-atomic probability distributions. The second condition guarantees sufficient exploration of the behaviour policy and is comparable with common coverage assumptions (cf. [211] for an overview).

To state the main result, we will introduce the K-NN path resampling estimator (KNNR) for the value function that corresponds to the estimator produced by Algorithm KNNR. With the notation introduced in the previous section, $R_\tau^{k^\tau}(x; u)$ denotes the reward in period τ of the K_n nearest neighbor path k^τ starting at the state $x \in \mathcal{X}$ under the target policy u. Draw independently $l \in \mathbb{N}$ sequences $\mathcal{K}_j^T = (\mathcal{K}_{0,j}, \ldots, \mathcal{K}_{T,j})$ of length $T+1$, where $\mathcal{K}_j^T$ is a sequence of iid random variables distributed uniformly on $\{1, \ldots, K_n\}$ and $j \leq l$ with $j \in \mathbb{N}$. Then, the resampling regression is given by $m_n^{\text{KNNR}}(x, u, \tau, j) = \sum_{k^\tau}^{K_n} R_\tau^{k^\tau}(x; u) I_{k^\tau = \mathcal{K}_j^\tau}$, where $\sum_{k^\tau}^{K_n} = \sum_{k_0=1}^{K_n} \cdots \sum_{k_\tau=1}^{K_n}$ and the resampling estimator for $\mathbb{E}[R_\tau|_u]$ is $m_{l,n}^{\text{KNNR}}(u, \tau) = \frac{1}{l} \sum_{j=1}^{l} m_n^{\text{KNNR}}(\tilde{X}_j, u, \tau, j)$, where $\tilde{X}_j$ is sampled independently from the initial state distribution. Then, the KNNR estimator for the value of policy u is given by

$$m_{l,n}^{\text{KNNR}}(u) = \sum_{\tau=0}^{T} m_{l,n}^{\text{KNNR}}(u, \tau). \tag{3.2}$$

With this definition, we can state that the estimator calculated by KNNR is consistent:

Theorem 1. *Given the setting described in Section 3.2 and under Assumption 1, $m_{l,n}^{KNNR}(u)$ is consistent at each family of feedback policies $u = (u_t)_{t=0}^T$ with $u_t : \mathcal{X} \to \mathcal{U}$, i.e.,*

$$\lim_{n\to\infty} \mathbb{E}\left[\left(m_{l_n,n}^{KNNR}(u) - m(u)\right)^2\right] = 0 \tag{3.3}$$

with $l_n \to \infty$, $K_n \to \infty$ and $\frac{K_n}{n} \to 0$.

The proof follows in three steps. We will briefly describe each of these steps, but will refer the details to Section 3.5.

3.3.1 Local averaging regression estimators

The first step is to show that a more general class of estimators based on local-averaging regressions is consistent under certain conditions on the local weights. Denote $D = \{(X_i, U_i)\}_{i=1}^n$. Define the local averaging regression for $R_\tau|_u$ for the initial condition x and policy u by $\widehat{m}_n(x, u, \tau) = \sum_{i^\tau,t^\tau}^{n,T} W_{n,i^\tau,t^\tau}(x, u) R_{i_\tau,t_\tau}$. The weights are for each (i^τ, t^τ) real-valued functions of x, u and all the state action pairs D (i.e., $W_{n,i^\tau,t^\tau}(x, u) = W_{n,i^\tau,t^\tau}(x, u, D) \in \mathbb{R}$). Note that for the local-averaging regression we sum τ-times over the rewards of the entire data set and assign each summand a unique weight. The estimator for $\mathbb{E}[R_\tau|_u]$ is given by $\widehat{m}_{l,n}(u, \tau) = \frac{1}{l}\sum_{j=1}^l \widehat{m}_n(\tilde{X}_j, u, \tau)$, where $\tilde{X}_j$ is sampled independently from the initial state distribution. The local-averaging regression estimator for $m(u)$ is $\widehat{m}_{l,n}(u) = \sum_{\tau=0}^T \widehat{m}_{l,n}(u, \tau)$.

We define $\{(\tilde{X}_{i,t+s}, \tilde{U}_{i,t+s})\}_{s=1}^{T-t}$ which is generated by the iid sequence of random functions $(\tilde{\alpha}_{i,t+s})_{s=0}^{T-t-1}$ distributed as in the original data set generation but independent of the original data set and starting from the pair $(X_{i,t}, U_{i,t})$, i.e. $(\tilde{X}_{i,t+s}, \tilde{U}_{i,t+s}) = \tilde{\alpha}_{i,t+s-1}\tilde{\alpha}_{i,t+s-2}\cdots\tilde{\alpha}_{i,t}(X_{i,t}, U_{i,t})$. Define $D_{i,t} = (D \setminus \{(X_{i,t+s}, U_{i,t+s})\}_{s=1}^{T-t} \cup \{(\tilde{X}_{i,t+s}, \tilde{U}_{i,t+s})\}_{s=1}^{T-t}$ as the data set where $\{(X_{i,t+s}, U_{i,t+s})\}_{s=1}^{T-t}$ is replaced by $\{(\tilde{X}_{i,t+s}, \tilde{U}_{i,t+s})\}_{s=1}^{T-t}$. Analogously, for any multi-index i^l, t^l, we recursively define $D_{i^{k:l},t^{k:l}} = (D_{i^{k+1:l},t^{k+1:l}} \setminus \{(X_{i_k,t_k+s}, U_{i_k,t_k+s})\}_{s=1}^{T-t_k}) \cup \{(\tilde{X}_{i_k,t_k+s}, \tilde{U}_{i_k,t_k+s})\}_{s=1}^{T-t_k}$. For a weight, we highlight the data set it depends on by writing $W_{n,i^\tau,t^\tau}(\tilde{D})$ where $\tilde{D}$ is a data set. With this notation we can now state the conditions on the weights of the local averaging regression under which the estimation is consistent:

Assumption 2. *For all $\tau = 0, ..., T$ and with $W_{n,i^\tau,t^\tau}(\tilde{X}_1, u)$ denoted by W_{n,i^τ,t^τ}, the following holds:*

1. *For either $\tilde{D}_1 = \tilde{D}_2 = D$ or $\tilde{D}_1 = D_{i^\tau,t^\tau}$, $\tilde{D}_2 = D_{j^\tau,s^\tau}$,*

$$\lim_{n\to\infty} \mathbb{E}\left[\sum_{i^\tau,t^\tau}^{n,T} \sum_{\substack{j^\tau,s^\tau \\ |[i^\tau,j^\tau]|<2\tau+2}}^{n,T} |W_{n,i^\tau,t^\tau}(\tilde{D}_1) W_{n,j^\tau,s^\tau}(\tilde{D}_2)|\right] = 0$$

2. *There is a $L \geq 1$ such that*

$$\mathbb{P}\left(\sum_{i^\tau,t^\tau}^{n,T} |W_{n,i^\tau,t^\tau}| \leq L\right) = 1$$

3. *For all $a > 0$,*

$$\lim_{n\to\infty} \mathbb{E}\left[\sum_{i^\tau,t^\tau}^{n,T} |W_{n,i^\tau,t^\tau}| \left(I_{d\left((X_{i_0,t_0},U_{i_0,t_0}),(\tilde{X}_1,u_0(\tilde{X}_1))\right)>a} + \sum_{k=0}^{\tau} I_{\tau-k}(a)\right)\right] = 0$$

where

$$I_m(a) = I_{d\left((X_{i_m,t_m},U_{i_m,t_m}),(X_{i_{m-1},t_{m-1}+1},u_m(X_{i_{m-1},t_{m-1}+1}))\right)>a}$$

and $X_{i_{-1},t_{-1}} = \tilde{X}_1$ for all i^τ, t^τ.

4. It holds in probability that

$$\sum_{i^\tau,t^\tau}^{n,T} W_{n,i^\tau,t^\tau} \to 1$$

5.

$$\lim_{n\to\infty} \mathbb{E}\left[\left(\sum_{i^\tau,t^\tau}^{n,T} |W_{n,i^\tau,t^\tau}(D_{i^\tau,t^\tau}) - W_{n,i^\tau,t^\tau}(D)|\right)^2\right] = 0$$

and

$$\lim_{n\to\infty} \mathbb{E}\Bigg[\sum_{i^\tau,t^\tau}^{n,T} \sum_{\substack{j^\tau,s^\tau \\ |[i^\tau,j^\tau]|=2\tau+2}}^{n,T} |W_{n,i^\tau,t^\tau}(D_{i^\tau,t^\tau})| \cdot |W_{n,j^\tau,s^\tau}(D_{j^\tau,s^\tau}) - W_{n,j^\tau,s^\tau}(D_{(i^\tau,j^\tau),(t^\tau,s^\tau)})|\Bigg] = 0$$

Theorem 2. *Given the setting described in Section 3.2 and under Assumption 2, $\hat{m}_{l,n}(u)$ is consistent at each family of feedback policies $u = (u_t)_{t=0}^T$ with $u_t : \mathcal{X} \to \mathcal{U}$, i.e.,*

$$\lim_{n\to\infty} \mathbb{E}\left[\left(\hat{m}_{l_n,n}(u) - m(u)\right)^2\right] = 0 \tag{3.4}$$

where $l_n \to \infty$.

The proof of Theorem 2 is given in Section 3.5.1.

Remark 8. *Theorem 2 gives consistency of the local averaging regression estimator under certain conditions on the weights. This theorem can be viewed as a generalization of Stone's Theorem (see [87], Theorem 4.1). In particular, Theorem 2 generalizes Stone's Theorem in three important aspects: First, Theorem 2 allows an episodic sampling setting where within each episode the data is not independent and not identically distributed. Second, the counterfactual estimation problem arising through off-policy evaluation is taken into account and, third, we allow for a sequential dependence structure in the weights and through this in the regression. As the conditions on the weights that we propose are still general, we believe that this result can be of independent interest and potentially be applied to a wide range of local averaging regression estimators such as partitioning or kernel-based methods.*

3.3.2 The K_n-NN path regression estimator

The second step of the proof of Theorem 1 is to show that the conditions on the weights in Theorem 2 are satisfied by a regression estimator based on K-NN paths. Define the set $\Lambda_n(x,u) = \{K_n\text{-NN paths starting at } x \text{ under } u\}$ and introduce the K_n-NN path regression by $m_n^{\mathrm{NN}}(x,u,\tau) = \sum_{i^\tau,t^\tau}^{n,T} \frac{1}{(K_n)^\tau} I_{(i^\tau,t^\tau)\in\Lambda_n(x,u)} \cdot R_{i_\tau,t_\tau}$ or equivalently by $m_n^{\mathrm{NN}}(x,u,\tau) = \frac{1}{(K_n)^\tau} \sum_{k^\tau}^{K_n} R_\tau^{k^\tau}(x;u)$. For $\mathbb{E}[R_\tau|_u]$, we use $m_{l,n}^{\mathrm{NN}}(u,\tau) = \frac{1}{l}\sum_{j=1}^{l} m_n^{\mathrm{NN}}(\tilde{X}_j,u,\tau)$ where $\tilde{X}_j$ is sampled independently from the initial state distribution and, for $m(u)$ the estimator is $m_{l,n}^{\mathrm{NN}}(u) = \sum_{\tau=0}^{T-1} m_{l,n}^{\mathrm{NN}}(u,\tau)$. We identify the weights in the K_n-NN path regression by $V_{n,i^\tau,t^\tau} \coloneqq (1/(K_n)^{\tau+1}) I_{(i^\tau,t^\tau) \text{ is a } K_n\text{-NN path}}(x,u)$.

Theorem 3. *Given the setting described in Section 3.2 and under Assumption 1, it holds at each family of feedback policies $u = (u_t)_{t=0}^T$ with $u_t : \mathcal{X} \to \mathcal{U}$, that*

$$\lim_{n\to\infty} \mathbb{E}\left[\left(m_{l_n,n}^{NN}(u) - m(u)\right)^2\right] = 0 \tag{3.5}$$

with $l_n \to \infty$, $K_n \to \infty$ and $\frac{K_n}{n} \to 0$.

This theorem is proved by showing that the weights V_{n,i^τ,t^τ} satisfy the conditions of Theorem 2. This is done by first analysing the asymptotic behaviour of $\mathbb{E}[V_{n,i^\tau,t^\tau}]$. This analysis is key to showing that a nearest neighbour-based OPE algorithm can directly use episodic sampling. With this result, we investigate the limit of $\sum_{i^\tau,t^\tau}^{n,T} V_{n,i^\tau,t^\tau}$. Finally, it remains to show that with Assumption 1 at any relevant point in $\mathcal{S}$ the distance to the K_n-nearest neighbor vanishes. The proof of Theorem 3 is given in Section 3.5.2.

As the third and final step, one identifies $m_{l,n}^{\text{KNNR}}(u)$ as a subsampled version of the estimator $m_{l,n}^{\text{NN}}(u)$ in the sense that $m_{l,n}^{\text{KNNR}}(u)$ averages over a growing number of rewards chosen along randomly selected K_n-paths that are also contained in $m_{l,n}^{\text{NN}}(u)$. Then showing that the mean squared error between $m_{l,n}^{\text{KNNR}}(u)$ and $m_{l,n}^{\text{NN}}(u)$ vanishes yields the statement. This proof is given in Section 3.5.3.

3.4 Experiments

We demonstrate the effectiveness of the KNNR algorithm for off-policy evaluation on three stochastic control environments with varying control mechanisms, areas of applications, sources of uncertainty and dimensionality. Namely, we study a linear quadratic regulator environment, trade execution in limit order books and online stochastic bin packing. As baselines, we consider weighted per episode importance sampling (PEIS) [173], weighted per decision importance sampling (PDIS) [173], weighted doubly-robust (WDR) [207], model-free Monte Carlo (MFMC) [62, 63], fitted Q-evaluation with a linear-quadratic regression (FQE Lin) [131], fitted Q-evaluation with a nearest neighbor regression (FQE NN) [131] and the naive average of rewards generated by the behaviour policy (NA). Before presenting the results, we shortly highlight the differentiating properties of the test environments. Note that more details on the test environments, the target and behaviour policies, the baselines, the choice of hyper-parameters, and results are included in the supplementary materials.

Linear Quadratic Regulator The linear quadratic regulator problem (LQR) is often seen as the simplest non-trivial example for stochastic control and was proposed as an important test-bed in reinforcement learning by [179]. The LQR problem has a continuous state and action space, with the dynamics linear in the state and action, and the reward quadratic in the state and action. It can be viewed as a local approximation of more general non-linear stochastic control problems. For this experiment, we consider the double integrator example from [179] with a higher variance to highlight the benefits of the resampling approach in stochastic settings. The LQR environment has a two dimensional state space and a one dimensional action space.

Optimal Execution in Limit Order Book Markets The second environment models optimal execution in limit order books (LOB), as in [39], Chapter 8.2. In this scenario, a trading agent has the task to liquidate a position of assets in a fixed amount of time, trying to maximise trade revenues. They do so by quoting a price difference to a mid-price that varies exogenously. A larger difference to the mid-price is associated with a lower probability of a sale but a higher revenue, and vice versa. The state consists of the current price of the asset (continuous) and inventory (finite). The action space is one dimensional and continuous. The target policy is non-stationary, depending on time and the current level of inventory. Similar control mechanisms are central in the dynamic pricing literature (e.g., [65]).

Stochastic Online Bin Packing We consider a slightly altered version of the stochastic online bin packing (BP) environment as introduced in [13, 86]. The general idea of the bin packing problem is that, over time, items of random size arrive and need to be allocated to bins of fixed size, with the goal to minimize open capacities at each point in time. Bin packing is used to model many different resource allocation problems in both operations research and computer science. By our modification, we allow the size of the next arriving item to be negatively correlated with current free capacity. In operations research, this dependence could originate from having a separate dynamic pricing policy depending on available capacities [16, 65]. This problem differs from the previous problems in that both the state and action space are finite. The state space consists of the number of bins at each

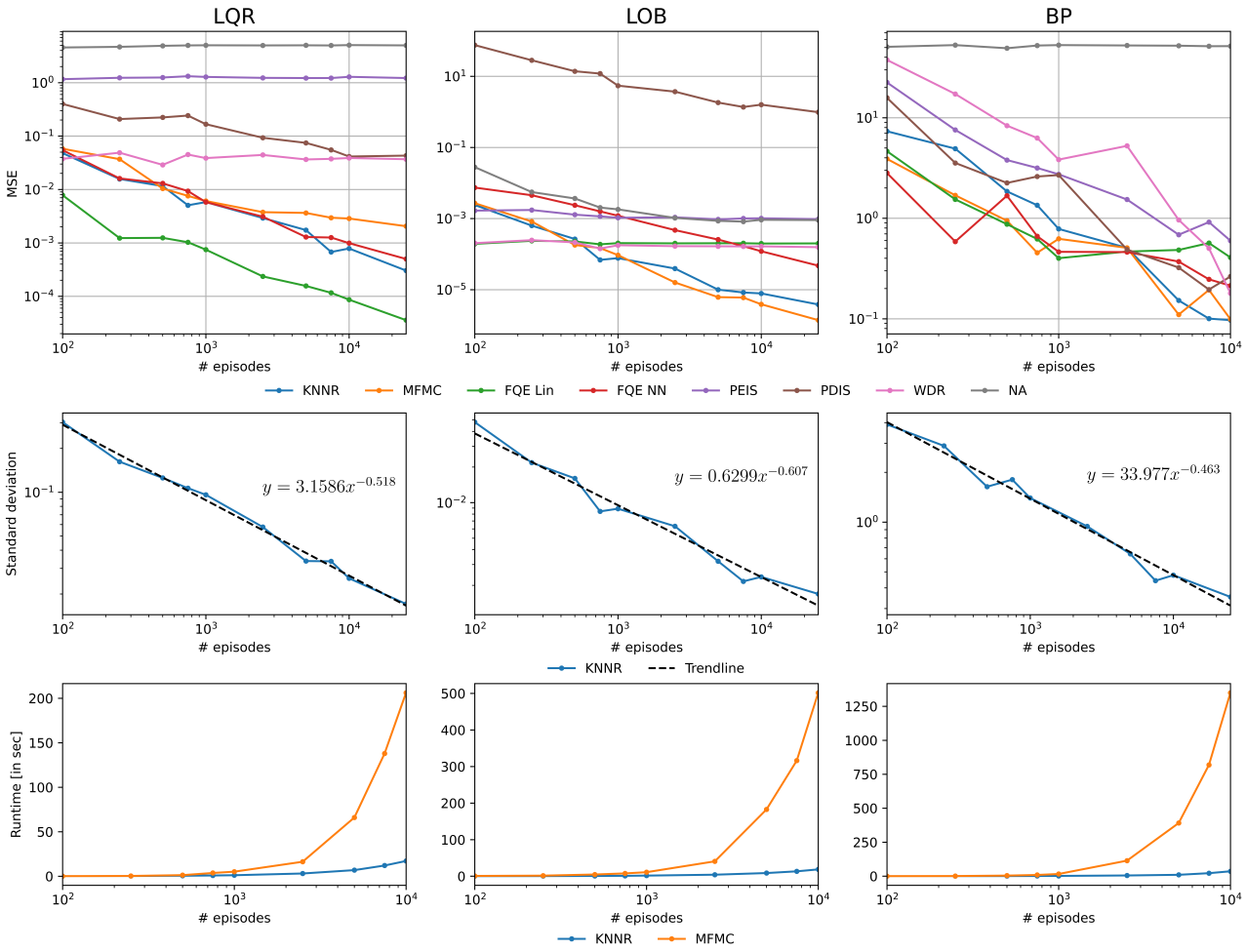


Figure 3.2: Experimental results: Each column corresponds to one test environment. First row: Median MSEs of the value estimates (where the target for the value is computed via Monte Carlo roll outs using the target policy) versus the number of episodes sampled under the behaviour policy (log-log plot). Second row: Standard deviation in the resampling estimates of the value as a function of the number of episodes in the data set. Third row: Comparison of median runtimes for the resampling algorithm and MFMC depending on the number of episodes in the data set.

potential fill level and the size of the arriving item. The action space contains the possible assignments of the arriving item. We choose this problem as a benchmark as it has a higher dimensional state space ($d = 10$) and highlights the wider applicability of the resampling algorithm to non-standard stochastic control environments.

Results We report our results in Figure 3.2. From the first row, one can see that in all environments our resampling algorithm exhibits decreasing Mean-Square Errors (MSEs) for increasing data set sizes, measured in the number of episodes. The performance of the resampling algorithm versus the baselines is competitive. It consistently outperforms PEIS, PDIS and naive averages, and is mostly better or performs similarly to the other baselines. The exception is FQE Lin in the LQR environment. This can be explained by the fact that the value function in the LQR problem has exactly the same functional form as the linear quadratic regression used in FQE Lin for value function estimation. The second row highlights that the standard deviation of the resampling estimates vanishes with approximately the rate $O(n^{-0.5})$ as for Monte Carlo methods. Finally, the third row shows that the actual runtime for the resampling algorithm is lower compared to MFMC and reaches an order of magnitude difference for medium sized data sets.

3.5 Proofs

In this section, we give the proofs for the main results. Other additional supporting results are included in the appendix.

3.5.1 Proof of Theorem 2

Proof of Theorem 2. Note that

$$\mathbb{E}\left[\left(\sum_{\tau=0}^{T}\hat{m}_{l_n,n}(u,\tau)-\mathbb{E}[(R_\tau|_u)]\right)^2\right]\leq(T+1)\sum_{\tau=0}^{T}\mathbb{E}\left[\left(\hat{m}_{l_n,n}(u,\tau)-\mathbb{E}[(R_\tau|_u)]\right)^2\right].$$

It will be sufficient to show that

$$\lim_{n\to\infty}\mathbb{E}\left[\left(\hat{m}_{l_n,n}(u,\tau)-\mathbb{E}[(R_\tau|_u)]\right)^2\right]=0$$

for a fixed τ since the arguments are analogous for all τ.

For that purpose, consider

$$\begin{aligned}
&\mathbb{E}\left[\left(\hat{m}_{l_n,n}(u,\tau)-\mathbb{E}[(R_\tau|_u)]\right)^2\right]\\
&\quad\leq 2\mathbb{E}\left[\left(\frac{1}{l_n}\left(\sum_{j=1}^{l_n}\left(\sum_{i^\tau,t^\tau}^{n,T}W_{n,i^\tau,t^\tau}(\tilde{X}_j,u)r_{i_\tau,t_\tau}\right)-\mathbb{E}[R_\tau|_u|\tilde{X}_j]\right)\right)^2\right]\\
&\quad\quad+2\mathbb{E}\left[\left(\frac{1}{l_n}\sum_{j=1}^{l_n}\mathbb{E}[R_\tau|_u|\tilde{X}_j]-m(u)\right)^2\right]\\
&=:2C_n+2D_n.
\end{aligned}$$

By Jensen's inequality, the fact that the sampling of the initial state is iid and independent of the data set, we get

$$C_n\leq\mathbb{E}\left[\left(\sum_{i^\tau,t^\tau}^{n,T}W_{n,i^\tau,t^\tau}(\tilde{X}_1,u)R_{i_\tau,t_\tau}-\mathbb{E}[R_\tau|_u|\tilde{X}_1]\right)^2\right]=:C_n'.$$

Further[7],

$$\begin{aligned}
C_n' \leq & 3\mathbb{E}\left[\left(\sum_{i^\tau,t^\tau}^{n,T} W_{n,i^\tau,t^\tau}(R_{i_\tau,t_\tau} - r(X_{i_\tau,t_\tau}, U_{i_\tau,t_\tau}))\right)^2\right] \\
& + 3\mathbb{E}\left[\left(\sum_{i^\tau,t^\tau}^{n,T} W_{n,i^\tau,t^\tau}\Big(r(X_{i_\tau,t_\tau}, U_{i_\tau,t_\tau}) \right.\right. \\
& \qquad \left.\left. - r(\alpha_{i_{\tau-1},t_{\tau-1}}|_u \circ \cdots \circ \alpha_{i_0,t_0}|_u(\tilde{X}_1, u_0(\tilde{X}_1)))\Big)\right)^2\right] \\
& + 3\mathbb{E}\left[\left(\sum_{i^\tau,t^\tau}^{n,T} W_{n,i^\tau,t^\tau} r(\alpha_{i_{\tau-1},t_{\tau-1}}|_u \circ \cdots \circ \alpha_{i_0,t_0}|_u(\tilde{X}_1, u_0(\tilde{X}_1))) \right.\right. \\
& \qquad \left.\left. - \mathbb{E}[R_\tau|_u|\tilde{X}_1]\right)^2\right] \\
=: & 3E_n + 3F_n + 3G_n
\end{aligned}$$

In Proposition 7 in the supplementary materials we show that D_n, E_n, F_n, and G_n vanish using Assumption 2. □

3.5.2 Proof of Theorem 3

Due to the recursive definition of K_n-NN paths, V_{n,i^τ,t^τ} can be written as a product of weights:

$$V_{n,i^\tau,t^\tau}(\tilde{D}) = V_{n,i_0,t_0}(\tilde{D}) \prod_{k=1}^{\tau} V_{n,[(i_{k-1},t_{k-1}),(i_k,t_k)]}(\tilde{D})$$

where

$$V_{n,i_0,t_0}(\tilde{X}_1, u, \tilde{D}) = \frac{1}{K_n} I_{(X_{i_0,t_0}, U_{i_0,t_0}) \text{ is a } K_n\text{−NN in } \tilde{D}}(\tilde{X}_1, u_0(\tilde{X}_1))$$

and

$$\begin{aligned}
& V_{n,[(i_{k-1},t_{k-1}),(i_k,t_k)]}(\tilde{D}) = \\
& \quad \frac{1}{K_n} I_{(X_{i_k,t_k}, U_{i_k,t_k}) \text{ is a } K_n\text{−NN in } \tilde{D}}(X_{i_{k-1},t_{k-1}+1}, u_{t+1}(X_{i_{k-1},t_{k-1}+1})).
\end{aligned}$$

We omit the dependence on the data set $\tilde{D}$, if the unaltered data D is used. Finally, note that by definition it holds almost surely for $k = 1, \ldots, \tau$ that

$$\sum_{i=1}^{n}\sum_{t=0}^{T-1} V_{n,[(i_{k-1},t_{k-1}),(i,t)]} = 1 \quad \text{and} \quad \sum_{i=1}^{n}\sum_{t=0}^{T-1} V_{n,i,t}(\tilde{X}_1, u) = 1.$$

The next two propositions analyse the asymptotic behaviour of $\mathbb{E}[V_{n,i^\tau,t^\tau}]$ and $\sum_{i^\tau,t^\tau}^{n,T} V_{n,i^\tau,t^\tau}$, respectively.

Proposition 1. *Let Assumption 1.1 be satisfied. For all $\tau \leq T$ and (i^τ, t^τ), it holds for all n that*

$$\mathbb{E}[V_{n,i^\tau,t^\tau}] \leq (n - \tau - 1)^{-|i^\tau|} K_n^{-(\tau+1-|i^\tau|)}. \tag{3.6}$$

[7] For readability, we omit $(\tilde{X}_1, u)$ from the weights.

Proof. We prove this statement by induction over τ. First, consider $\tau = 0$:

$$\begin{aligned}\mathbb{E}[V_{n,i_0,t_0}(\tilde{X}_1, u)] =& \frac{1}{K_n}\mathbb{P}((X_{i_0,t_0}, U_{i_0,t_0}) \text{ is a } K_n - \text{NN of } (\tilde{X}_1, u_0(\tilde{X}_1)) \text{ in } D)\\ \leq& \frac{1}{K_n}\mathbb{P}((X_{i_0,t_0}, U_{i_0,t_0}) \text{ is a } K_n - \text{NN of } (\tilde{X}_1, u_0(\tilde{X}_1)) \text{ in } D_{t_0})\\ \leq& \frac{K_n}{nK_n} \leq \frac{1}{n-1}\end{aligned}$$

where $D_{t_0} \coloneqq \{(X_{i,t_0}, U_{i,t_0})\}_{i=1}^n$ and we use that the samples in D_{t_0} are iid.

For $\tau \Rightarrow \tau + 1$ where $\tau + 1 \leq T$, we consider two cases. In the first case we assume that $|i^\tau| = |i^{\tau+1}|$:

$$\begin{aligned}&\mathbb{E}[V_{n,i^{\tau+1},t^{\tau+1}}] = \mathbb{E}[V_{n,[(i_\tau,t_\tau),(i_{\tau+1},t_{\tau+1})]}V_{n,i^\tau,t^\tau}]\\ =&\mathbb{E}[V_{n,[(i_\tau,t_\tau),(i_{\tau+1},t_{\tau+1})]} \mid V_{n,i^\tau,t^\tau} = K_n^{-(\tau+1)}]\mathbb{E}[V_{n,i^\tau,t^\tau}]\\ \leq&(n-1)^{-|i^\tau|}K_n^{-(\tau+2-|i^\tau|)}\\ &\cdot \mathbb{P}\Big((X_{i_k,t_k}, U_{i_k,t_k}) \text{ is a } K_n - \text{NN of } (X_{i_{k-1},t_{k-1}+1}, u_{t+1}(X_{i_{k-1},t_{k-1}+1})) \text{ in } D, (i^\tau, t^\tau)\\ &\qquad \mid V_{n,i^\tau,t^\tau} = K_n^{-(\tau+1)}\Big)\\ \leq&(n-1)^{-|i^{\tau+1}|}K_n^{-(\tau+2-|i^{\tau+1}|)}\end{aligned}$$

For the second case consider $|i^\tau| + 1 = |i^{\tau+1}|$. Assume that

$$\mathbb{E}[V_{n,i^{\tau+1},t^{\tau+1}}] > (n-\tau-2)^{-|i^{\tau+1}|}K_n^{-(\tau+2-|i^{\tau+1}|)}.$$

By the iid assumption for the sampling of the different episodes, it has to hold that

$$\mathbb{E}[V_{n,[(i_\tau,t_\tau),(j,t_{\tau+1})]}V_{n,i^\tau,t^\tau}] > (n-\tau-2)^{-|i^{\tau+1}|}K_n^{-(\tau+2-|i^{\tau+1}|)} \tag{3.7}$$

for all j where $|(i^\tau, j)| > |i^\tau|$. By induction hypothesis it follows that

$$\mathbb{E}[V_{n,[(i_\tau,t_\tau),(j,t_{\tau+1})]} \mid V_{n,i^\tau,t^\tau} = 1/K_n^{\tau+1}] > (n-\tau-2)^{-1}$$

Recall that $\sum_{i=1}^n \sum_{t=0}^{T-1} V_{n,[(i_\tau,t_\tau),(i,t)]} = 1$ almost surely and, thus,

$$\sum_{i=1}^n \sum_{t=0}^{T-1} \mathbb{E}[\mathbb{E}[V_{n,[(i_\tau,t_\tau),(i,t)]} \mid V_{n,i^\tau,t^\tau} = 1/K_n^{\tau+1}]] = 1$$

We use the notation $i \in i^\tau$ as shorthand for $i \in i_k{}_{k=0}^{\tau}$ and $i \notin i^\tau$ for $i \notin i_k{}_{k=0}^{\tau}$. Splitting the sum and using

condition (3.7), yields

$$\begin{aligned}
1 =& \sum_{\substack{i=1,\\ i\in i^\tau}}^{n}\sum_{t=0}^{T-1}\mathbb{E}[V_{n,[(i_\tau,t_\tau),(i,t)]} \mid V_{n,i^\tau,t^\tau} = 1/K_n^{\tau+1}] \\
&+ \sum_{\substack{i=1,\\ i\notin i^\tau}}^{n}\sum_{\substack{t=0,\\ t\neq t_{\tau+1}}}^{T-1}\mathbb{E}[V_{n,[(i_\tau,t_\tau),(i,t)]} \mid V_{n,i^\tau,t^\tau} = 1/K_n^{\tau+1}] \\
&+ \sum_{\substack{i=1,\\ i\notin i^\tau}}^{n}\mathbb{E}[V_{n,[(i_\tau,t_\tau),(i,t_{\tau+1})]} \mid V_{n,i^\tau,t^\tau} = 1/K_n^{\tau+1}] \\
\geq& \sum_{\substack{i=1,\\ i\notin i^\tau}}^{n}\mathbb{E}[V_{n,[(i_\tau,t_\tau),(i,t_{\tau+1})]} \mid V_{n,i^\tau,t^\tau} = 1/K_n^{\tau+1}] \\
>& \frac{n-\tau-2}{n-\tau-2} = 1
\end{aligned}$$

This is a contradiction. Hence, the induction step is satisfied and the proposition holds. □

Proposition 2. *Let Assumption 1.1 be satisfied. Then it holds that:*

1.

$$\lim_{n\to\infty}\mathbb{E}\left[\sum_{i^\tau,t^\tau}^{n,T}\sum_{\substack{j^\tau,s^\tau\\ |[i^\tau,j^\tau]|<2\tau+2}}^{n,T} V_{n,i^\tau,t^\tau}(\tilde{D}_1)V_{n,j^\tau,s^\tau}(\tilde{D}_2)\right] = 0 \tag{3.8}$$

for $\tilde{D}_1 = \tilde{D}_2 = D$ *and* $\tilde{D}_1 = D_{i^\tau,t^\tau}$, $\tilde{D}_2 = D_{j^\tau,s^\tau}$.

2.

$$\lim_{n\to\infty}\mathbb{E}\left[\left(\sum_{i^\tau,t^\tau}^{n,T}|V_{n,i^\tau,t^\tau}(D_{i^\tau,t^\tau}) - V_{n,i^\tau,t^\tau}(D)|\right)^2\right] = 0. \tag{3.9}$$

3.

$$\begin{aligned}
\lim_{n\to\infty}\mathbb{E}\Bigg[&\sum_{i^\tau,t^\tau}^{n,T}\sum_{\substack{j^\tau,s^\tau\\ |[i^\tau,j^\tau]|=2\tau+2}}^{n,T} V_{n,i^\tau,t^\tau}(D_{i^\tau,t^\tau}) \\
&\cdot |V_{n,j^\tau,s^\tau}(D_{j^\tau,s^\tau}) - V_{n,j^\tau,s^\tau}(D_{(i^\tau,j^\tau),(t^\tau,s^\tau)})|\Bigg] = 0
\end{aligned} \tag{3.10}$$

Proof. The proof relies on different variations of Proposition 1. The details can be found in the appendix. □

The following Lemma guarantees that under Assumption 1 at any relevant point in $\mathcal{S}$ the distance to the K_n-nearest neighbor vanishes.

Lemma 1. *Under Assumption 1 , let* $\lim_{n\to\infty} K_n/n = 0$ *and* $x_0 \in supp(\pi_t|_x)$ *for some* $t = 0,\dots,T-1$, *then*

$$d\left((x_0,u_0(x_0)),(X_{(K_n,n)},U_{(K_n,n)})(x_0,u_0(x_0))\right) \to 0 \tag{3.11}$$

and, for all $\tau = 1, \dots, T-1$ *and any fixed* K_n*-nearest neighbor path* $k^{\tau-1} = [k_0, \dots, k_{\tau-1}]$,

$$d\Big((X^{k^{\tau-1}}_{i_{\tau-1},t_{\tau-1}+1}, u_\tau(X^{k^{\tau-1}}_{i_{\tau-1},t_{\tau-1}+1})) \\ , (X^{k^{\tau-1}}_{(K_n,n)}, U^{k^{\tau-1}}_{(K_n,n)})((X^{k^{\tau-1}}_{i_{\tau-1},t_{\tau-1}+1}, u_\tau(X^{k^{\tau-1}}_{i_{\tau-1},t_{\tau-1}+1})))\Big) \to 0 \tag{3.12}$$

almost surely.

Proof. Let $\varepsilon > 0$ and fix arbitrarily $k^{\tau-1}$, τ and x_0. $S(x, \varepsilon)$ denote the ε-ball centred at $x \in S$. For proving (3.12), observe that

$$\Big\{ d\left((X^{k^{\tau-1}}_{i_{\tau-1},t_{\tau-1}+1}, u_\tau(X^{k^{\tau-1}}_{i_{\tau-1},t_{\tau-1}+1})), (X^{k^{\tau-1}}_{(K_n,n)}, U^{k^{\tau-1}}_{(K_n,n)})\right) > \varepsilon \Big\} \\ = \left\{ \frac{1}{nT} \sum_{i=1}^{n} \sum_{t=0}^{T-1} I_{(X_{i,t},U_{i,t}) \in S\left((X^{k^{\tau-1}}_{i_{\tau-1},t_{\tau-1}+1}, u_\tau(X^{k^{\tau-1}}_{i_{\tau-1},t_{\tau-1}+1})), \varepsilon\right)} < \frac{k_n}{nT} \right\} =: M_\tau$$

where $k^{\tau-1}$ and τ are fixed and

$$M_\tau \subseteq \bigcup_{t=0}^{T-1} \left\{ \frac{1}{nT} \sum_{i=1}^{n} I_{(X_{i,t},U_{i,t}) \in S\left((X^{k^{\tau-1}}_{i_{\tau-1},t_{\tau-1}+1}, u_\tau(X^{k^{\tau-1}}_{i_{\tau-1},t_{\tau-1}+1})), \varepsilon\right)} < \frac{k_n}{nT} \right\}.$$

For any $\tilde{x} \in \bigcup_{t=0}^{T-1} supp(\pi_t|_x)$,

$$\pi(M_0 | X^{k^{\tau-1}}_{i_{\tau-1},t_{\tau-1}+1} = \tilde{x}) \\ \leq \sum_{t=0}^{T-1} \pi_t \Bigg(\left\{ \frac{1}{nT} \sum_{i=1}^{n} I_{(X_{i,t},U_{i,t}) \in S\left((X^{k^{\tau-1}}_{i_{\tau-1},t_{\tau-1}+1}, u_\tau(X^{k^{\tau-1}}_{i_{\tau-1},t_{\tau-1}+1})), \varepsilon\right)} < \frac{k_n}{nT} \right\} \\ \Bigg| X^{k^{\tau-1}}_{i_{\tau-1},t_{\tau-1}+1} = \tilde{x} \Bigg).$$

where $\pi(\cdot | X^{k^{\tau-1}}_{i_{\tau-1},t_{\tau-1}+1} = \tilde{x})$ and $\pi_t(\cdot | X^{k^{\tau-1}}_{i_{\tau-1},t_{\tau-1}+1} = \tilde{x})$ for all t denotes the conditional probability measure. Note that by the strong law of large number

$$\sum_{t=0}^{T-1} \frac{1}{n} \sum_{i=1}^{n} I_{(X_{i,t},U_{i,t}) \in S((\tilde{x}, u_\tau(\tilde{x})), \varepsilon)} \\ \to \sum_{t=0}^{T-1} \pi_t \left(S\left((X^{k^{\tau-1}}_{i_{\tau-1},t_{\tau-1}+1}, u_\tau(X^{k^{\tau-1}}_{i_{\tau-1},t_{\tau-1}+1})), \varepsilon \right) \Big| X^{k^{\tau-1}}_{i_{\tau-1},t_{\tau-1}+1} = \tilde{x} \right)$$

almost surely and that by Assumption 1,

$$\sum_{t=0}^{T-1} \pi_t \left(S\left((X^{k^{\tau-1}}_{i_{\tau-1},t_{\tau-1}+1}, u_\tau(X^{k^{\tau-1}}_{i_{\tau-1},t_{\tau-1}+1})), \varepsilon \right) \Big| X^{k^{\tau-1}}_{i_{\tau-1},t_{\tau-1}+1} = \tilde{x} \right) > 0.$$

Since by assumption, $K_n/n \to 0$, it holds that

$$d\left((\tilde{x}, u_\tau(\tilde{x})), ((X^{k^{\tau-1}}_{(K_n,n)}, U^{k^{\tau-1}}_{(K_n,n)}))(\tilde{x}, u_\tau(\tilde{x})) \right) \to 0$$

almost surely for all $\tilde{x} \in \bigcup_{t=0}^{T-1} supp(\pi_t|_x)$. Let $\mu_{X^{k^{\tau-1}}_{i_{\tau-1},t_{\tau-1}+1}}$ be the measure describing $X^{k^{\tau-1}}_{i_{\tau-1},t_{\tau-1}+1}$ then it holds that $supp(\mu_{X^{k^{\tau-1}}_{i_{\tau-1},t_{\tau-1}+1}}) \subseteq \bigcup_{t=0}^{T-1} supp(\pi_t|_x)$. Thus, (3.12) holds. (3.11) is shown analogously. □

Proof of Theorem 3. For proving this theorem we check the assumptions of Theorem 2. Assumption 2.2 and Assumption 2.4 hold automatically by the definition of the weights. Proposition 2.1 implies Assumption 2.1. Proposition 2.2 and Proposition 2.3 imply Assumption 2.5. Let $\varepsilon > 0$, then for all $\tau = 0, \dots, T-1$ it holds that

$$\begin{aligned}
&\mathbb{E}\left[\sum_{i^\tau,t^\tau}^{n,T} V_{n,i^\tau,t^\tau} I_{d\left((X_{i_0,t_0},U_{i_0,t_0}),(\tilde{X}_1,u_0(\tilde{X}_1))\right)>\varepsilon}\right] \\
&= \int_{\mathcal{X}} \mathbb{E}\left[\sum_{i^\tau,t^\tau}^{n,T} V_{n,i^\tau,t^\tau} I_{d((X_{i_0,t_0},U_{i_0,t_0}),(x,u_0(x)))>\varepsilon}\right]\pi_0(dx) \\
&= \int_{\mathcal{X}} \mathbb{E}\left[\frac{1}{K_n^\tau}\sum_{i^\tau,t^\tau}^{n,T} I_{i^\tau,t^\tau \text{ is a } K_n\text{-NN path}} I_{d((X_{i_0,t_0},U_{i_0,t_0}),(x,u_0(x)))>\varepsilon}\right]\pi_0(dx) \\
&= \int_{\mathcal{X}} \mathbb{E}\left[\frac{1}{K_n^\tau}\sum_{k^\tau}^{K_n} I_{d((X_0^{k^\tau},U_0^{k^\tau})(x,u_0(x)),(x,u_0(x)))>\varepsilon}\right]\pi_0(dx) \to 0,
\end{aligned}$$

if

$$\int_{\mathcal{X}} \mathbb{P}\left(d\left((x,u_0(x)),(X_{(K_n,n)},U_{(K_n,n)})(x,u_0(x))\right)\right)\pi_0(dx).$$

This holds since Lemma 1 implies

$$\mathbb{P}\left(d\left((x,u_0(x)),(X_{(K_n,n)},U_{(K_n,n)})(x,u_0(x))\right)\right) \to 0$$

for all $x \in supp(\pi_0)$. Note for some $X_t^{k^\tau}$ corresponding to some $X_{j,s}$ in the original indexing, we denote $X_{j,s+1}$ by $X_{t,+}^{k^\tau}$ and by $U_{t,+}^{k^\tau}$ we denote $u_{t+1}(X_{j,s+1})$. For any $l = 0, \dots, T-1$, we have that

$$\begin{aligned}
&\mathbb{E}\left[\sum_{i^\tau,t^\tau}^{n,T} V_{n,i^\tau,t^\tau}(\tilde{X}_1) I_{\tau-l}(\varepsilon)\right] = \int_{\mathcal{X}} \mathbb{E}\left[\sum_{i^\tau,t^\tau}^{n,T} V_{n,i^\tau,t^\tau}(x) I_{\tau-l}(\varepsilon)\right]\pi_0(dx) \\
&= \int_{\mathcal{X}} \mathbb{E}\left[\frac{1}{K_n^\tau}\sum_{i^\tau,t^\tau}^{n,T} I_{i^\tau,t^\tau \text{ is a } K_n\text{-NN path}}(x) I_{\tau-l}(\varepsilon)\right]\pi_0(dx) \\
&= \int_{\mathcal{X}} \mathbb{E}\left[\frac{1}{K_n^\tau}\sum_{k^\tau}^{K_n} I_{d\left((X_{\tau-l}^{k^\tau},U_{\tau-l}^{k^\tau})(x),(X_{\tau-l-1,+}^{k^\tau},U_{\tau-l-1,+}^{k^\tau})(x)\right)>\varepsilon}\right]\pi_0(dx) \to 0
\end{aligned}$$

if for any sequence k^τ

$$\int_{\mathcal{X}} \mathbb{P}\left(d\left((X_{\tau-l}^{(k^{\tau-l-1},K_n,k^{\tau-l+1:\tau})},U_{\tau-l}^{k^\tau})(x),(X_{\tau-l-1,+}^{k^\tau},U_{\tau-l-1,+}^{k^\tau})(x)\right)\right)\pi_0(dx) \to 0$$

where $(k^{\tau-l-1}, K_n, k^{\tau-l+1:\tau}) = (k_0^\tau, \dots, k_{\tau-l-1}^\tau, K_n, k_{\tau-l+1}^\tau, \dots, k_\tau^\tau)$. This since Lemma 1, implies

$$\mathbb{P}\left(d\left((X_{\tau-l}^{(k^{\tau-l-1},K_n,k^{\tau-l+1:\tau})},U_{\tau-l}^{k^\tau})(x),(X_{\tau-l-1,+}^{k^\tau},U_{\tau-l-1,+}^{k^\tau})(x)\right)\right) \to 0$$

for all $x \in supp(\pi_0)$. Thus, Assumption 2.3 is satisfied. □

3.5.3 Consistency of the KNNR estimator

Proof of Theorem 1. Consider

$$\begin{aligned}
&\mathbb{E}\left[\left(m_{l_n,n}^{\text{KNNR}}(u) - m(u)\right)^2\right] \\
&\quad \le 2\mathbb{E}\left[\left(m_{l_n,n}^{\text{KNNR}}(u) - m_{l_n,n}^{\text{NN}}(u)\right)^2\right] + 2\mathbb{E}\left[\left(m_{l_n,n}^{\text{NN}}(u) - m(u)\right)^2\right]
\end{aligned}$$

where second term vanishes as $n \to \infty$ by Theorem 3. For the first term,

$$\mathbb{E}\left[\left(\sum_{\tau=0}^{T} m_{l_n,n}^{\mathrm{KNNR}}(u,\tau) - m_{l_n,n}^{\mathrm{NN}}(u,\tau)\right)^2\right]$$
$$\leq (T+1)\sum_{\tau=0}^{T}\mathbb{E}\left[\left(m_{l_n,n}^{\mathrm{KNNR}}(u,\tau) - m_{l_n,n}^{\mathrm{NN}}(u,\tau)\right)^2\right].$$

It will be sufficient to show that

$$\lim_{n\to\infty}\mathbb{E}\left[\left(m_{l_n,n}^{\mathrm{KNNR}}(u,\tau) - m_{l_n,n}^{\mathrm{NN}}(u,\tau)\right)^2\right] = 0$$

for a fixed τ since the arguments are analogous for the other terms. Consider

$$\begin{aligned}
&\mathbb{E}\left[\left(\frac{1}{l_n}\sum_{j=1}^{l_n}\sum_{k^\tau}^{K_n} r_\tau^{k^\tau}(\tilde{X}_j,u)(I_{k^\tau=\mathcal{K}_j^\tau} - \frac{1}{K_n^\tau})\right)^2\right] \\
&= \frac{1}{l_n^2}\mathbb{E}\left[\sum_{j_1=1}^{l_n}\sum_{j_2=1}^{l_n}\sum_{k_1^\tau}^{K_n}\sum_{k_2^\tau}^{K_n} r_\tau^{k_1^\tau}(\tilde{X}_{j_1},u) r_\tau^{k_2^\tau}(\tilde{X}_{j_2},u)(I_{k_1^\tau=\mathcal{K}_{j_1}^\tau} - \frac{1}{K_n^\tau})(I_{k_2^\tau=\mathcal{K}_{j_2}^\tau} - \frac{1}{K_n^\tau})\right] \\
&= I_n
\end{aligned}$$

Since $\mathbb{E}[I_{k^\tau=\mathcal{K}_j^\tau}] = \frac{1}{K_n^\tau}$ for all j and due to the independence assumptions on $\mathcal{K}_j^\tau$, we have for $j_1 \neq j_2$ and any k_1^τ and k_2^τ:

$$\begin{aligned}
&\mathbb{E}\left[r_\tau^{k_1^\tau}(\tilde{X}_{j_1},u) r_\tau^{k_2^\tau}(\tilde{X}_{j_2},u)(I_{k_1^\tau=\mathcal{K}_{j_1}^\tau} - \frac{1}{K_n^\tau})(I_{k_2^\tau=\mathcal{K}_{j_2}^\tau} - \frac{1}{K_n^\tau})\right] \\
&= \mathbb{E}\left[r_\tau^{k_1^\tau}(\tilde{X}_{j_1},u) r_\tau^{k_2^\tau}(\tilde{X}_{j_2},u)\right]\mathbb{E}\left[I_{k_1^\tau=\mathcal{K}_{j_1}^\tau} - \frac{1}{K_n^\tau}\right]\mathbb{E}\left[I_{k_2^\tau=\mathcal{K}_{j_2}^\tau} - \frac{1}{K_n^\tau}\right] \\
&= 0
\end{aligned}$$

For $j_1 = j_2$ and $k_1^\tau \neq k_2^\tau$,

$$\begin{aligned}
&\mathbb{E}\left[r_\tau^{k_1^\tau}(\tilde{X}_{j_1},u) r_\tau^{k_2^\tau}(\tilde{X}_{j_2},u)(I_{k_1^\tau=\mathcal{K}_{j_1}^\tau} - \frac{1}{K_n^\tau})(I_{k_2^\tau=\mathcal{K}_{j_2}^\tau} - \frac{1}{K_n^\tau})\right] \\
&= \mathbb{E}\left[r_\tau^{k_1^\tau}(\tilde{X}_{j_1},u) r_\tau^{k_2^\tau}(\tilde{X}_{j_2},u)\right]\left(-\frac{1}{K_n^{2\tau}}\right)
\end{aligned}$$

since $\mathbb{E}[I_{k_1^\tau=\mathcal{K}_{j_1}^\tau} I_{k_2^\tau=\mathcal{K}_{j_2}^\tau}] = 0$. For $j_1 = j_2$ and $k_1^\tau = k_2^\tau$,

$$\begin{aligned}
&\mathbb{E}\left[r_\tau^{k_1^\tau}(\tilde{X}_{j_1},u) r_\tau^{k_2^\tau}(\tilde{X}_{j_2},u)(I_{k_1^\tau=\mathcal{K}_{j_1}^\tau} - \frac{1}{K_n^\tau})(I_{k_2^\tau=\mathcal{K}_{j_2}^\tau} - \frac{1}{K_n^\tau})\right] \\
&= \mathbb{E}\left[(r_\tau^{k_1^\tau}(\tilde{X}_{j_1},u))^2\right]\left(\frac{1}{K_n^\tau}\right)
\end{aligned}$$

where we used that $I_{k^\tau=\mathcal{K}_j^\tau}$ is Bernoulli distributed. Since for all k_1^τ, k_2^τ and for all j_1, j_2, it holds that $\mathbb{E}\left[(r_\tau^{k_1^\tau}(\tilde{X}_{j_1},u))^2\right] \leq 4M^2$ and $\mathbb{E}\left[|r_\tau^{k_1^\tau}(\tilde{X}_{j_1},u) r_\tau^{k_2^\tau}(\tilde{X}_{j_2},u)|\right] \leq 4M^2$. Thus,

$$
\begin{aligned}
I_n &\leq \frac{1}{l_n^2} \sum_{j=1}^{l_n} \left(\sum_{k^\tau}^{K_n} \mathbb{E}\left[(r_\tau^{k_1^\tau}(\tilde{X}_j, u))^2 \right] \left(\frac{1}{K_n^\tau} \right) \right. \\
&\quad \left. + \sum_{k_1^\tau}^{K_n} \sum_{\substack{k_2^\tau \\ k_1^\tau \neq k_2^\tau}}^{K_n} \mathbb{E}\left[|r_\tau^{k_1^\tau}(\tilde{X}_j, u) r_\tau^{k_2^\tau}(\tilde{X}_j, u)| \right] \left(\frac{1}{K_n^{2\tau}} \right) \right) \\
&\leq \frac{1}{l_n^2} \sum_{j=1}^{l_n} 8M^2 = \frac{8M^2}{l_n} \to 0
\end{aligned}
$$

as $n \to \infty$. □

3.6 Conclusion

We believe that this work offers multiple possibilities for future research. First, bridging between reinforcement learning and stochastic control, we expect that the K-NN resampling algorithm can be applied to a multitude of practical stochastic control problems in diverse fields such as finance, operations research or engineering. Another interesting avenue of research is how to choose the state representation for our algorithm. With the right state representation the curse of dimensionality can be overcome in the nearest neighbor search [166] or it could be applied to non-Markovian control problems. Related to this, the metric that is used for nearest neighbor search can be viewed as an engineering choice and potentially improved performance can be seen with a learned metric as in [52]. Our generalization of the local averaging theorem can also be employed to guide the design of other OPE methods based on local averaging. This could include kernel-based or other partitioning-based approaches.

Chapter 4

Limit Order Book Simulation and Trade Evaluation with K-Nearest-Neighbor Resampling

4.1 Introduction

Central limit order books (LOBs) are the prevalent organisational mechanism for trading in a wide range of assets, from stocks and bonds to derivatives. A LOB is a centralized registry of traders' commitments to buy or sell a certain amount of an asset for a certain price, referred to as limit orders. Other traders may choose to execute a certain amount against those limit orders, by posting market orders. The appeal of LOBs is their ability to efficiently aggregate traders' preferences via limit and market orders. Due to their practical importance and the complex interaction patterns present in them, LOBs are a widely researched subject in academia and industry. From a practical perspective, an important question is to understand how well a trader's strategy performs within a limit order book and how the market reacts to this trader's strategy, noting that the placement of orders can have an adversarial effect (see, e.g. [67]).

Direct testing of trading strategies within an actual market is inherently risky and can lead to sizeable losses or a disorderly market. On the other hand, large amounts of historical data for LOBs are available. In reinforcement learning, the subfield of off-policy evaluation deals with the problem of how to extract information about the performance of a strategy using such observational data. In this context, we proposed K-nearest neighbor (K-NN) resampling, a method that has theoretical convergence guarantees under general conditions, performs well in experiments in environments with intrinsic noise, is easy to implement and computationally efficient [69]. The main idea of K-NN resampling is that similar states in a metric sense lead to similar transitions of the underlying random dynamical system. Given an initial state, the algorithm selects at random a new state from a list of nearest neighbors from the initial state. From this neighboring state, the historical transition from the data is used leading to a subsequent state. Chaining these random matchings together yields an approximate path for the underlying system and repeatedly restarting at the initial states leads to a collection of paths, i.e. a simulation. In this chapter, we show how we can apply this idea for the simulation of LOBs and the evaluation of trading strategies within LOBs.

The academic interest in the simulation of LOBs is well documented by the broad and recent literature. A current literature review of this subject matter [109] identifies four major approaches to LOB simulations: Point processes [11, 19, 31, 51, 103, 111, 141, 150], agent based modelling [18, 34, 164, 192], stochastic differential equations models [49, 50, 101, 104, 123] and deep/statistical learning [46, 48, 106, 135, 138, 157, 193]. We note that the first three approaches usually require explicit modelling choices on either market dynamics or

interactions of market participants. While this view allows for the direct choice of relevant market characteristics, it poses the risk of fundamental modelling biases in the market dynamics. For applications in actual markets, these approaches often require the additional non-trivial step of calibration to market data and may require a high level of granularity in the data.

The algorithm we propose can be understood as a statistical learning method to order book simulation. In particular, our algorithm selects transitions between LOB snapshots using a non-parametric approach. This is most similar to the scope and methodology proposed in [48]. The authors also sample transitions between LOB snapshots where the sampling distribution is given by a conditional generative adversarial neural network (CGAN). They show that their model is capable of reproducing stylized facts observed in the market and that the impact of trading within their simulator behaves as expected. Since the scope of our work is partially similar, we reuse some of the stylized facts considered in this paper for testing our model's quality. Furthermore, we use the CGAN approach as a benchmark and see that our method outperforms it on key statistics for simulation quality. Beyond the scope of [48], we also consider the evaluation and selection of trading strategies using limit orders.

The other papers using deep learning for LOB simulation often have a slightly different focus. Namely, [46, 106, 135, 138, 157, 193] considers event-by-event simulation within the LOB rather than the transition between LOB snapshots as in [48] and in this chapter. We note that this may depend on the objective of the simulation. Event-by-event simulation are particularly useful for ultra-high frequency traders where a single event may cause the trader to react. Considering transitions between LOB snapshots on the other hand is often sufficient for practical applications in high to medium-high frequency trading. Note that our algorithm could be readily modified for event-by-event simulations – we leave this investigation for future work.

Furthermore, all published statistical learning models that we are aware of use deep learning for generating simulations. In contrast, our method only uses nearest neighbor search. Not relying on deep learning models has several advantages. First, our algorithm does not need any optimization. In contrast, deep learning models require solving a highly non-convex optimization problem where optimization procedures often exhibit instability, as observed in [48]. Furthermore, our method can be efficiently implemented using standard machine learning packages and purely relies on historical observations rather then sampling from a black box distribution. Finally, it does not require extensive hyperparameter tuning.

Since we apply our algorithm to a LOB with a pro-rata type matching mechanism, we also highlight [18], which is to our knowledge the only LOB simulator explicitly used for pro-rata markets. In contrast to our work, the focus of their work is on the explanation of market phenomena by aggregation of agents rather than simulating a specific market or evaluating trading strategies. Furthermore, our method is applicable to different types of execution mechanisms and pro-rata type matching is only exploited for evaluating trading strategies with limit orders (see Section 4.6).

Our contributions in this chapter can be summarized as follows:

1. We provide a framework for how K-NN resampling can be adapted for simulating LOBs and how to incorporate trades within these simulations.

2. We describe how K-NN resampling can be used for evaluating trading strategies with both limit and market orders.

3. We provide empirical evidence that our method is capable of realistically simulating LOB dynamics with and without trade interventions. In particular, we show that our method can outperform a strong deep learning-based benchmark on key statistics.

4. We demonstrate how K-NN resampling can be used to evaluate and choose trading strategies in a LOB with a pro-rata type matching mechanism. Specifically, we show how to calibrate the quantity of limit orders with our resampling algorithm.

5. We illustrate how K-NN resampling can effectively be used in a higher dimensional state space and how the algorithm can be combined with dimension reduction techniques.

We organize the chapter as follows. Section 4.2 introduces our notation and assumptions for LOBs. Section 4.3 describes our algorithm and how it can be used for LOB simulations. Section 4.4 details the data set that we use to test the resampling algorithm. Section 4.5 explores the quality of the simulations, with and without trade interventions. In Section 4.6, we demonstrate how trading strategies with limit orders can be evaluated and chosen based on our simulation approach. Section 4.7 considers a higher dimensional state space and usage of dimension reduction for the resampling. Section 4.8 concludes the chapter.

Notation: $\mathbb{R}_{>0}$ and $\mathbb{N}_{>0}$ denotes the space of positive real and natural numbers, respectively. For $\delta \in \mathbb{R}_{>0}$, $\delta\mathbb{N}_{>0}$ denotes all the positive multiples of δ.

4.2 Limit order books

This section describes our notation and assumptions for LOBs as necessary for a precise definition of our algorithm.

4.2.1 Set-up

Consider a trading time horizon $[0, T]$ where $T \in \mathbb{R}_{>0}$ and the tick size $\delta \in \mathbb{R}_{>0}$, then we can describe market orders, limit orders and limit order cancellations as follows:

- A limit order is a quadruple $x = (t, p, q, r)$ where $t \in [0, T]$ is the time the order is placed, $p \in \delta\mathbb{N}_{>0}$ is the price level, $q \in \mathbb{N}_{>0}$ is the quantity and $r \in \{\text{bid}, \text{ask}\}$ is the order side.
- A market order is a triple $y = (t, q, r)$ where $t \in [0, T]$ is the time the order is placed, $q \in \mathbb{N}$ is the quantity and $r \in \{\text{buy}, \text{sell}\}$ is the aggressor side of the order.
- An order cancellation for a limit order $(t, p, \hat{q}, r)$ with remaining volume $\hat{q} \geq 0$ and $t < T$ is a 5-tuple $c = (t, \hat{t}, p, \hat{q}, r)$ where $\hat{t} \in [t, T]$ is the time of cancellation.

The limit order book is the set of all active limit orders $L(t)$ at a given point in time $t \in [0, T]$. $L_p(t)$ denotes the set of all active limit orders at time $t \in [0, T]$ and price level $p \in \delta\mathbb{N}_{>0}$. We refer to $L_p(t)$ as LOB level. Note that each LOB level with active limit orders (i.e. $L_p(t) \neq \emptyset$) can be uniquely assigned to either order side at any point in time as otherwise LOB levels containing both bid and ask side would be executed against each other. This assignment is ordered and unique in the sense that there is a (non-unique) price $\hat{p} \in \mathbb{R} \setminus \delta\mathbb{N}$ for which all elements of $L_p(t)$ are bid orders, if $p < \hat{p}$, and ask orders, if $p > \hat{p}$. We denote the set of all such prices by $\hat{P}$. To deal with LOB levels with zero volumes in the description of the algorithm, we introduce the following auxiliary tools. Namely, we assume we have a procedure h available [1] that maps the set $\hat{P}$ onto a unique element $p^* \in \hat{P}$, i.e. $p^* = h(\hat{P})$. We call p^* the (unique) dividing price and we assign all levels $L_p(t)$ where $p < p^*$ to the bid side and $p > p^*$ to the ask side of the book, even for empty LOB levels (i.e. $L_p(t) = \emptyset$. Furthermore, the volume $V_p(t)$ at a LOB level $L_p(t)$ is defined as $V_p(t) = \sum_{x \in L_p(t)} q_x$, if $L_p(t) \neq \emptyset$, and $V_p(t) = 0$, if $L_p(t) = \emptyset$.

For the dividing price $p^* \in \mathbb{R} \setminus \delta\mathbb{N}$, we denote the largest $p \in \delta\mathbb{N}$ for which $p < p^*$ by $\lfloor p^* \rfloor$ and define $\lceil p^* \rceil$ analogously. We say $\lfloor p^* \rfloor$ is the best virtual bid price and $\lceil p^* \rceil$ the best virtual ask price[2]. This notation allows

[1] A simple choice for h is $h(\cdot) = (\inf(\cdot) + \sup(\cdot))/2 + \varepsilon$ where $\varepsilon > 0$. In our practical implementation (Section 4.5), we use a slightly more involved assignment procedure to reduce the risk of systematic biases. Namely, if there is no empty LOB levels in the spread, we assign p^* the (unweighted) mid-price. If there is an empty LOB level in the spread, we try to assign it to either ask or bid based on which side was last active on the corresponding level and calculate p^* as the mid price based on this information. If this is not feasible, we fall back to the simple procedure described above.

[2] We describe the prices as "virtual" since the price levels $\lfloor p^* \rfloor$ and $\lceil p^* \rceil$ can potentially have zero order volume. Note that the conventionally defined best bid price and the best ask price correspond to their virtual counterparts in the one-tick-spread case. Since our data set mostly consists of one-tick-spreads, we refrain from making this distinction going forward.

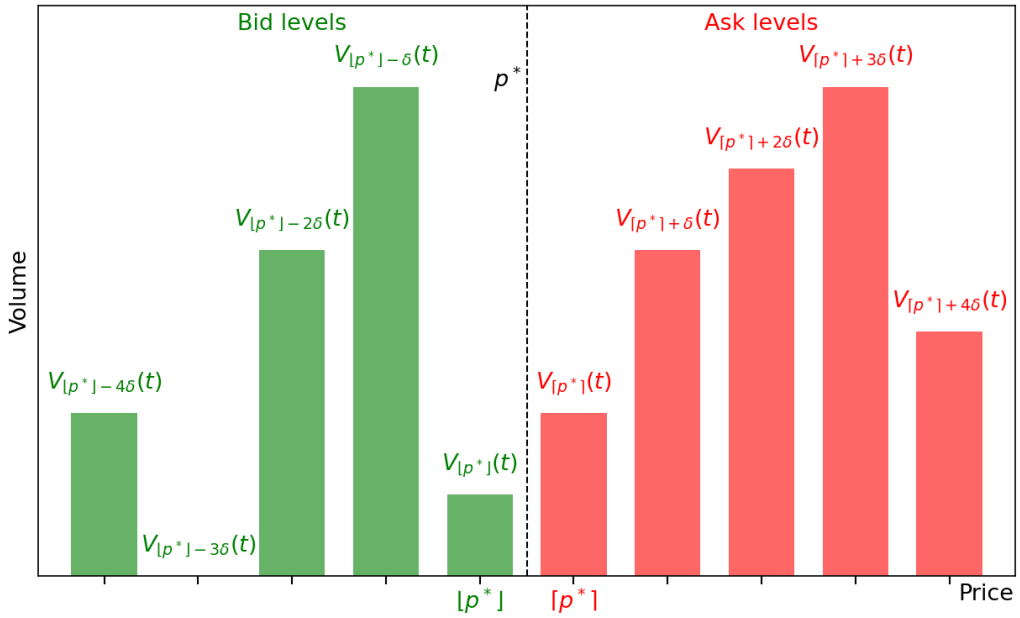


Figure 4.1: LOB snapshot $S^5_{p^*}(t)$ at time t with dividing price p^*, 5 levels on each market side and tick size δ.

us to define a centered LOB snapshot with l levels. Let $l \in \mathbb{N}_{>0}$ and $p^* \in \mathbb{R} \setminus \delta\mathbb{N}$ a dividing price with $\lfloor p^* \rfloor > l\delta$, then the LOB snapshot with l levels is defined as

$$S^l_{p^*}(t) = (V_{\lfloor p^* \rfloor - (l-1)\delta}(t), \dots, V_{\lfloor p^* \rfloor}(t), V_{\lceil p^* \rceil}(t), \dots, V_{\lceil p^* \rceil + (l-1)\delta}(t)).$$

The first l entries correspond to the bid levels while the last l entries are for the ask values. Figure 4.1 shows an example for an LOB snapshot.

Choosing the smallest $k_b, k_a \in \mathbb{N}$ such that $V_{\lfloor p^* \rfloor - (k_b-1)\delta}(t) > 0$ and $V_{\lceil p^* \rceil + (k_a-1)\delta}(t) > 0$, the (unweighted) mid-price is given by

$$p_m(t) = ((\lfloor p^* \rfloor - (k_b - 1)\delta) + (\lceil p^* \rceil + (k_a - 1)\delta))/2.$$

The weighted mid-price is then given by

$$p_w(t) = \frac{(\lfloor p^* \rfloor + (k_b - 1)\delta)V_{\lfloor p^* \rfloor - (k_b-1)\delta}(t) + (\lceil p^* \rceil - (k_a - 1)\delta)V_{\lceil p^* \rceil + (k_a-1)\delta}(t)}{V_{\lfloor p^* \rfloor - (k_b-1)\delta}(t) + V_{\lceil p^* \rceil + (k_a-1)\delta}(t)}.$$

Figure 4.2 demonstrates for an example LOB snapshot how prices can differ for different price definitions. The dividing price is seen as given (see footnote) while the mid-price and the weighted mid-price depend on the current shape of the order book.

Furthermore, we define the return for either the mid-price or the weighted price as the log-return, i.e.

$$\ln p_\cdot(t) - \ln p_\cdot(\tilde{t})$$

where $\tilde{t} < t$. Finally, we define analogous to [37] the order book imbalance (OBI) as

$$\rho(t) = \frac{V_{\lfloor p^* \rfloor}(t) - V_{\lceil p^* \rceil}(t)}{V_{\lfloor p^* \rfloor}(t) + V_{\lceil p^* \rceil}(t)} \in [-1, 1]$$

if $V_{\lfloor p^* \rfloor}(t) + V_{\lceil p^* \rceil}(t) > 0$ and $\rho(t) = 0$ else.

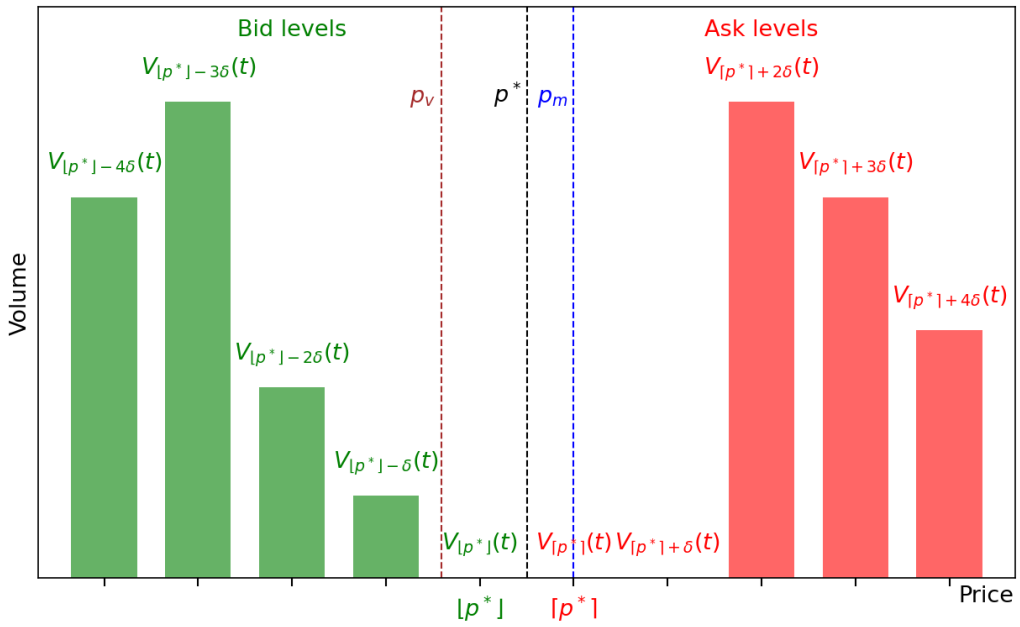


Figure 4.2: LOB snapshot $S^3_{p^*}(t)$ at time t with dividing price p^*, mid-price p_m and weighted mid-price p_v.

4.2.2 Assumptions on limit order book dynamics

Limit order books are usually seen as evolving continuously over time, where market orders and limit orders may arrive at any time. The arriving orders are often modelled as random but reactive to the current state of the LOB (see e.g. [51, 176]). We focus on a discretization that allows us to consider high to medium-high frequency trade decisions. In particular, we assume that the LOB evolves as a discrete random dynamical system transitioning from one LOB state-price pair (Z_t, p^*_t) to the next, $(Z_{t+\Delta t}, p^*_{t+\Delta t})$, where $\Delta t \in \mathbb{R}_{>0}$ is the time increment and Z_t the state representation for the LOB. We note that the state representation can be chosen in many different ways and is not limited to LOB snapshots. We discuss possible choices of the state below. Using transition functions f_Z, f_p and a random noise ε, a single LOB transition is described by

$$(Z_{t+\Delta t}, p^*_{t+\Delta t}) = (f_Z(Z_t, \varepsilon), f_p(Z_t, p^*_t, \varepsilon)).$$

For an initial state-price pair (Z_0, p^*_0), the LOB dynamics are then defined by recursively applying the transition functions, i.e.

$$(Z_{(k+1)\Delta t}, p^*_{(k+1)\Delta t}) = (f_Z(Z_{k\Delta t}, \varepsilon_{k\Delta t}), f_p(Z_{k\Delta t}, p^*_{k\Delta t}, \varepsilon_{k\Delta t}))$$

where we assume that the noise is identically and independently distributed (iid) for all $s \in \mathbb{N}$.

We choose to model the price explicitly in order to facilitate the assignment of empty tick sizes in the LOB snapshots and we assume that the price level does not have an impact on the shape of the subsequent order book. Furthermore, the iid assumption on the system noise implies that the system is Markovian and a Markovian dynamical system is necessary for the resampling algorithm as explained in the previous chapter. Note that this assumption is not very restrictive as all time-homogenous Markov processes can be generated via a random dynamical system with iid random noise, i.e. [6, 27]. Furthermore, recall that any discrete-time non-Markovian dynamic system can be made Markovian by including its history in the state, i.e. [6]. Finally, random dynamical system formulations for LOBs are common in the literature and can, for example, be found in [1, 49, 51, 101].

Note that the LOB state can be chosen in potentially many different ways depending on the investigation. We will focus on a Markovian interpretation of the LOB snapshots where we use

$$Z_t = S^l_{p^*_t}(t)$$

for some LOB depth l. Other potential choices of Z_t can include a statistic on the history of LOB snapshots to account for non-Markovian behavior of order books [195] or information on seasonalities (e.g. distance to fixing times) and other external events impacting the order book (e.g. trades in a derivative order book).

4.2.3 Trading agent in the limit order book

We describe how an additional trading agent can be added to the random dynamical system for the purpose of interacting with the LOB or evaluating trading strategies. We assume that for a trader their trading decision is based on the current state of the LOB and personal information. This personal information can contain a wide variety of variables. For example, it can include information on the trader's inventory and their risk profile, but it also could include alpha signals or information on other active traders in the market. Usually, we assume that the information contains the current LOB position $P(t)$ of the trader. A trader's LOB position is the set of all active limit orders placed by the trader.

We formalize the interaction of a trading agent with the LOB by defining the trading strategy of a trader. Let u be a function mapping from an LOB state Z_t and trader information I_t to a triple consisting of a market order y, a set of cancellations of outstanding limit orders $\{c_j\}_{j=1}^m$ and a set of new limit orders $\{x_i\}_{i=1}^n$, i.e.

$$u(Z_t, I_t) = (\{c_j\}_{j=1}^m, y, \{x_i\}_{i=1}^n)^3.$$

To limit the possibility of self-fulfilment, we assume the following hierarchy in the execution in our simulation algorithm. First, a trader cancels some of their outstanding orders according to $\{c_j\}_{j=1}^m$, then, they execute their market order if the market order quantity is larger than zero and, finally, the new limit orders $\{x_i\}_{i=1}^n$ are placed.

4.2.4 Interaction with the limit order book

In the context of the random dynamical system formulation of the LOB dynamics, we make the assumption that we can incorporate the immediate effect of a trader's action on the LOB state. For this purpose, we define functions for limit orders and market orders that map onto a new LOB state.

For limit orders, we only consider the case where the order does not cross the spread[4]. For a $k \in \mathbb{N}$ with $k < l$, consider the limit order $x = (t, \lfloor p^* \rfloor - k\delta, q, \text{bid})$ and the LOB state-price pair $(S^l_{p^*_t}(t), p^*_t)$ at the same point in time, then the function

$$g_L((S^l_{p^*_t}(t), p^*_t), x) = (\tilde{S}^l_{p^*_t}(t), p^*_t)$$

where

$$\tilde{S}^l_{p^*_t}(t) = (\dots, V_{\lfloor p^* \rfloor - (k+1)\delta}(t), V_{\lfloor p^* \rfloor - k\delta}(t) + q, V_{\lfloor p^* \rfloor - (k-1)\delta}(t), \dots)$$

incorporates bid side limit orders. The extension to the ask side is analogous. The third plot in Figure 4.3 shows how the volume of a buy limit order and a sell limit order is placed in a centered LOB snapshot. Furthermore, we analogously define the function $g_C((S^l_{p^*_t}(t), p^*_t), c)$ where instead of adding the cancelled volume is subtracted from the appropriate place in the centered LOB snapshot (see first plot in Figure 4.3).

[3]Note that any trading strategy u is set valued in the first and third component. Thus, the cardinalities of the sets n and m also depend on the LOB state Z_t and the trader information I_t according to the trading strategy u.

[4]The general case where a limit order can cross the spread can be thought of as a straightforward combination of the immediate impact of a market order and a limit order where the mid-price shifts accordingly.

Consider the market order $y = (t, q, \text{buy})$ and a LOB state-price pair $(S^l_{p^*_t}(t), p^*_t)$ where $\sum_{i=0}^{k} V_{\lceil p^* \rceil + i\delta}(t) \geq q > \sum_{i=0}^{k-1} V_{\lceil p^* \rceil + i\delta}(t)$ for $k \in \mathbb{N}$ with $k \leq l-1$. Then,

$$g_M((S^l_{p^*_t}(t), p^*_t), y) = (\tilde{S}^l_{p^*_t}(t), p^*_t)$$

where

$$\tilde{S}^l_{p^*_t}(t) = (\dots, V_{\lfloor p^* \rfloor}(t), 0, \dots, 0, \sum_{i=0}^{k} V_{\lceil p^* \rceil + i\delta}(t) - q, V_{\lceil p^* \rceil + (k+1)\delta}(t), \dots)$$

is used to account for the immediate effect of a buy market order on the LOB state[5]. We visualize the placement of a buy market order in the second plot of Figure 4.3. A sell order can be treated analogously. For a market order, we can further directly observe the trade revenue. Using the buy order and the order book from above the trader's revenue is

$$-(\lceil p^* \rceil + k\delta)\left(\sum_{i=0}^{k} V_{\lceil p^* \rceil + i\delta}(t) - q\right) - \sum_{i=0}^{k-1}(\lceil p^* \rceil + i\delta) V_{\lceil p^* \rceil + i\delta}(t).$$

A sell order changes the sign of the trade revenue accordingly.

4.2.5 Limit order execution mechanisms

There are multiple different matching mechanisms used in different markets that decide which limit order is executed, and to what extent, for an incoming market order. The order book mechanism producing our data is CME's Allocation algorithm that is most prominently used for trading money market futures on the CME and is a variation of pro-rata matching.

In pro-rata matching, orders are matched according to their relative size compared to the entire volume at a price level. For instance, for a fixed price level p, let V_p be the volume in the order book and consider an arriving market order of size $q_M < V_p$. The executed amount of a limit order of size q_L is given by

$$\left\lfloor \frac{q_M q_L}{V_p} \right\rfloor,$$

where $\lfloor \cdot \rfloor$ rounds to the next smaller integer value[6]. Note that in this case only three values need to be known to determine the executed amount of a limit order. In particular, this allows us to observe the executed amount of limit orders by a trading agent by keeping track of the volume at the time market orders arrive and, thus, also allows us to calculate trade revenues of a limit order strategy. Pure pro-rata mechanisms are for example used for Eurex equity options and CME agricultural options.

In a variation of this, the Allocation algorithm prioritizes the execution of the first limit order placed at a more aggressive level over the limit orders arriving at the same level at a later point in time. An arriving market order is first executed against the aggressing limit order, if present. The remaining volume of the market order is executed against the non-aggressing limit orders on a pro-rata basis. For instance, for a fixed price level p, let V_p be the volume in the order book, q_A the size of the aggressing limit order and consider an arriving market order of size $q_A < q_M < V_p$. The executed amount of a non-agressing limit order of size q_L is then given by

$$\left\lfloor \frac{(q_M - q_A) q_L}{V_p - q_A} \right\rfloor.$$

[5] If $q > \sum_{i=0}^{\infty} V_{\lceil p^* \rceil + i\delta}(t)$, the market order is only partially filled to a volume of $\sum_{i=0}^{\infty} V_{\lceil p^* \rceil + i\delta}(t)$ and the aggressed side of the LOB will be empty $(\dots, V_{\lfloor p^* \rfloor}(t), 0, 0, \dots,)$. If $q < V_{\lceil p^* \rceil}(t)$, only the first level on the aggressed side of the LOB will change, i.e. $(\dots, V_{\lfloor p^* \rfloor}(t), V_{\lceil p^* \rceil}(t) - q, V_{\lceil p^* \rceil + \delta}(t), \dots)$.

[6] The amount of the market order not executed pro-rata due to the rounding error is usually executed with a different matching algorithm as residual. In this chapter we neglect residual execution although explicit modelling would be possible.

While for the Allocation algorithm it is slightly more complicated to determine the executed amount of a non-aggressing limit order, it only requires tracking one additional variable. Thus, it is still feasible to calculate trade revenues for limit order strategies. [7]

4.3 Algorithm

In this section, we explain how K-NN resampling can be applied to LOB simulation and trade evaluation, making use of the fact that we can observe the direct impact of a trading agent's decision on the LOB. We explain first how K-NN resampling as in [69] can be used for simulating the dynamics of LOBs with or without interaction before extending the algorithm for the evaluation of trading strategies.

Algorithm KNNR-LOB K-nearest neighbor resampling for LOB simulation

Input: Data set $\mathcal{D}$; trading strategy $u(\cdot,\cdot)$; order update functions g_C, g_M, g_L; nearest neighbors parameter K; path length T_n; number of resampled trajectories N

paths $\leftarrow [\cdot]$
for r from 1 to N **do**
  path $\leftarrow [\cdot]$
  Sample an initial state-price pair $(\widehat{S^l_{p_0}}(0), \widehat{p_0})$ from $\mathcal{D}$ and append path with $(\widehat{S^l_{p_0}}(0), \widehat{p_0})$
  for s from 0 to $T_n - 1$ **do**
    Get agent's orders from the trading strategy $u(\widehat{S^l_{p_s}}(s), s) = (\{c_j\}_{j=1}^m, y, \{x_i\}_{i=1}^n)$
    Apply agent's orders to current LOB snapshot, i.e.

$$(\widetilde{S^l_{p_s}}(s), \widetilde{p_s}) = \prod_{j=1}^{n} g_L \circ g_M \circ \prod_{i=1}^{m} g_C \left(\widehat{S^l_{p_s}}(s), \widehat{p_s}, (\{c_j\}_{j=1}^m, y, \{x_i\}_{i=1}^n)\right).$$

    Randomly choose k from $1, \ldots, K$
    Find the k-nearest neighbor of $\widetilde{S^l_{p_s}}(s)$ in $\mathcal{D}$ and denote it by $\overline{S^l_{p^*_{t_j}}}(t_j)$
    Look up the corresponding price $\overline{p^*_{t_j}}$, subsequent price $\overline{p^*_{t_j+\Delta t}}$ and subsequent state $\overline{S^l_{p^*_{t_j+\Delta t}}}(t_j + \Delta t)$ to $\overline{S^l_{p^*_{t_j}}}(t_j)$ in $\mathcal{D}$
    Update price $\widehat{p_{s+1}} \leftarrow \widehat{p_s} + (\overline{p^*_{t_j+\Delta t}} - \overline{p^*_{t_j}})$
    Update state $\widehat{S^l_{p_{s+1}}}(s+1) \leftarrow \overline{S^l_{p^*_{t_j+\Delta t}}}(t_j + \Delta t)$
    Append path with $(\widehat{S^l_{p_{s+1}}}(s+1), \widehat{p_{s+1}})$
  end for
  Append paths with path
end for

Output: paths

[7] Another popular matching mechanism is the first-in-first-out (FIFO) algorithm. The queue structure of FIFO matching makes following one particular limit order a relatively high dimensional problem as one needs to track all other limit orders that are currently ahead in the queue.

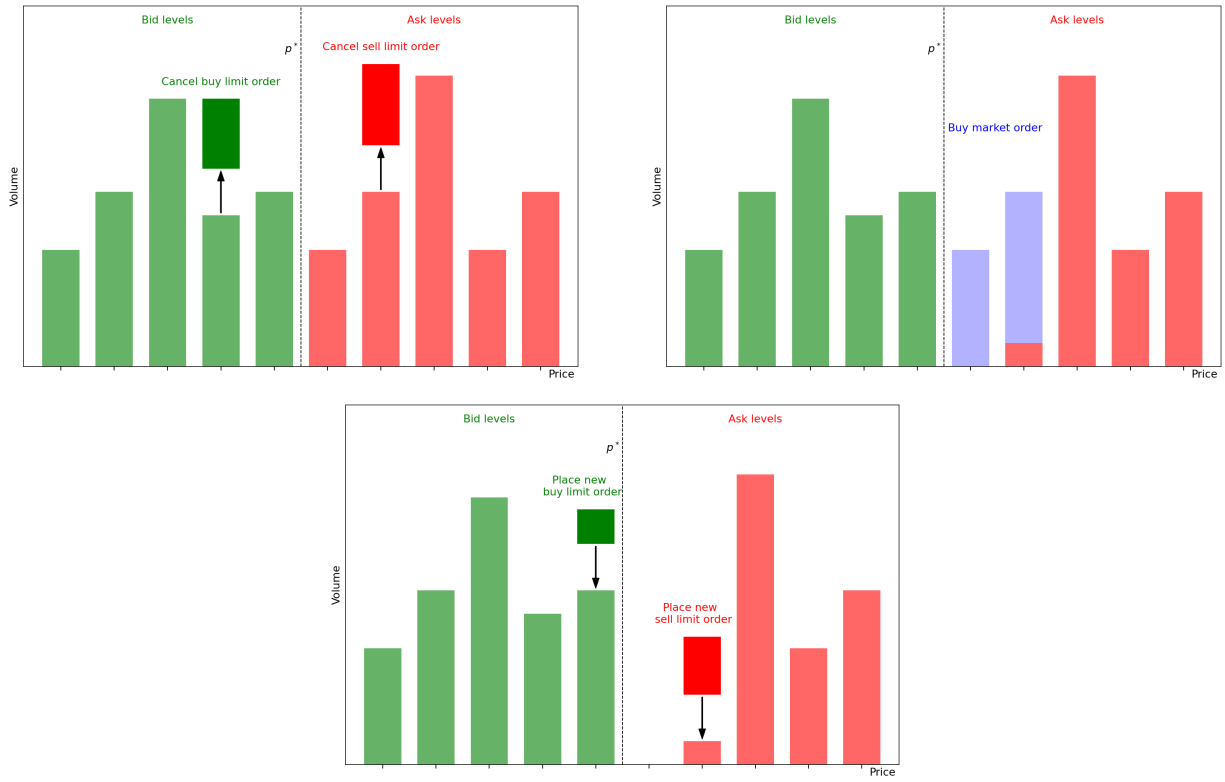


Figure 4.3: This figure gives an example for the incorporation of actions from a trading agent into the LOB (cf. line 7 in Algorithm KNNR-LOB). In the first plot, the trader cancels a buy limit order and a sell limit order. Then, in the second plot, a buy market order is placed depleting some of the liquidity in the LOB. In the third plot, the trading agent then places a new buy limit order and a new sell limit order in the LOB.

4.3.1 K-NN resampling for LOB simulation with a trading agent

We give a full description of K-NN resampling for LOB simulation in Algorithm KNNR-LOB. We assume that we have access to a data set containing LOB transitions $\mathcal{D} = \{(S^l_{p^*_{t_i}}(t_i), p^*_{t_i,i}, S^l_{p^*_{t_i+\Delta t,i}}(t_i + \Delta t), p^*_{t_i+\Delta t,i})\}_{i=1}^n$ and a trading agent with trading strategy $u(S^l_{p^*_s}(s), s)$ which takes an LOB snapshot $S^l_{p^*_s}(s)$ and a time $s \in [0, T]$ as input. The trading strategy then gives us the cancellations, the market order and the limit orders of the trading agent. We choose uniformly randomly an initial state $(\widehat{S^l_{p_0}}(0), \widehat{p_0}) = (S^l_{p^*_{t_i}}(t_i), p^*_{t_i,i})$. We then pass the state information to the trading strategy to get the trader's action $(\{c_j\}_{j=1}^m, y, \{x_i\}_{i=1}^n) = u(\widehat{S^l_{p_0}}(0), 0)$. The trader's actions are then incorporated using the functions $g_C(\cdot), g_M(\cdot)$ and $g_L(\cdot)$ leading to the modified state $\widetilde{S^l_{p_0}}(0)$ and price $\widetilde{p_0}$ corresponding to line 7 in Algorithm KNNR-LOB. We visualize an example for the steps of line 7 in Algorithm KNNR-LOB in Figure 4.3. We choose uniformly at random one of the K nearest neighbors of $\widehat{S^l_{p_0}}(0)$ and denote it by $\overline{S^l_{p^*_{t_j}}}(t_j)$. We then assign $\widehat{S^l_{p_1}}(1) = \overline{S^l_{p^*_{t_j+\Delta t}}}(t_j + \Delta t)$ and calculate the new price $\widehat{p_1} = \widehat{p_0} + (\overline{p^*_{t_j+\Delta t}} - \overline{p^*_{t_j}})$[8]. Repeating this procedure T_n-times for $T_n \in \mathbb{N}_{>0}$ gives us one simulated path. We can repeat the path simulation and use the saved paths for our analysis.

Algorithm KNNR-LOB is a so-called 'lazy learning' algorithm as it does not require any optimization. This

[8]Note that this is feasible since we assumed that the price level has no impact on the state transition.

is in contrast to generative deep learning models, where training can be unstable. However, the method still provides a large variability in path generation as for any initial state the algorithm is able to build up to K^{T_n} different possible paths which are all equally likely to occur. Furthermore, Algorithm KNNR-LOB can be parallelized and efficiently implemented with a tree-based nearest neighbor search using standard machine learning libraries. Also, it only requires the tuning of one hyper-parameter – the number of nearest neighbors. These considerations make Algorithm KNNR-LOB easy to implement and to run on any given data set. We refer the reader to [69] for more details on implementation and computational considerations.

Remark 9.

1. *For simplicity, we assume that the LOB state is given by centered LOB snapshots. For incorporating a trader's action in a more general LOB state we can introduce a mapping that uses the previous LOB state and the modified LOB snapshot to update the LOB state.*

2. *We assume a trading strategy for the trading agent in this section that does not require the explicit modelling of the trader's information process. For the evaluation of trading strategies, it may be necessary to do this, for example, by keeping track of the trader's position.*

3. *If a simulation without trade intervention is desired, one can simply skip lines 6 and 7 of Algorithm KNNR-LOB.*

4. *In our experiments, we choose the nearest neighbor parameter $K = 20$ heuristically, which is common practice in machine learning, i.e. [96]. We note that most theoretical results on the choice of K only give growth rates in dependence of the data set size, i.e. [87], rather then prescribing a specific K for a fixed data set size. An additional consideration, beyond the classical bias-variance trade-off in nearest neighbor regressions is that the parameter K in Algorithm KNNR-LOB also determines the number of possible paths that can be generated and may significantly impact the computational costs of the algorithm.*

5. *As metric for the nearest neighbor search, we use Euclidean distances to keep the implementation as simple as possible. Other choices could be a weighted Euclidean norm putting more emphasis on order book levels closer to the dividing price or an unbalanced Wasserstein metric (e.g. [191]) emphasising a mass interpretation of the order book. To us, the appropriate notion of a metric in a LOB space is an open research question and is beyond the scope of this current work.*

4.3.2 K-NN resampling for trading strategy evaluation

Algorithm KNNR-LOB cannot only be used for LOB simulations but with slight adjustments also for the evaluation of trading strategies. In particular, the trader's information process may need to be modelled explicitly to keep track of the trader's current position, inventory and trading revenues. The agent's trading strategy may depend on this additional information. To gather the information on position, inventory and trading revenues, we record all the trades executed against either limit orders or market orders entered by the trader. For market orders, this is feasible for any common execution mechanism. For limit orders, keeping track is directly possible in LOBs using either pro-rata or CME's Allocation execution mechanism. We do this by replaying the trades that occurred between the LOB snapshots in the historical data set and using CME's Allocation mechanism for attribution as explained in the previous section. For FIFO LOBs, one could potentially use a parametric or non-parametric model for the execution of limit orders.

4.4 Data set

We use trading data over a period of two years for 3-month Secured Overnight Financing Rate (SOFR) futures traded at the CME from January 2022 to January 2024 (see first plot in Figure 4.4). SOFR replaced the USD

LIBOR as the main overnight interest rate in the US and is based on actual repo transactions on US treasuries [98]. The 3-month SOFR futures traded at the CME replaced Eurodollar futures and are a main source for price discovery in short-term interest rates. The contracts are used by traders for hedging short-term interest rates exposures and for yield curve and spread strategies. Furthermore, they are highly liquid with trading of over 5 million contracts each day on average [45]. For execution, the futures are centrally cleared using a limit order book where limit orders are executed according to CME's Allocation algorithm.

We use daily level 2 LOB data from CME, which contains all order book movements but does not allow us to follow individual orders. For the order book, we are given the first 5 levels with active orders for both bid and ask side with tick size $\delta = 0.005$. Additionally, we observe all executed trades from the same source and match this data with the LOB data over the exchange time stamp. We restrict ourselves to the contracts SR4H4, SR4M4, SR4U4 and SR4Z4 (see first plot in Figure 4.4 for monthly trading activity during data collection) and treat them as a single data set[9]. Furthermore, we only consider the liquid trading times from 12pm to 8pm New York time. For an exemplary day, the second and third plot in Figure 4.4 support this choice.

After compiling this raw data, we aggregate it by taking a snapshot of the LOB every 250 order book events. We record all trades, order book sizes at trade execution and limit order sizes of aggressions to new ticks occurring between the snapshots to allow us to evaluate limit order trading strategies according to CME's Allocation algorithm. This leads to a data set with around 1.2×10^7 samples where each sample contains an initial LOB snapshot and price and the corresponding LOB snapshot and price after 250 order book events. Additionally, a sample contains the information as described above on all trades that occurred during the 250 order book events. To simplify the replay of historical LOB dynamics for testing, we order the transitions chronologically by contracts.

Remark 10 (Event time vs. real time)**.** *Note that in our experiments the time we consider is event time. This means we take each change in the order book as a discrete time step and the elapsed time corresponds to the number of order book changes since the beginning of the time period. We make this modelling decision as it allows the trader to automatically adapt their trading speed with market activity. Algorithm KNNR-LOB can equally be applied to real time scenarios.*

4.5 Simulation results

This section presents and discusses different outputs from applying Algorithm KNNR-LOB to real LOB data. In parts, we will use evaluation methods for LOB simulations proposed in [46, 48, 106, 157]. First, we will focus on the simulation capability of Algorithm KNNR-LOB without any interactions by a trading agent. If we compare the simulation results to actual market replays, we consider out-of-sample performance. This means we divide the data set in two parts with a 80%-to-20% split[10]. We use the smaller set as ground truth and for initial states in resampling. The larger set is used as data source for the resampled transitions. We use this splitting procedure to ensure that the shown simulation results are not biased from reusing the actual market observations. For investigating market impact, we then consider simulations where a trading agent either places market or limit orders.

If not mentioned otherwise, the results are obtained with the following choices of parameters. As nearest neighbor parameter, we use $K = 20$. For the episode length, we choose $T_n = 60$ transitions[11] corresponding to 1.5×10^4 order book events. The experiment evaluation is based on $N = 10^4$ resamplings.

[9] This is motivated by the fact that all contracts are futures on the same underlying and by the observation in [195] that models trained on LOB data from multiple assets tend to perform better in replicating price dynamics compared single asset models.

[10] We split our data set deterministically in accordance to the ordering introduced in Section 4.4. This means that testing data stems mostly from a single contract and chronological consistency of the split is kept within each contract, see e.g. [182].

[11] The number of transitions was chosen for computational reasons. While the corresponding real time may vary, 60 transitions are well contained within a single trade day in our data set. For much longer time-horizons, we expect a curse of horizon, i.e. [117]. Combining different downsampling discretizations of the dataset can potentially alleviate this problem. This investigation is beyond the scope of this chapter.

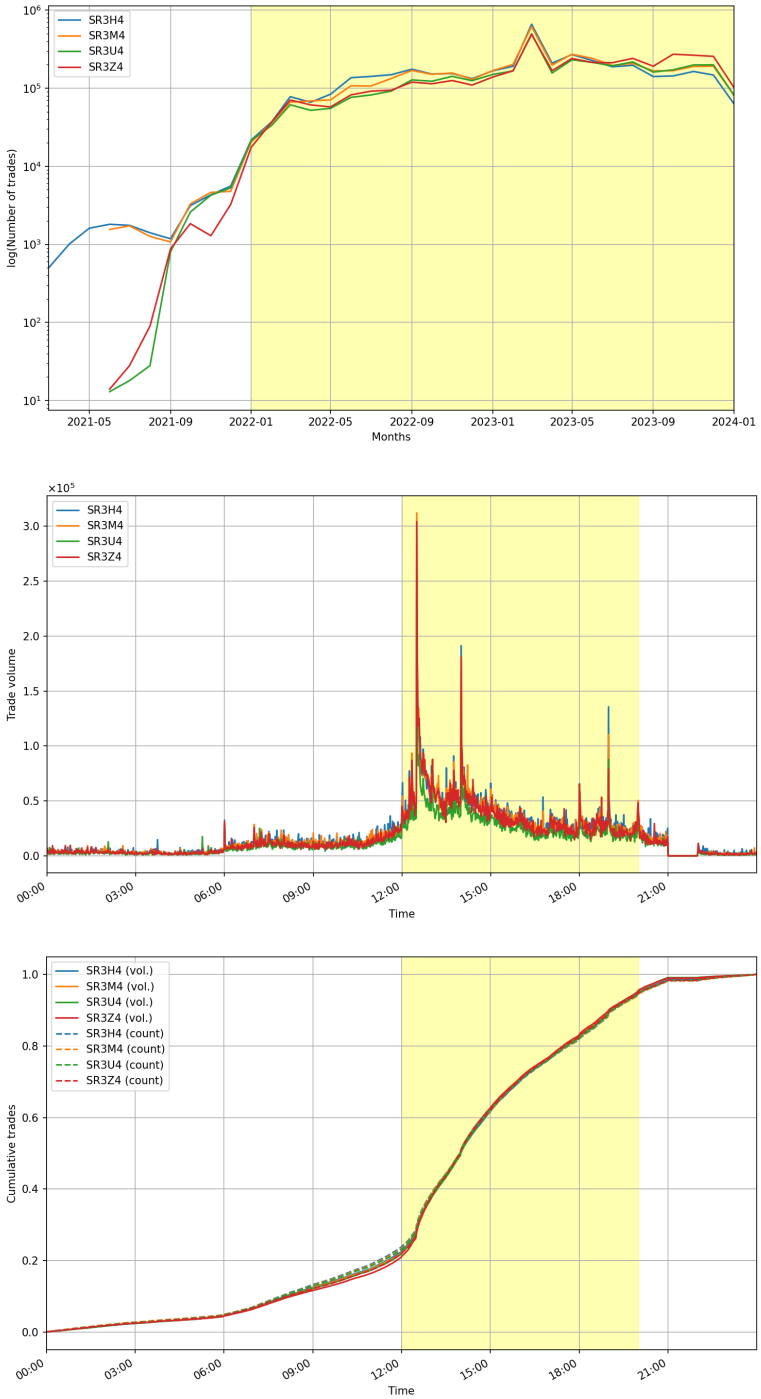


Figure 4.4: Monthly trade count for different active contracts in data collection (top); Minute-by-minute trading volume on 02/11/2023 (middle); Cumulative trading volume and count normalized by the total daily trade volume and the total daily trade count on 02/11/2023 (bottom)

4.5.1 Resampling statistics

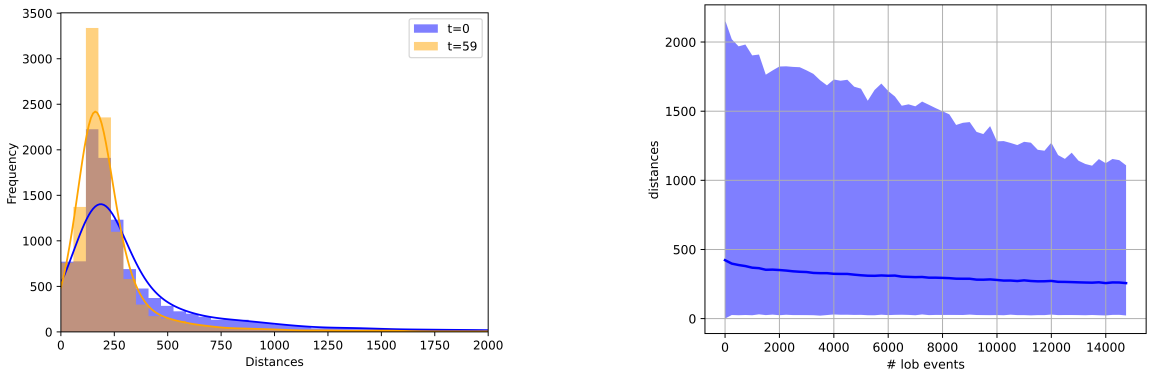


Figure 4.5: Resampling statistics: Distances distribution (left); distances dynamics (right)

We start by reporting several observations on Algorithm KNNR-LOB. The first plot in Figure 4.5 shows the distribution of distances between matched states at the beginning of the resampling path and at the end, i.e.

$$\|\widetilde{S^5_{p_s}}(s) - \overline{S^5_{p^*_{t_j}}}(t_j)\|$$

where $\|\cdot\|$ is the Euclidean norm and $s = 0$ or $s = 59$. In general, smaller distances in a nearest neighbor matching are associated with a smaller bias in the quantity of interest. We observe that the distributions at both points in time are heavily skewed to the left with outliers to the right. Comparing the distances at the initial and the final matching reveals that the distances at the beginning are larger on average and the right tail is less heavy compared to the last transition. The second plot in Figure 4.5 confirms this observation. Here, we plot the average distance and its 0.95-quantile against time. We observe that both metrics decrease over time. This can potentially be explained by the uniformly random initialization of the state as edge cases are more likely to occur in a uniform random framework compared to a nearest neighbor choice where centrality is preferred. This effect could be mitigated by a firmer restriction on less liquid market regimes or an adapted initial sampling scheme. On the other hand, the decreasing distance over time also implyies a reduced bias contribution of later transitions.

4.5.2 Single transitions

As a next step we compare several properties of a single LOB transition from an actual historic transition to a transition induced by Algorithm KNNR-LOB. For all experiments, we use the same initial states for the historical roll-outs and the matched nearest neighbors transitions. First, we consider the real and resampled marginal volume distribution at different order book levels. For this, we will use a slightly different formulation for LOB snapshots introduced in [48], i.e., we will fix the dividing price p_0^* from before transitioning and center the LOB after the transition at that price. We use signed volumes that are negative for bid order volumes and positive for ask order volumes: for a bid price level $\lfloor p_0^* \rfloor - k\delta$ with $k \in \mathbb{N}_{>0}$ and $t = 1$, we will consider the volume $V_{\lfloor p_0^* \rfloor - k\delta}(t)$ with a negative sign if $\lfloor p_0^* \rfloor - k\delta$ is on the bid side at time t, and with a positive sign else. We denote the signed volume by $\tilde{V}_{\lfloor p_0^* \rfloor - k\delta}(t)$. The definition is analogous for ask prices. This formulation allows us to compare the volume at a given price level and include information on order book movements, i.e., if we consider a bid price $\lfloor p_0^* \rfloor - k\delta$ for positive $\tilde{V}_{\lfloor p_0^* \rfloor - k\delta}(t)$ we can infer that the market moved downwards and $\lfloor p_0^* \rfloor - k\delta$ is an ask price at $t = 1$. Furthermore, we normalize the volume V with the square root transformation $\text{sign}(V)\sqrt{|V|}/100$ as in [48] to control the skewness of the volume distribution.

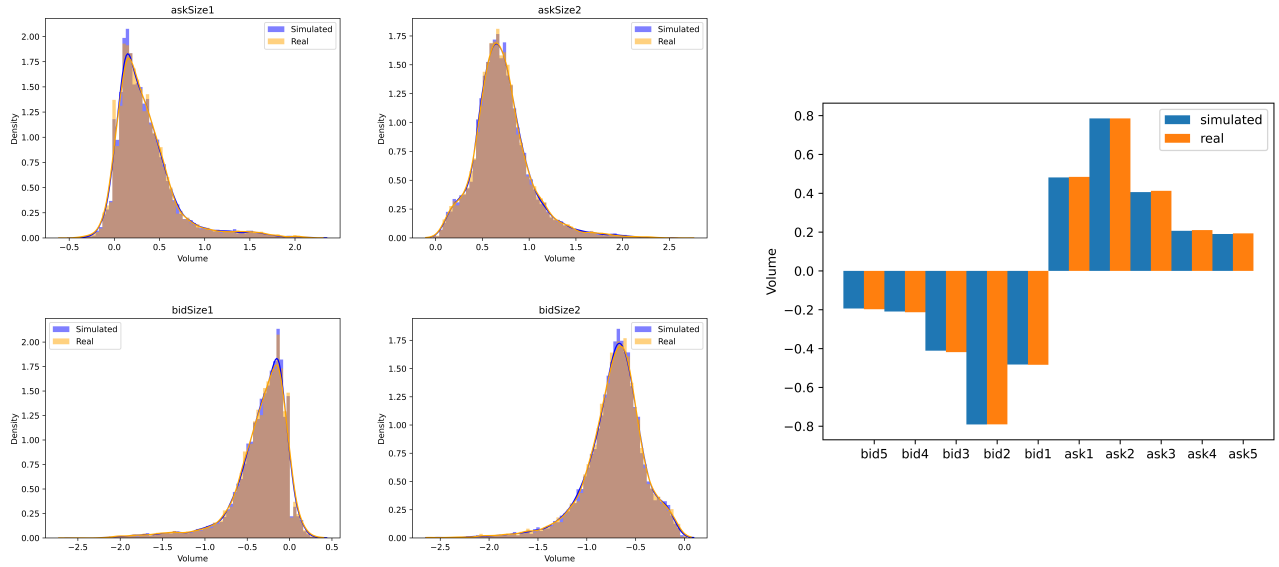


Figure 4.6: Volume comparison: Marginal volumes per order book level after one transition (left); Average LOB shape after one transition (right)

On the left in Figure 4.6, we give the distribution of signed volumes after one transition for the real market and the simulated market. We report the results for the two best bid and ask price levels from before transitioning, i.e. $k = 0, 1$. Overall, the simulated market provides a very good fit to the real market. Furthermore, we observe that for the best price levels (askSize1 and bidSize1) the volumes tend to be smaller than for the second best price levels (askSize2 and bidSize2). If the volume's sign at the best price level changes, the newly established volume tends to be small on the opposing order book side and the probability of a sign change is relatively small. For second best price levels there is almost no mass on the opposite sign implying that market moves of two ticks are highly unlikely.

On the right in Figure 4.6, we give the average (normalized) volume after a transition for the real and the simulated LOB on different LOB levels. We report bid volumes as being negative signed and ask volumes as positive. The real and simulated average volumes are almost indistinguishable. Very small differences can be observed on the LOB levels furthest away from the spread on both sides where the simulated market has slightly less volume compared to the real market. Comparing this result to a benchmark which unconditionally and uniform randomly selects a transition from the training data (cf. Figure B.1) reveals that Algorithm KNNR-LOB is better in fitting the average volumes for the samples from the test data. The discrepancy between the benchmark and the averages from the test data points to a slight distributional shift between the training and test split of the data. This distributional shift seems to be captured by Algorithm KNNR-LOB and shows that our method goes beyond an unconditional law of large numbers. We further discuss this comparison in Appendix B.1.

Figure 4.7 compares the correlation structure of LOB volumes. The first row gives the correlation between different order book level volumes after the resampled transitions and after the real transitions. The simulated correlation structure and the real structure correspond very well. Together with the results on the marginals (Figure 4.6), this points to an overall good fit for the joint distribution of volumes generated by a resampling transition. In the second row of Figure 4.7, we consider the correlation of changes in volume for a given order book level over one transition. For central order book levels (bid and ask 1-3), the correlation of the difference coincides well for the real and the simulated transition. Thus, also the interdependence in the change of volume is

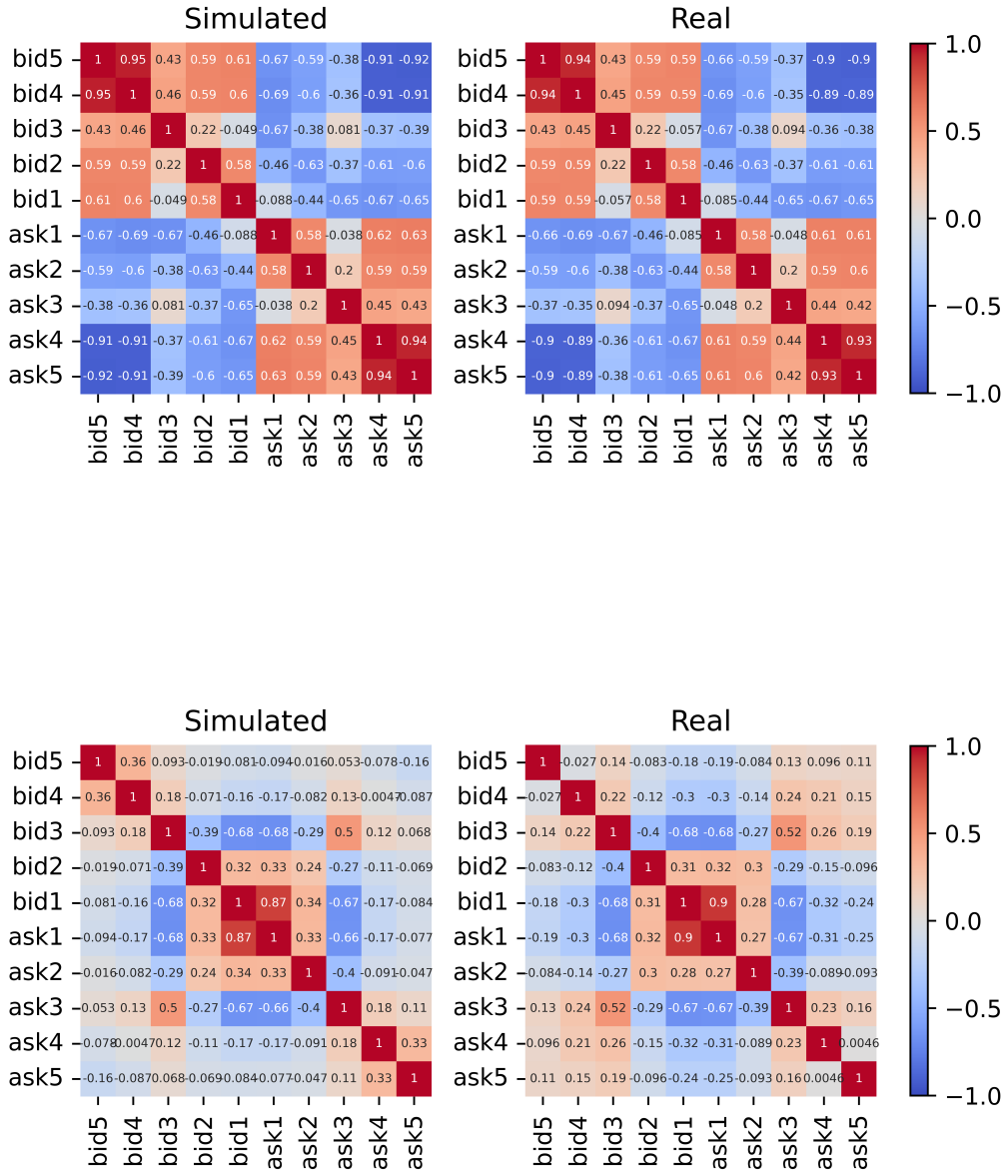


Figure 4.7: Correlation: Absolute volumes (first row); Volume differences (second row)

appropriately captured for the central order book levels. For the outer order book levels, we observe mismatches. This may be related to the observation that volume on these levels are often relatively time persistent and do

not tend to change much. As these levels are often less important for price formation, one could remove them from the order book matching depending on the simulation objective.

4.5.3 Dynamic comparison

We turn to evaluating the simulation performance of Algorithm KNNR-LOB over multiple time steps. First, we

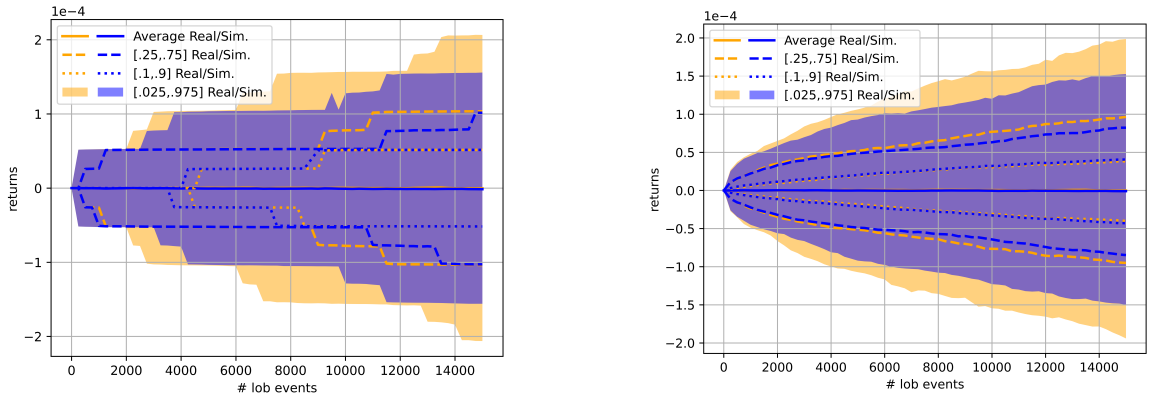


Figure 4.8: Comparison of real return time series with simulations by Algorithm KNNR-LOB: Mid-price (left); Weighted mid-price (right)

compare the unconditional return distribution of the resampled LOB dynamics and the real market returns. For this purpose, we choose 10^4 real LOB paths with 60 transitions each and calculate the corresponding return paths. Reusing the initial states from the real paths as starting points, we simulate 10^4 LOB paths using Algorithm KNNR-LOB with the same length and calculate the returns along them. Figure 4.8 plots the average return (line) and several of its quantiles for the real and simulated market. On the left, the mid-price returns are compared and one can see that the averages almost coincide. The quantiles mostly match as well, however one can see a slight tendency that the real market reaches more extreme levels of return slightly faster compared to the simulation. The weighted returns, shown on the right in Figure 4.8, do not only reflect tick-sized changes in the price but also represent shifts in the positioning on the highest levels of the LOB. This is especially relevant in pro-rata order book markets as the execution mechanism requires us to have a realistic view on the volumes as well. The averages of the real and simulated return almost coincide again. The return quantiles have a similar coverage. However, the quantiles of the real return tend to be slightly larger compared to the simulated return quantiles.

To evaluate the effectiveness of simulating the conditionality inherent in order book transitions, we evaluate the predictive power of the initial state on the simulated return. For that purpose, consider an initial LOB snapshot $S^5_{p^*_0}(0) \in \mathcal{D}$, and let $R^r_{k\Delta t}(S^5_{p^*_0}(0))$ denote the real return $k\Delta t$-steps in the future starting at $S^5_{p^*_0}(0)$ and let $R^s_{k\Delta t}(S^5_{p^*_0}(0))$ denote the simulated return $k\Delta t$-steps in the future starting at $S^5_{p^*_0}(0)$. If the choice of the initial state $S^5_{p^*_0}(0)$ has a similar impact on the simulated return $R^s_{k\Delta t}(S^5_{p^*_0}(0))$ as the real return $R^r_{k\Delta t}(S^5_{p^*_0}(0))$ one expects them to be positively correlated, i.e. the simulated return is a predictor for the real return. However, this correlation should not account for newly incorporated noise within the dynamical system to allow the simulator to be effective. Further, the impact of the initial state should decrease over time assuming ergodicity and thus also the correlation between the returns.

In Figure 4.9, we estimate the correlation of real and simulated returns starting from the same initial states over time. The left plot compares the correlation of the mid-price returns and the right plot the correlation of the weighted mid-price returns. The shaded area in the plots corresponds to the 95%-confidence intervals and

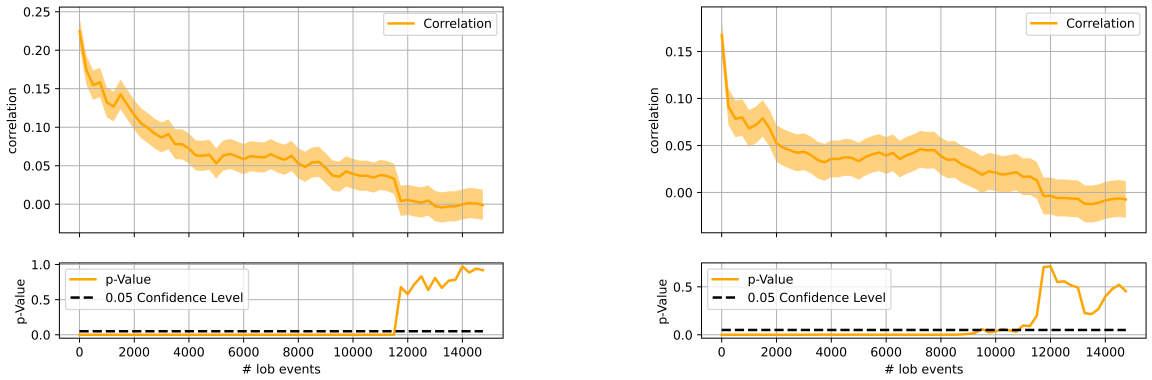


Figure 4.9: Return correlation: Mid-price (left); Weighted mid-price (right)

the smaller plots in the second row are the corresponding p-values for testing that the correlation is not equal to zero. Both plots show the expected behaviour for the correlation. In particular, the correlation between real and simulated returns is significantly larger than zero over a large number of LOB events and is decreasing over time. The weighted mid-price return correlation is not as strongly correlated compared to unweighted return correlation. However, both correlation estimates lose significance at a similar time after around 1.1×10^4 LOB events.

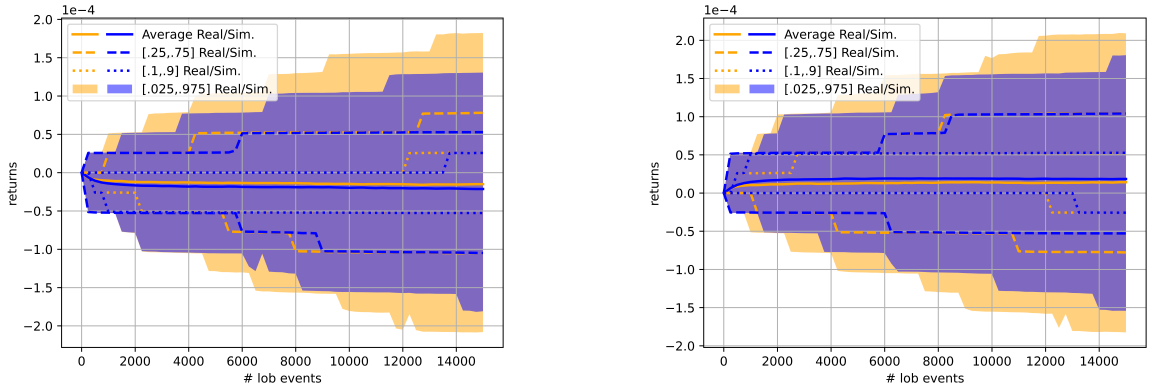


Figure 4.10: Comparison of real return time series with simulations by Algorithm KNNR-LOB with large order book imbalances in initial state: Mid-price (left); Weighted mid-price (right)

To enhance the point that Algorithm KNNR-LOB captures conditionality, we consider an additional experiment where the initial states show strong order book imbalances. These are known to be predictors for price movements, e.g. [37]. A large negative value signifies a much lower volume on the bid compared to the ask. Hence, the best bid level is more likely to be depleted, giving a larger probability for downward move in the price. For larger positive values of the order book imbalance the opposite is implied. In Figure 4.10, we randomly choose initial states for which the order book imbalance is within the 0-0.05 quantile (left) or the 0.95-1 quantile (right). With these initial states, we compare the real mid-price return time-series with the simulated ones (cf. Figure 4.8). As expected, the distribution of the real returns exhibits the impact of the large scale order book imbalances. For large negative values of the order book imbalance at the initial state, the mean and the quantiles shift downward (left) at the beginning of the time horizon and then remain at this new level.

For large positive values (right), we see an analogous upward shift. The simulated mid-price returns exhibit for both cases the same behavior where the distributional attributes closely match the real returns. This allows the conclusion that Algorithm KNNR-LOB is capable of capturing the conditional dependence in the order book transition on the current order book imbalance.

4.5.4 Benchmark comparison

Feature Name	KNNR-LOB	CGAN	naive
bidSize2	**.024 ± .0062**	.034 ± .0089	.049 ± .0160
bidSize1	**.024 ± .0053**	.040 ± .0078	.046 ± .0110
askSize1	**.029 ± .0044**	.043 ± .0098	.058 ± .0141
askSize2	**.027 ± .0070**	.029 ± .0053	.054 ± .0158
OBI ($s = 1$)	**.033 ± .0062**	.035 ± .0095	.038 ± .0126
OBI (s=10)	.040 ± .0100	.062 ± .0140	**.034 ± .0069**
OBI ($s = 30$)	.045 ± .0132	.058 ± .0153	**.042 ± .0155**
OBI ($s = 60$)	**.038 ± .0082**	.077 ± .0162	.042 ± .0052
mid-price return ($s = 1$)	**.020 ± .0054**	.023 ± .0060	.048 ± .0111
mid-price return ($s = 10$)	**.040 ± .0091**	.047 ± .0126	.154 ± .0165
mid-price return ($s = 30$)	**.041 ± .0063**	.051 ± .0071	.171 ± .0173
mid-price return ($s = 60$)	**.053 ± .0121**	.065 ± .0138	.184 ± .0094
weighted return ($s = 1$)	**.075 ± .0140**	.080 ± .0123	.258 ± 0.0183
weighted return ($s = 10$)	**.066 ± .0115**	.091 ± .0183	.203 ± 0.0140
weighted return ($s = 30$)	**.056 ± .0137**	.010 ± .0218	.196 ± 0.0166
weighted return ($s = 60$)	**.059 ± .0164**	.106 ± .0201	.193 ± 0.0067

Table 4.1: Mean and standard deviation for the KS test statistic for Algorithm KNNR-LOB and benchmarks.

The previous analysis of the simulation results for Algorithm KNNR-LOB focused on a comparison with real markets. To benchmark our method against other LOB simulation techniques, we follow [106] and compute the two-sample Kolmogorov Smirnov (KS) test statistic for the empirical distribution of the simulation with the real empirical distribution for certain features of interest. The two-sample Kolmogorov Smirnov (KS) test statistic is given by

$$\max_x |F_1(x) - F_2(x)|,$$

where F_1 and F_2 are one-dimensional empirical cumulative distribution functions of the two samples. Thus, a smaller value suggests a better fit of the distributions, with all values in the interval $[0, 1]$.

We evaluate the KS statistic for several different features, namely, we consider the marginal distributions for order book sizes for one transition (as in Figure 4.6), the weighted mid-price and the mid-price return distribution (cf. Figure 4.8) and the distribution of order book imbalances (OBI)[12] at different times s. To assess the realized values of the test statistic, we compare Algorithm KNNR-LOB to benchmark models, specifically an out-of-the-box implementation of the LOB generator (CGAN) suggested in [48] and a simple randomized replay of LOB transitions (naive). Implementation details and the corresponding comparison of the benchmarks with the real market can be found in Appendix B.1. Similar to [106], for reporting, we calculate the KS statistic for 1000 samples of the simulation and compare them to 1000 samples from the real distribution. We repeat this procedure 10 times and report the mean and the standard deviation for the KS statistic.

[12]For the definition of the order book imbalance, see Section 4.2.

Table 4.1 shows that Algorithm KNNR-LOB consistently outperforms our benchmarks for most variables. In particular, we highlight that all features related to returns outperform both the CGAN and the naive benchmark. We note that features related to returns are of particular interest as they capture the conditional dynamics of the order book by the evolution of prices in contrast to the volume-based measures. For these volume-based features, an unconditional central limiting behaviour is potentially sufficient as evidenced by the strong performance of the naive benchmark for order book imbalances after multiple transitions. Furthermore, we observe that the CGAN benchmark outperforms the naive benchmark for the majority of features and, in particular, on all return features.

4.5.5 Market order impact

Next, we investigate the impact of market orders by a trading agent on the market simulation. In general, markets are expected to respond to market orders of sufficient size in an adverse way. For example, if a trader places a large sell order, the market price will tend to decrease. The impact of market orders is a widely researched topic, e.g. [220], as it can have a significant negative effect on profits from trading. It is suggested e.g. in [67, 80, 208] that the market impact is proportional to the square root of the order size P divided by the daily trading volume $\overline{V}$,

$$\text{price impact} \propto \sqrt{P/\overline{V}}.$$

Other empirical findings, e.g. [10, 151], more generally suggest that the impact of order size follows a concave power law,

$$\text{price impact} \propto \left(P/\overline{V}\right)^{\gamma}.$$

with $\gamma \in [0, 1]$, where the exact choice of γ depends on the market.

We will study the scenario where a trader seeks to liquidate a long positions of varying sizes using market orders. To do this, the trader will split a large parent order in smaller child orders and disperse them over time. If P is the size of the parent order and $\widetilde{t}_n$ the maximum trading time, then a single child order will be of size $P/\widetilde{t}_n$ for LOB snapshot times $s \leq \widetilde{t}_n$. After time $\widetilde{t}_n$, no orders will be executed. In our experiments, we trade over half of the entire trading time horizon, i.e. $\widetilde{t}_n = 30$. In total, we will sample 5000 paths per order size[13].

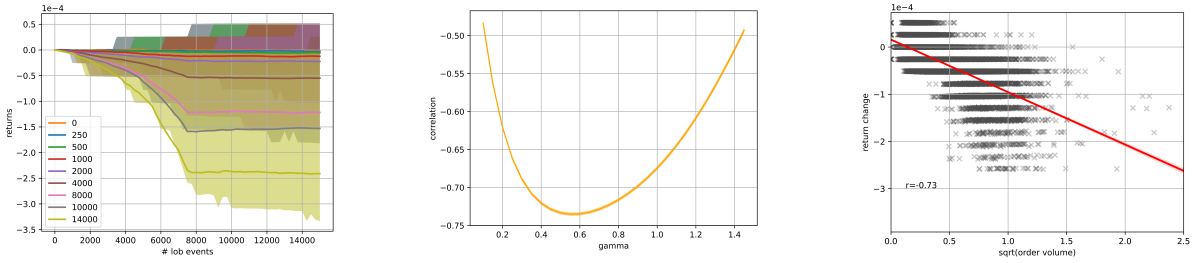


Figure 4.11: Impact of market orders: Mid-price returns for different parent order sizes (left); Explanatory power for returns depending on functional shape (middle); $\sqrt{\cdot}$-order volume return fit (right)

The first plot in Figure 4.11 shows the evolution of the average mid-price market returns for different parent order sizes over time. The shaded areas reflect the central quartiles (0.25-0.75). One can immediately see the adverse effect of market sell orders on the price as the average return decreases over time up until trading stops, after which the prices seem to remain constant. This corresponds to the simulation results for market orders in [46, 48]. The average price impacts are ordered by the size of market orders and the return ranges increase for increasing market orders. Especially, for very large orders, the average return is at places smaller than the 0.25-quantile, suggesting that outliers impact the mean paths.

[13]For all succeeding experiments, we will use $N = 5000$ unless mentioned otherwise.

We note that regimes with extremely large market orders of the size tested in our experiments are very rare in our data set and, thus, coverage under resampling is not as good as in ordinary market regimes. This leads to poor tail behaviour for the resampled scenarios. Thus, we will focus on the central quartiles of market responses to exclude outliers in the following analysis. To ensure comparability, we do this across all position sizes. A possible alternative could be the implementation of a mixture model that combines our non-parametric approach with a parametric model for regions with insufficient coverage similar to [76].

Next, we investigate the shape of the market impact. For this purpose, we calculate the correlation between the mid-price return at time $\widetilde{t}_n$ and the power of the weighted parent order size $\left(P/\overline{V}\right)^{\gamma}$ for different values of γ. As a proxy for the daily volume $\overline{V}$, we use the volume of the entire order book at the initial starting point. This is necessary since our data set contains trading regimes with varying levels of liquidity due to the seasonality of trading in particular futures contracts. The second graph in Figure 4.11 plots this correlation against different values of γ. The resulting image is roughly parabola-shaped with a minimum value of 0.74 at around $\gamma = 0.55$. This suggests that the weighted parent order size has the most explanatory power for the mid-price return if $\gamma = 0.55$. The varying thickness of the line represents the 99%-confidence interval for this estimate. Hence, the correlation value at $\gamma = 0.55$ is significantly smaller than at $\gamma = 1$, giving strong evidence for a concave function shape. To show the fit of the classical $\sqrt{\cdot}$-law, we regress $\left(P/\overline{V}\right)^{0.5}$ against the mid-price returns and plot the fit in the right-most graph in Figure 4.11. The varying thickness of the line indicates the 95%-confidence interval for the estimate. The plot reveals a good fit to a square-root model with nearly optimal explanatory power out of the power-law model class. One should note, however, that the confidence bands reveal a slight heteroscedastic error with increased errors for large values of weighted order volumes.

4.5.6 Limit order impact

Similar to market orders, we expect an adversarial effect of large limit orders on the price, i.e. [19, 55, 97, 183].

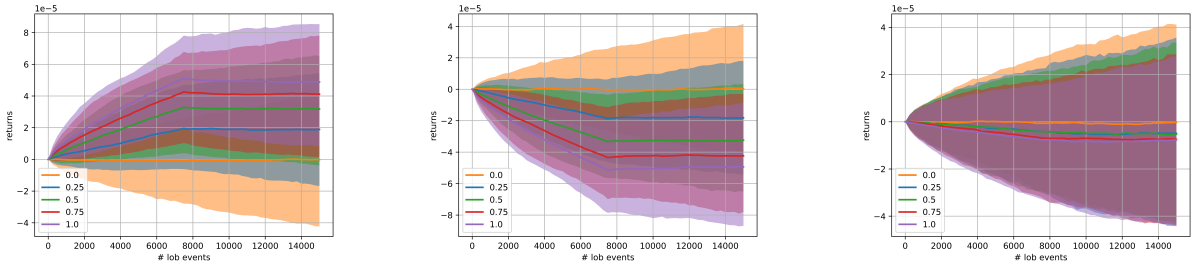


Figure 4.12: Impact of limit orders with different parent order sizes: Best bid (left); Best ask (middle); Second best ask (right)

For the experiments, we assume that a trader continuously places limit orders over time at particular levels of the centered LOB. Similarly, as for market orders, we fix the size of the parent order P and a maximum trading time $\widetilde{t}_n$, such as a single child order corresponds to order size $P/\widetilde{t}_n$ for LOB snapshot times $s \leq \widetilde{t}_n$. After time $\widetilde{t}_n$, no limit orders will be placed, and we choose $\widetilde{t}_n = 30$. Instead of using mid-price returns, we will consider weighted mid-prices to reflect repositioning on the highest order book level. Furthermore, we will use sizes relative to the overall order book volume at initialization for parent orders to have a comparable distortion for the weighted returns.

We exhibit our results in Figure 4.12. The plots show for different order book levels the weighted return paths for different parent order sizes and the shaded areas correspond to the central quartiles (0.25-0.75). For the case where the orders are placed on the best bid and ask level, one can directly see the adverse effect of the limit orders on the weighted return. Namely, for large ask orders, the weighted return decreases while for large bid orders it increases. The average return impacts are ordered by the size of the orders. Once trading stops,

the return levels stabilize at the new price level (cf. [48]). For limit orders placed on the second best ask level, the impact is not perfectly ordered by order size any more and is smaller in scale compared to order placement at the best level. This is consistent with the observation that the SOFR futures market only tends to move by single tick increments (cf. Figure 4.6) as the relatively small variability allows traders to reposition themselves once a new price level is reached.

4.6 Policy evaluation

In this section, we show how Algorithm KNNR-LOB can be used to evaluate and choose trading strategies. A central theme for placing limit orders in markets with a pro-rata type execution mechanism is choosing the right quoting amount. In particular, it is well known that market participants tend to place limit orders with a volume larger than they would like to execute (overquoting), e.g. [59, 83]. Market participants overquote because it increases their relative volume share on a given order book level and with it the probability of being allocated an incoming market order. The trade-off present with overquoting is the possibility that too many trades are executed against the volume posted by the trader. Balancing this trade-off via the quoting volume is key for execution and market making strategies in markets with narrow spreads and high trading volumes such as the SOFR futures market.

First, we want to show that the market simulations generated by Algorithm KNNR-LOB capture the necessity of overquoting for achieving per-transition trade targets and that the resulting market shows favourable characteristics for overquoting. For this purpose, we introduce a trading agent to Algorithm KNNR-LOB who keeps a constant position on the best bid level in the order book over time. If the market price moves, the agent cancels their remaining volume at the old level and places a limit order at the new best bid level. If the limit order is (partially) executed, the trading agent replenishes their order to the prescribed order size.

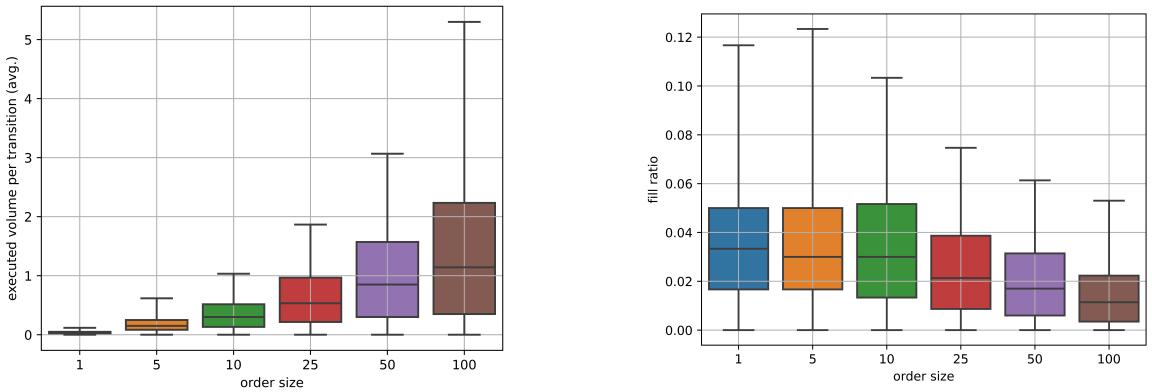


Figure 4.13: Constant order sizes: Time-averaged executed volumes per transition (left); Fill ratios (right)

In Figure 4.13, we report the results for this experiment. In the left-hand plot, we use box plots to report the distributions of executed volumes normalized by the number of transitions for different order sizes. The whiskers highlight the 0.025 and the 0.975 quantile, the box represent the first and third quartile and the median is given by the line contained in the box[14]. First, one can see that all reported quantiles, except the 0.025 one, increase for an increasing order size. If the trading agent only places an order with volume 1 each period the executed volume per transition is close to zero. On the other hand, if order sizes are between 50 and 100 contracts, the median executed volume per transition is around one. However, the upper quartile and the 0.975-quantile can be significantly higher. To achieve a time-averaged executed volume per transition of one contract, a trader would

[14]We use this convention for all box plots in this chapter.

need to increase their order size substantially above one, which brings the risk of overshooting the target and having to trade more volume than desired. Note that this behavior is consistent with the empirical findings and the theoretical model described in [59] and that the exact choice of order size would depend on the trader's risk profile. The box plot on the left in Figure 4.13 shows the ratio between filled orders and total posted volume over time (fill ratio). Overall, the fill ratios and their variation tend to decrease for increasing order sizes. This behavior of the market further encourages overquoting as it counteracts the risk of overfilling.

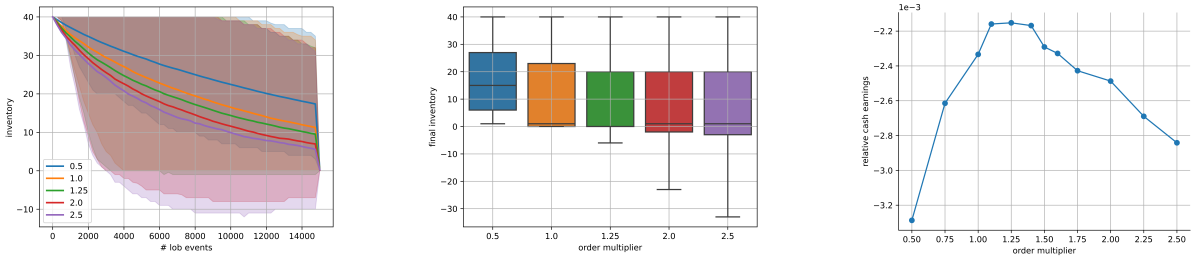


Figure 4.14: Trading strategy calibration: Inventory paths for varying order multipliers k (left); Final inventory distribution for varying order multipliers k (middle); Relative cash earnings for varying order multipliers k (right)

In our second experiment, we demonstrate how Algorithm KNNR-LOB can be used to calibrate a trading strategy. For this purpose, we consider a trader who seeks to unwind their long position using limit orders over a certain time horizon[15]. We consider a simple execution strategy whereby the trader quotes a multiple of the remaining inventory on the best ask level, i.e. the quoted volume on the best ask will be kI_s, with I_s the inventory after the s-th transition and k is the order multiplier. Should the inventory become negative, meaning the trader has a short position, all remaining volume on the best ask will be cancelled. Instead, the trader will place an order of size $k|I_s|$ on the best bid level. All remaining inventory at the last transition will be executed via a market order. To calibrate the trading strategy, we run Algorithm KNNR-LOB for different values of k ranging from 0.5 to 2.5 with the starting inventory $I_0 = 40$.

The results of this experiment are reported in Figure 4.14. The left plot shows mean inventory paths (lines) and the 90%-quantile intervals for inventories (shaded areas) for different values of the order multiplier. The mean inventory paths and quantile intervals are decreasing and are ordered by the value of the order multiplier. The lower quantile values decrease especially fast for the medium and larger order multipliers, hinting that a single trade is potentially enough to execute all the inventory. Further note that for values where the order multiplier is larger than one, the trader may overshoot their trading target and may end up with a short position that then needs to be unwinded during the remaining trading period. The upper quantile limit on the other hand remains close to 40 over a longer period of time for all order multipliers. This implies that there is a possibility that no trade is executed against our trader's limit orders. The upper quantiles then start decreasing after some time for all k. For larger k, this decrease starts earlier, signifying that larger orders are more likely to be (partially) executed in this pro-rata type market.

The middle plot in Figure 4.14 shows the distribution of inventories at the last time step before the market orders are executed for different order multipliers. For $k = 0.5$, it is infeasible that the entire inventory is executed via limit orders and the median is relatively large. This means that a large proportion of inventory will be executed via market orders. If the order multiplier is equal to one, the trader cannot overshoot the execution target with limit orders. The median final inventory is close to zero, but, compared to larger k, the upper quartile is larger, implying a higher reliance on market orders. For $k = 1.25$, the lower quartile and the median coincide at a zero inventory and the trade target is overshot only in the tails. Furthermore, the upper quartile is lower compared to $k = 1.0$, implying less reliance on market sell orders. For large values of k, we

[15]Note that the first experiment and [59] differ in context to this second experiment. Instead of focussing on a one shot/per-transition trade target, this experiment takes into account the time dependency of the execution problem limiting its comparability.

observe that the trade target is consistently overshot, which will require market buy orders to unwind short positions. On the other hand, the upper quartile remains almost unchanged compared to $k = 1.25$. Finally, note that the upper whisker is for all order multipliers equal to the initial inventory. This points to the possibility that no order is filled via limit orders for all multipliers.

Since executing market orders requires paying the spread, we expect that order multipliers requiring large market orders in the final period will exhibit a higher cost of execution. To investigate this hypothesis, we compare the average earnings per asset for different values of the order multiplier. We quantify the earnings by dividing the total earned cash by the initial inventory and then subtracting the initial price. The corresponding plot can be found in Figure 4.14 on the right. First, note that for all order multipliers the relative average cash earnings are negative, implying a cost of execution on average. The scale of the execution costs varies around half a tick size. Highest earnings (or lowest execution costs) are reached at $k = 1.25$, while they decrease consistently for smaller and larger values in k. Considering the box plot on final inventories and its discussion, this observation is consistent with the hypothesis that order multipliers requiring larger market orders will perform worse. In particular, for $k = 1.25$, market sell orders are less often needed compared to smaller k and this benefit seems to outweigh the occasional costs associated to overshooting. On the other hand, for larger k, the additional costs of overshooting outweigh any potential benefit of additional unit execution. From the view point of strategy calibration, this experiment implies that a risk neutral trader should choose $k = 1.25$ to minimize expected execution costs.

4.7 State space extension

In this section, we provide a strategy on how to extend Algorithm KNNR-LOB to a more complex state space. In particular, we expand the state space by multiple features beyond LOB snapshots. Adding these additional features will increase the dimensionality beyond the level usually seen as appropriate for nearest neighbor search due to the curse of dimensionality; see e.g. [87]. We show how this issue can be overcome with simple dimension reduction techniques and how this can lead to beneficial results.

In addition to the current volume levels in the order book, we include features containing information on recent price moves and trading activity in the LOB state. Namely, we add mid-price returns and trade imbalances[16] over different time windows for each sample in the data set. This is done by slightly modifying the data set generation. In addition to the procedure described in Section 4.4, we look up for each LOB snapshot the mid-price that was observed a certain number of LOB events before the snapshot was taken. With the previous mid-price and the mid-price of the current snapshot we calculate the mid-price return for each sample. We do this for a differing number of preceding order book events. In this experiment, we choose the number of order book events for mid-price returns by 250, 1250, and 5000 order book events. This gives us 3 additional matching dimensions in the nearest neighbor search. Similarly, we add the trade imbalances where the feature is calculated for the previous 250, 2500, and 12500 order book events for each snapshot. In total, we then have a 16-dimensional state space. For normalization, we take the square root of the order book volumes and then apply a z-score transformation to all variables. With the normalized features, we calculate the weight vector for a principal component analysis (PCA) and reduce the dimension of the data to the first eight principal components, which explain around 80% of the data's variability.

For comparability, we conduct the same benchmark analysis based on KS statistics as in Section 4.5.4 and report the results in Table 4.2. We highlight the variables that improved in comparison to Table 4.1, and we see that Algorithm KNNR-LOB with an extended state space improves on the majority of tested features. Note that we investigated only one particular combination of features and dimension reduction. Other combinations may be more effective and their choice may depend on the task at hand. We leave the exploration of more principled approaches for LOB representations in the context of order book simulations for future work.

[16]The trade imbalance is given as the volume of all buyer-initiated trades divided by the volume of all trades over a given period of time, see e.g. [46].

Feature Name	KNNR-LOB (ext.)	Feature Name	KNNR-LOB (ext.)
bidSize2	.027 ± .0060	mid-price return ($s = 1$)	**.015 ± .0027**
bidSize1	.026 ± .0058	mid-price return ($s = 10$)	**.023 ± .0054**
askSize1	**.025 ± .0044**	mid-price return ($s = 30$)	**.036 ± .0075**
askSize2	.028 ± .0079	mid-price return ($s = 60$)	.054 ± .0100
OBI ($s = 1$)	**.030 ± .0061**	weighted return ($s = 1$)	.084 ± .0128
OBI ($s = 10$)	**.034 ± .0079**	weighted return ($s = 10$)	**.057 ± .0137**
OBI ($s = 30$)	**.040 ± .0100**	weighted return ($s = 30$)	.059 ± .0167
OBI ($s = 60$)	.043 ± .0108	weighted return ($s = 60$)	**.055 ± .0071**

Table 4.2: Mean and standard deviation for the KS test statistic for the extended KNNR-LOB Algorithm.

4.8 Further research directions

The results presented in this chapter point to multiple directions for possible future research. First, the effectiveness of K-NN resampling for LOB simulations suggests that the algorithm should be considered for simulating and evaluating strategies in other stochastic control environments where direct testing is costly and historical data available. Other applications in finance such as hedging or portfolio management with market impact could be considered. On the algorithm, we believe that further investigations in how to include dimension reduction and the combination with other generative models can be fruitful. In particular, we would be interested in how to choose the state space for the algorithm optimally such that most of the relevant information is contained while keeping its dimension controlled. A systematic investigation on the choice of metric for the nearest neighbor search for LOBs would equally be of interest. Finally, in the context of trade evaluation in LOBs, we believe that several extensions could be of interest. Combining the algorithm with a simulator for FIFO executions would allow trade evaluation in major asset classes such as equities. For pro-rata type markets, more complex trading strategies could be considered, i.e. liquidation with limit and market orders or the inclusion of alpha signals.

Chapter 5

Policy Gradient Methods for Limit Order Books

5.1 Introduction

As we discussed in Chapter 2, policy gradient methods recently gained a considerable attention due to there effectiveness for solving complex stochastic control problems, e.g. [33, 93] and advances in their theoretical understanding, e.g. [88]. In this chapter, we discuss the usage of policy gradient methods for optimizing execution strategies in a limit order book (LOB) using only limit orders. A main challenge in this context is to consistently estimate gradients with respect to the strategy parameters.

As discussed in Section 2.4.2, one may consider zeroth-order or path-wise gradient estimators for a policy gradient algorithm. First, for zeroth-order gradient estimation, the main challenge is the inherent stochasticity of the LOB dynamics. This stochasticity leads to additional noise in the gradient estimator. For a simple implementation of the zeroth-order gradient estimator, the noise may outweigh the informative content for a reasonable number of samples (cf. Section 5.2.4). Second, path-wise estimation of gradients such as in [33, 93] may not be possible even in simple parametric models of an LOB. For example, in the LOB model described in [39] (see Section 5.2.1), the strategy parameter changes the distribution of the probability of the limit order being matched with an incoming market order. However, for a single realization of a path, it is unclear how a marginal change in the trade-matching distribution will impact the outcome. More complex LOB models, that allow trade evaluation for limit orders, may require algorithmic choices that lead to problems for path-wise gradient estimation. For example, in the nearest neighbour resampling applied to LOBs (see Chapter 4), the nearest neighbour search may not be sensitive enough to marginal changes in the strategy. Note that we are currently not aware of any LOB model allowing explicitly for path-wise gradients with respect to trading strategies for limit orders.

We present two applications of policy gradient methods for trade execution in LOBs where we overcome these difficulties. First, in a parametric model for LOB trading from [39], we use a policy gradient algorithm with a zeroth-order approximation to learn effective execution strategies. In particular, we introduce several numerical modifications to standard policy gradient methods to reduce the variance in the gradient estimator and achieve convergence. Second, to overcome issues with lacking realism of parametric models for LOBs, we take inspiration from previous chapters and introduce a simulator trained on actual LOB data. In particular, we introduce a GAN LOB model for a pro-rata market which is differentiable with respect to the strategy parameters along paths and allows the evaluation of limit orders. With this simulator, we are capable of using path-wise gradient estimation for the policy gradient method where the number of samples used for estimation is only restricted by the computational cost of simulation. Making additional numerical adjustments, we introduce a policy

gradient method that learns effective trading strategies allowing the investigation of an otherwise intractable asset execution problem.

5.2 Policy gradients for a parametric LOB model

This section presents some results on applying policy gradient methods in the context of an asset execution problem with a parametric model for the limit order book using only limit orders. These considerations are of interest since they are, to our knowledge, the first application of policy gradients in such a context. Furthermore, they highlight certain numerical/algorithmic challenges and possible solutions to overcome these. The execution problem that we are considering has the following characteristics: We have an initial inventory of assets that need to be liquidated over a fixed time horizon and at all times we can place them as limit orders in a LOB at a continuous depth. The depth the order was placed determines the probability of the order being filled by randomly arriving market orders. Our goal is to place the limit order as deep in the order book as possible so as to achieve the best price while making sure that the entire inventory is liquidated (or receive a penalty for having to execute with a market order). This problem offers a non-standard control mechanism that requires several interesting adjustments for the policy gradient method to work.

The model we assume for asset execution in limit order book markets was introduced by [17] and [82], simultaneously. We will consider a slightly altered version from Chapter 8.2 of [39]. In contrast to [17], their model dynamics contain an extra diffusion term and [39] use a risk-neutral objective function instead of a risk-averse exponential utility maximiser as in [82] (cf. [39] Chapter 8.3). We choose this version of the execution problem since it simplifies the model while retaining the problem's main mechanism and hence computational challenges.

We make several adjustments compared to standard policy gradient methods. We introduce a backward induction approach for the policy gradient method since we observe that a more precise approximation of the optimal value at the end of the time horizon has an impact on the gradient at earlier points in time. Effectively, backward induction allows us to exploit dynamic programming principles for policy gradients. Additionally, we modify the sampling oracle that is usually seen as fixed in reinforcement learning. In particular, we use a conditional sampling method. Namely, we require that at least one market order arrives in the period, where we perturb the policy. This helps us to avoid sampling uninformative periods, where the policy has no impact on the objective. Finally, we reuse random seeding in our finite difference sampling scheme to reduce the variance in the gradient estimation. With these algorithmic adjustments, we are able to learn a good policy for the given model problem. Indeed, our learned policy slightly outperforms a naively discretized version of the optimal continuous policy in numerical experiments while sharing structural resemblance. We further test numerically the role of variance in gradient estimation and its impact on the performance of the policy gradient method. We also present some results on our choice of optimization procedure.

The rest of this section is structured as follows: First, we will introduce the execution problem. Then, we share the parametrization of the policy necessary for policy gradients. After that, we present a discretization of the original problem that we use for our numerical experiments, before introducing our policy gradient algorithm and sharing some numerical results.

5.2.1 The parametric LOB model

Here we will present the optimal execution problem from [39]. Let $(\Omega, \mathcal{F}, \mathbb{P})$ be a probability space. We will denote by

- $\mathfrak{N}$ the initial inventory of assets,
- T the terminal time,

- $S = (S_t)_{0\leq t\leq T}$ a stochastic process for the mid-price given by $S_t = S_0 + \sigma W_t, \sigma > 0$, where $W = (W_t)_{0\leq t\leq T}$ is a standard Brownian motion,
- $\delta = (\delta)_{0\leq t\leq T}$ the depth at which a limit order is placed in the limit order book, which is the control variable,
- $M = (M_t)_{0\leq t\leq T}$ a Poisson process with intensity λ counting how many market orders arrived at any time,
- $N^\delta = \left(N_t^\delta\right)_{0\leq t\leq T}$ the (controlled) counting process corresponding to the number of limit orders from the original inventory who have been filled,
- $P(\delta) = e^{-\kappa\delta}$, with $\kappa > 0$, the probability that a limit order is filled if a market order arrives,
- $X^\delta = \left(X_t^\delta\right)_{0\leq t\leq T}$ the process indicating the proceeds from trade satisfying $dX_t^\delta = (S_t + \delta_t)\, dN_t^\delta$,
- $Q_t^\delta = \mathfrak{N} - N_t^\delta$ the remaining inventory to be executed.

We note that the filtration $\mathcal{F}$ can be constructed using a weak formulation approach (see e.g. [15] , Section 3.1). The control is assumed to be $\mathcal{F}$-predictable and left continuous with right limits.

The optimization problem is then given by

$$H(x,S) = \sup_{\delta\in\mathcal{A}} \mathbb{E}\left[X_\tau^\delta + Q_\tau^\delta\left(S_\tau - \alpha Q_\tau^\delta\right) \mid X_{0^-}^\delta = x, S_0 = S, Q_{0^-}^\delta = \mathfrak{N}\right],$$

where $\alpha \geq 0$ is the parameter for the linear market impact from executing the terminal inventory with market orders, $\mathcal{A}$ the admissible set of controls δ bounded from below, and $\tau = T \wedge \min\left\{t : Q_t^\delta = 0\right\}$ a stopping time. Thus, the optimal value function is

$$H(t,x,S,q) = \sup_{\delta\in\mathcal{A}} \mathbb{E}_{t,x,S,q}\left[X_\tau^\delta + Q_\tau^\delta\left(S_\tau - \alpha Q_\tau^\delta\right)\right],$$

where $\mathbb{E}_{t,x,S,q}[\cdot]$ is the expectation conditional on $X_{t^-}^\delta = x$ $S_t = S$, and $Q_{t^-}^\delta = q$ where the minus represents left limits.

With the dynamic programming principle (DPP), [39] derive the following PIDE

$$\begin{cases} & \partial_t H + \frac{1}{2}\sigma^2\partial_{SS}H \\ & + \sup_\delta\left\{\lambda e^{-\kappa\delta}[H(t,x+(S+\delta),S,q-1) - H(t,x,S,q)]\right\} = 0, \\ & H(t,x,S,0) = x \\ & H(T,x,S,q) = x + q(S-\alpha q) \end{cases}$$

and find a closed form solution.

A parametrized policy At this point, our exposition slightly deviates from [39]. First, let $0 = t_0 < ... < t_K = T$ with $K \in \mathbb{N}$ be an equidistant partition of the time interval with stepsize Δt. Then, choose a feature set consisting of indicator functions depending on the time period and the current inventory at the start of this time period, i.e. $\{1_{\bar{q}=Q_{t_k}, t_k\leq t<t_{k+1}}(Q,t)\}_{0\leq k\leq K, 0\leq\bar{q}\leq\mathfrak{N}}$ and consider linearly parametrized policies of the form

$$\delta_\Theta(Q,t) = \sum_{k=0}^{K}\sum_{\bar{q}=0}^{\mathfrak{N}} \delta_{t_k,\bar{q}} 1_{\bar{q}=Q_{t_k}, t_k\leq t<t_{k+1}}(Q,t)$$

with parameters $\{\delta_{t_k,q}\}_{0\leq k\leq K, 0\leq q\leq\mathfrak{N}} \in \Theta := [-c,\infty)^{K\times\mathfrak{N}} \subset \mathbb{R}^{K\times\mathfrak{N}}$ where $c > 0$. We lower bound Θ to ensure that $\delta_\Theta \in \mathcal{A}$. Note that this parametrization will enforce piecewise-constant policies for each time period.

This choice of parametrization for the control is motivated by mainly two factors. First, we considered the optimal control of the continuous problem as derived in [39]. This optimal control is given by

$$\delta^*(t,q) = \frac{1}{\kappa}\left[1 + \log\frac{\sum_{n=0}^{q}\frac{\tilde{\lambda}^n}{n!}\exp(-\kappa\alpha(q-n)^2)(T-t)^n}{\sum_{n=0}^{q-1}\frac{\tilde{\lambda}^n}{n!}\exp(-\kappa\alpha(q-1-n)^2)(T-t)^n}\right],$$

where $\tilde{\lambda} = \lambda\exp(-1)$, which only depends on the current time and the current number of inventory. Hence, we will only need a parametrization that takes into account these two factors.

This reduces the number of features to be considered but does not say anything about an appropriate representation. Because of the relatively complex non-linear structure of the problem, we decided against a standard linear approximation with basis function such as polynomials and opted for an almost tabular approach by using indicator functions on the state. We will refer to this approach as *semi-tabular*. Note, however, that this is not the standard tabular case as often used in simple reinforcement learning problems. The main difference is that in the tabular case, the action space is considered finite and the indicator function also depends on the action meaning that each indicator has a fixed rather than a variable weight. In the semi-tabular case, we only tabulate the state space while allowing an infinite action space via the variable coefficients in front of the indicator functions. This allows the representation of complex functional dependencies while avoiding a blow up in the number of parameters due to a large action space.

5.2.2 Discretized version of the parametric LOB model

In order to apply the policy gradient algorithm to the execution problem, we will also need to discretize the underlying stochastic process. Below we present the chosen discretization. Note that a priori it is unclear whether the optimal discrete value function converges to the true value function. Generally speaking, a finer discretization gives better approximations but leads to higher computational cost. However, in our setting a finer discretization would also require a higher action frequency. In turn, an increased action frequency can lead to problems in gradient estimation. A more detailed discussion on the impact of discretization on gradient estimation can be found in Chapter 6 and in [71].

Let $(\Omega, \mathcal{F}, \mathbb{P})$ be a probability space, $\mathfrak{N}$ the initial inventory, T the terminal time, and Δt a uniform step size on $[0,T]$ such $K\Delta t = T$ with $K \in \mathbb{N}$. Let $S = (S_{\Delta tk})_{k=0,\dots,K}$ be a stochastic process where $S_{\Delta tk} = S_{\Delta t(k-1)} + \sigma W_k^{\Delta t}$ with $\sigma > 0$ and $W_k^{\Delta t}$ iid. normally distributed with variance Δt. Let $M = (M_{\Delta tk})_{k=0,\dots,K}$ where $M_{\Delta tk} = M_{\Delta t(k-1)} + Y_k^{\Delta t}$ with Y_k iid. Poison distributed with intensity $\Delta t\lambda$ and let $\delta = (\delta_{\Delta tk})_{k=0,\dots,K}$ be a stochastic process adapted to the filtration generated by M and S. Note that with this definition, we can only choose δ once each time period and cannot specify a schedule of control variables dependent on inventory level within the time period. Moreover, $N = (N_{\Delta tk})_{k=0,\dots,K}$ is the controlled counting process where $N_{\Delta tk} = N_{\Delta t(k-1)} + Z_k^{\Delta t}$. The $Z_k^{\Delta t,\delta_k}$, conditioned on $Y_k^{\Delta t} = n$ are distributed according to a binomial distribution with n trials and a success probability of $p_k = \exp(-\kappa\delta_k)$. Note that the unconditional distribution of $Z_k^{\Delta t,\delta_k}$ is Poisson with intensity $\exp(-\kappa\delta_k)\Delta t\lambda$. The stochastic process $X^{\Delta t} = (X_{\Delta tk})_{k=0,\dots,K}$ is then given by $X_{\Delta tk} = X_{\Delta t(k-1)} + (S_{\Delta tk} + \delta_{\Delta tk})Z_k^{\Delta t,\delta_k}$. Lastly, $Q = (Q_{\Delta tk})_{k=0,\dots,K}$ is defined as $Q_{\Delta tk} = \mathcal{N} - N_{\Delta tk}$.

5.2.3 Algorithm

In this section, we will explore the modifications we made compared to a standard policy gradient method introduced in Chapter 2. These modifications are needed as convergence was not achievable with a standard method. We will first consider modifications directly to the algorithm. Then, we discuss some adaptations to the sampling procedure, since certain characteristics of the sampling such as the ability to run controlled experiments can be used to enhance learning. We will state our implemented procedure in Algorithm 5.

Algorithm 5 Policy gradient method for parametric LOB model

Input: Sampling oracle for parametric LOB model with assumptions as in section 5.2.3.2 and random seeding denoted by ω; Finite difference parameter $r > 0$; Number of time steps T; Number of trajectories N_{traj}; Number of gradient steps N_{grad}; Gradient step-sizes $(\gamma_l)_{l=0}^{N_{\text{grad}}}$

Initialize: Parameters $\delta_T^0 = (0, \dots, 0) \in \mathbb{R}^{\mathfrak{N}}$ for policy of the form δ_Θ

for t_{out} in T to 0 **do**

if $t_{\text{out}} < T$ **then**

$\delta_{t_{\text{out}}}^0 \leftarrow \delta_{t_{\text{out}}+1}^{N_{\text{grad}}}$

end if

for l in 0 to N_{grad} **do**

$\delta_{t_{\text{out}}}^{l,-} \leftarrow \delta_{t_{\text{out}}}^l - r$ and $\delta_{t_{\text{out}}}^{l,+} \leftarrow \delta_{t_{\text{out}}}^l + r$ {Addition with r is componentwise.}

$N_Q \leftarrow (0)_{q=1}^{\mathcal{N}}$ and $Q \leftarrow (0)_{k=1}^{N_{\text{traj}}}$

for k in 1 to N_{traj} **do**

Choose seed ω

Draw initial state $S_{t_{\text{out}}}(\omega), Q_{t_{\text{out}}(\omega)}$ and arrivals $M_{t_{\text{out}}}(\omega)$

Draw $N_{t_{\text{out}}}^{\delta_{t_{\text{out}}}^{l,-},q}(\omega)$ and $N_{t_{\text{out}}}^{\delta_{t_{\text{out}}}^{l,+},q}(\omega)$ where $q = Q_{t_{\text{out}}(\omega)}$

Sample the remaining paths $(S_t(\omega), M_t(\omega))_{t=t_{\text{out}}+1}^T$

Sample the counting processes $(N_t^{\delta_t,-}(\omega), N_t^{\delta_t,+}(\omega))_{t=t_{\text{out}}+1}^T$ starting at initial inventories$(q - N_{t_{\text{out}}}^{\delta_{t_{\text{out}}}^{l,-},q}(\omega), q - N_{t_{\text{out}}}^{\delta_{t_{\text{out}}}^{l,+},q}(\omega))$ and where $\delta_t = \delta_t^{N_{\text{grad}}}$ {Sampling is stopped when inventories are depleted}

Calculate gains of trades $(X_t^{\delta_t,-}(\omega), X_t^{\delta_t,+}(\omega))_{t=t_{\text{out}}+1}^T$, inventory paths $(Q_t^{\delta_t,-}(\omega), Q_t^{\delta_t,+}(\omega))_{t=t_{\text{out}}+1}^T$,

$$H_q^- = X_\tau^{\delta_\tau,-}(\omega) + Q_\tau^{\delta_\tau,-}(\omega)\left(S_\tau(\omega) - \alpha Q_\tau^{\delta_\tau,-}(\omega)\right)$$

where $\tau = T \wedge \min\left\{t : Q_t^{\delta_t,-} = 0\right\}$ and H_q^+ analogously

Calculate values (H_q^-, H_q^+) and

$$\hat{\nabla}_{k,l}^q H \leftarrow \frac{H_q^+ - H_q^-}{2r}$$

$N_Q(q) \leftarrow N_Q(q) + 1$ and $Q(k) = q$

end for

Calculate

$$\hat{\nabla}_l H \leftarrow \left(\frac{1}{N_Q(q)} \sum_{k=1}^{N_{\text{traj}}} \hat{\nabla}_{k,l}^q H 1_{Q(k)=q}\right)_{q=1}^{\mathfrak{N}} \tag{5.1}$$

Make Gradient ascent step

$$\delta_{t_{\text{out}}}^{l+1} \leftarrow \delta_{t_{\text{out}}}^l + \gamma_l \hat{\nabla}_l H$$

end for

end for

5.2.3.1 Changes to the original algorithm

As a first modification, we change the order of optimization. In the context of an episodic setting, [88] propose to simultaneously optimize over each period. We take a different approach by using a backward stepping scheme. This means we start optimizing in the last period. Once we are satisfied with the optimization for this period, we will go to the previous time-step and start learning the control within this period. This procedure is repeated until the first period is reached. Furthermore, if sampling costs are considered per time step rather than per episode, we believe that this modification improves sample efficiency. This is because instead of sampling an entire episode for optimizing the last period, using our scheme, we only need to sample one time period. This increase in sample efficiency can be enhanced by using the control of the subsequent period as a good prior for the current period, assuming that controls do not fluctuate too much. Once a good prior is established, one can often decrease the number of gradient steps in the optimization, reducing the number of required samples. In our application, we will try to exploit this behaviour by having a relatively large number of gradient steps in the last time period before reducing them for the previous time steps. As a caveat, one should point out that using a backward induction scheme requires the possibility to sample credible starting points for each time period from the sampling oracle. We will comment on this point in the next subsection. Note that [68, 121] recently proposed a similar approach for exploiting dynamic programming principles for policy gradients. Specifically, [68] applied these ideas to a backward SDE solver, while [121] focused on a finite-time horizon reinforcement learning problem with a finite state and action space.

In direct search policy gradient methods, it is common (see, e.g., [88, 95]) to use a perturbed gradient zeroth-order estimator as explained in [158]. The defining characteristic of such an estimator is that it is randomly perturbed in all parameter directions. However, using our parametrization we are in a semi-tabular case where features are mutually exclusive and only one parameter always determines the actual policy. Given a current inventory q and time-step t, it only makes sense to perturb the parameter that is currently active according to the indicator in the parametrization of the policy. Hence, we can use standard one dimensional finite difference estimators for the gradient at each possible case in the semi-tabular setting.

Finally, we use an adaptive step size heuristic for the ascent algorithm, which can be seen as an inexact line search method [162, 202]. Given a step size γ, we start by gathering the perturbed samples for the gradient estimate for a given point in parameter space. Then, we perform a gradient ascent step to a new point and sample again around the new point using the perturbation. Besides giving an estimate of the gradient at that new point, the samples can also be used to estimate the value of the policy at this new point. Assuming that we did the same for the previous point, we compare the value estimates at the new and the old point. If the value estimate at the new point is better than the value estimate at the old point, we continue using the same step size starting from the new point. In the other case, we halve the step size before continuing and we carry on with comparing the value estimate of the next point with the old point. If the step-size continues to diminish under a certain thresh-hold, we reset it to its original size and perform a gradient step. This procedure does not seem to be strictly necessary, but in our experience improves stability of the optimization. For simplicity, we will omit these details in the description of the general algorithm (Algorithm 5).

5.2.3.2 Changes to the sampling oracle

Besides the changes to the algorithm, we also make some additional assumptions on the sampling oracle. With these modifications to sampling, we can mitigate three major issues for policy gradient methods: exploration only along optimal paths; high variance of the gradient estimate; and slow learning due to rare events. In this section, we will describe these problems in more detail and explain how they can be mitigated by smart sampling strategies. Note, however, that while we make additional assumptions on how we sample, the underlying environment remains unaltered.

To enhance exploration, we assume that we can start randomly at each point in the environment at any point in time. In our case, this allows us to use random initializations over the initial inventory at each point in time. This random initialization has two major benefits. First, it enforces exploration for policies off the optimal path

and, thus, leads to more robust policies. In our case, this eases the comparison between the optimal control in the continuous model and the learned policy as well. The second benefit is that the ability of initializing in any given time period is required to allow the backward stepping as explained above. Without the random initialization, we would not be able to use backward stepping, because it would be unclear where to start in the later periods.

Another challenge to policy gradient methods is that gradient estimation is often noisy. In the worst case, the variance of the gradient dominates the actual information of the gradient. For finite difference estimators, it is well known that for small difference parameters r, the variance is higher (e.g. [73]). A common remedy is to reuse the same random seed for both function evaluations in the finite difference estimator. In the context of zeroth order optimization, this idea has been discussed recently in [29]. We extend this idea to the context of policy gradient methods by assuming that we can sample using the same random seeds for both elements of a finite difference estimator. This means for example that the diffusion W and the arrival process M is the same along each path. For the controlled counting process N^δ, we assume that if we have δ_t^1 and δ_t^2 with $\delta_t^1 < \delta_t^2$, we simulate first the counting $N_t^{\delta_t^2}$ with the lower probability parametrized by δ_t^2 before sampling the difference in counting $N_t^{\delta_t^2 - \delta_t^1}$ and then set $N_t^{\delta_t^1} = N_t^{\delta_t^2} + N_t^{\delta_t^2 - \delta_t^1}$. From a causal inference stand point, we basically assume that we can conduct perfectly controlled experiments in our given environment.

The final adjustment we make, is to assume sampling conditional on certain events occurring is possible. In our problem, to gain useful information for the gradient estimate, we require that at least one market order arrives in the period, where we perturb the policy (i.e. $M_t > 0$). Otherwise, if $M_t = 0$, the choice of δ_t will not impact the rewards. We can circumvent sampling in uninformative periods by sampling conditional that $M_t > 0$ for the period where we perturb the policy. In experiments, this modification improves sampling efficiency significantly. Note that while this adjustment does not impact the direction of the gradient estimate, conditional sampling increases the size of the gradient estimate. While an exact rescaling would require model knowledge, we can heuristically decrease the scale of the gradient by adjusting the learning rate of the gradient ascent algorithm accordingly.

5.2.4 Numerical results

With the implementation of the policy gradient method using the adjustments described above, we now show that the algorithm learns a policy that resembles key characteristics of the continuous model solution and even slightly outperforms a naively discretized version of the model policy in empirical evaluations. For our experiments, we use the parametrization

$$\begin{aligned} &T = 30, \quad \lambda = 50/60, \quad \kappa = 100, \quad \alpha = 0.001, \quad \mathfrak{N} = 5, \\ &S_0 = 30, \quad \Delta t = 1 \text{ and } \quad \sigma = 0.01. \end{aligned}$$

We simulate the policies in the discretized environment. In Figure 5.1, we show the optimal policy path for the continuous model and the learned policy depending on the current level of inventory denoted by q and the current time t.

One can see several similarities between the model policy and the learned solution. Namely, the scale of the δ parameters is of similar magnitude and both follow a downward trend. Furthermore, the policies are inversely ordered by inventory and are concave over time. However, one can easily spot differences in the policies, e.g. the learned policy is slightly shifted downwards and the curvature especially in the higher level of inventory deviates.

In Figure 5.2, we compare the distribution of gains from trade under the model policy and a learned policy. The distribution is relatively similar although the model policy has a slightly higher standard deviation (0.016 versus 0.020). In terms of mean return for 5×10^3 paths, the learned policy outperforms the model policy by around 0.0084. Performing a two-sample t-test reveals that the difference is statistically highly significant (i.e. the p-value equals 4.34×10^{-116}). The difference in performance, however, is relatively small from the viewpoint of the application. We suspect that the observed differences in policies are due to our choice of discretization.

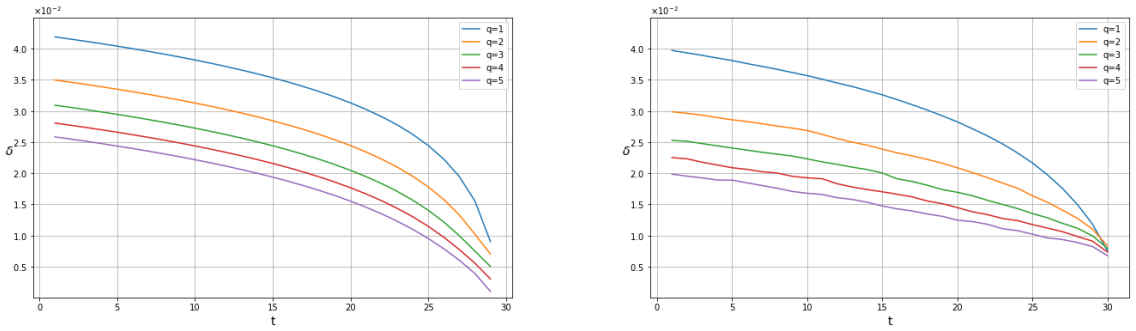


Figure 5.1: Plotting the optimal continuous policy δ^* (left) and a learned policy δ_Θ (right) in dependence of time and current inventory. Note for the learned policies 5×10^3 samples were used per gradient estimation step. For the learned policy, the initial policy was $\delta_\Theta = 0$.

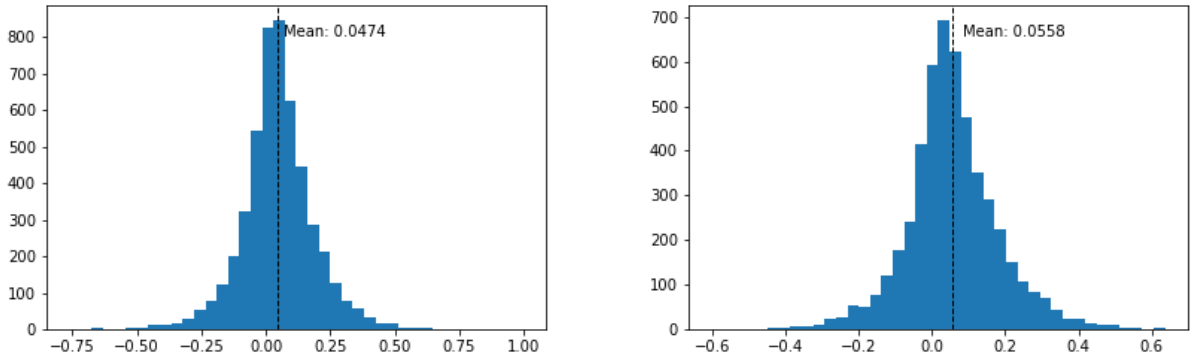


Figure 5.2: Histograms based on 5×10^3 evaluations of gains from trade for the discretized optimal continuous policy (left) and a learned policy (right).

Testing this hypothesis empirically is not straight forward since the appropriate scaling of a policy gradient method depends on the level of discretization (see Chapter 6).

In the rest of this section, we will discuss various empirical results on the variance of gradient estimates and its impact on optimization. We will especially highlight the effect of some of our algorithmic choices meant for variance reduction. After that, we will consider other aspects of the gradient procedure. Namely, we will examine the impact of initialization, length of training, and a backward inductive learning procedure. Finally, for purposes of presentation, we report results on the performance with regards to the objective in terms of costs, which is defined as the proceeds from trading with a switched sign.

5.2.4.1 Variance in gradient estimation and its impact

A major challenge in optimizing with zeroth-order methods is establishing a good estimator for the gradient. Specifically, in our setting, we found that the stochasticity of the underlying problem combined with a small difference parameter causes challenges with the variance of the gradient estimate. While in general having variance in the gradient estimate in a gradient-based optimization method is sometimes even desirable in

non-convex settings, we found in our case that if using naive sampling with a sensible number of paths the noise outweighed the informative part of the gradient estimate. As a remedy, we reuse the random seed in sampling for both components of the finite difference estimator. While this is a well-known methodology for variance reduction in finite difference estimators in Monte Carlo methods (see, e.g., [73]), we are not aware of any previous application of this technique for policy gradient methods in a similar setting. The effects of this change in sampling can be seen in Figure 5.3.

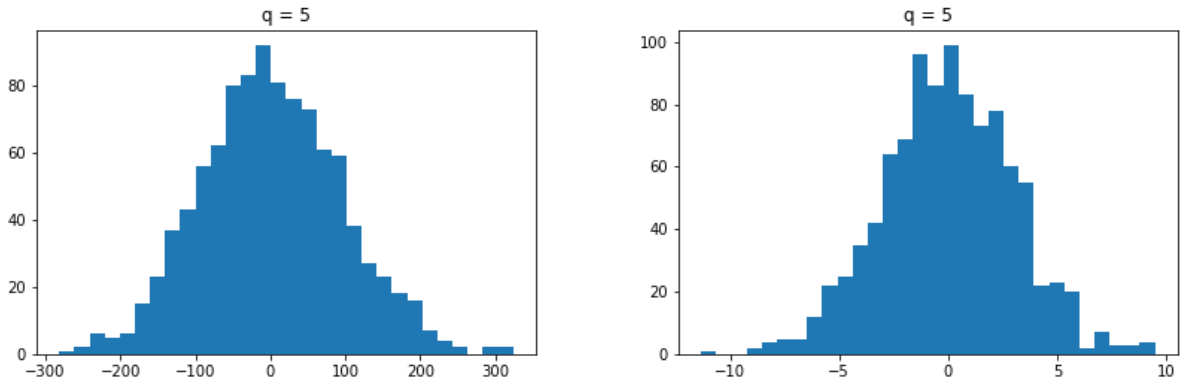


Figure 5.3: Histograms of a 10^3 gradient estimates with 10^3 finite difference components each comparing naive sampling (left) and sampling with reused seeds (right) for an inventory $q = 5$ at the first time step.

In Figure 5.3, we compare the distribution of the two sampling methods for an inventory $q = 5$ at the first time step. Looking at the difference in scale, one can see that the variation of gradient estimates is reduced by a factor of more than 10. In terms of standard deviation the change in sampling leads to a 42-fold reduction. We will not compare the learned policies of both sampling mechanisms since we were not able to achieve convergence for the naive sampling procedure. In Figure 5.4, we show that a similar impact can be observed across all levels of inventory. Henceforth, we will only consider the estimator with a fixed seed.

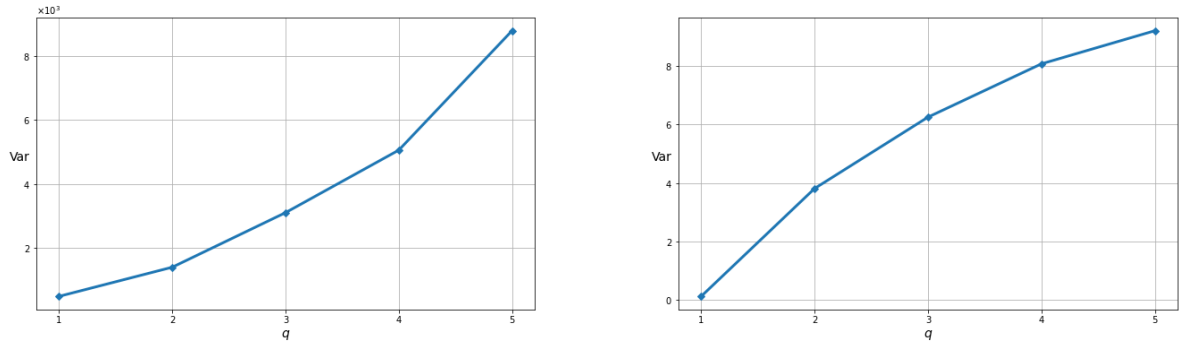


Figure 5.4: Variance of gradient estimates in the number of initial inventory starting at the first time step using 10^3 gradient estimates with 10^3 finite difference components each comparing naive sampling (left) and sampling with reused seeds (right).

Figure 5.4 also reveals another interesting observation on gradient estimates. Namely, that the variance of

the estimates is increasing in the initial inventory. This behaviour is expected since the stochastic process is stopped once a level of zero is reached. Since it is more likely that the stopping time is earlier if the level of initial inventory is low, this leads to shorter stochastic processes. Indeed, looking at the variance in comparison to the number of time steps left, we observe a similar behaviour in the propagation of variance over time (see Figure 5.5).

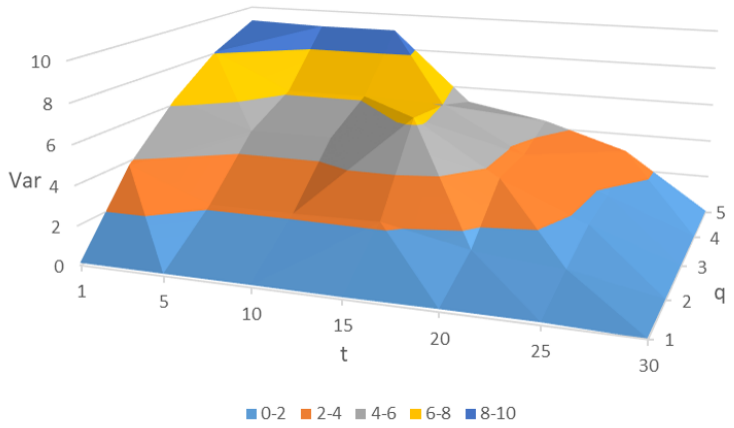


Figure 5.5: Evolution of variance in gradient estimates over inventories and time steps using 10^3 gradient estimates with 10^3 finite difference components per time step in the seeded case.

Next, we consider the impact of the number of samples on the variance of the gradient estimator. In the first graph of Figure 5.6, one can see that the standard deviation of the gradient estimate declines as the number of samples increases. While this behaviour is expected, it is unclear what the effect of the decreasing variance is on the overall learning. We investigate this effect in the second graph of Figure 5.6. In this graph, we compare the costs associated with a learned policy where we vary the number of samples for each gradient estimate. We observe that there is an initial benefit of reducing the variance of the gradient estimate for a relatively low number of samples. This effect diminishes as the number of samples increases and the curve becomes flat. However, comparing the functional forms of policies learned with different number of samples (bottom row, Figure 5.6) reveals that a policy learned with a lower number of samples tends to be more volatile over time and breaches the ordering for different levels of inventory. The effect of this undesirable policy features is almost non-existent in terms of incurred costs using empirical test but impacts the interpretability of the policy.

A hyperparameter central to the estimation of finite difference methods is the perturbation parameter r. It is well known, that there is a direct trade off between variance and bias of the finite difference estimator in the parameter r, e.g. [73]. Namely, a larger r is associated to a lower variance but a higher bias and vice versa. In the context of optimization, it seems reasonable to take into account the width of the extremum and choose r accordingly. If, for example, one would like to optimize a function of the form given in Figure **??** using a zeroth-order descent method based on a finite difference approximation, a too large r could lead to a gradient pointing in the wrong direction. This, in turn, can imply a bad optimization outcome. We make this observation for our problem using too large values of r as can be seen in Figure 5.7 on the right. For a decreasing r, the performance of the optimization strongly improves, if using a similar or smaller scale as the width of the extrema in Figure **??**. However, if r becomes too small, the variance of the estimator explodes as can be seen in Figure 5.7 on the left[1]. This, again, leads to a bad optimization outcome and, hence, the bias-variance trade off of

[1]According to [73], Section 7.1.1, reusing the random seed is expected to mitigate the increase in variance, i.e. from $\mathcal{O}(1/r^2)$ to $\mathcal{O}(1/r)$ as r diminishes. However, since the path-wise trading results are not continuous in the parameter due to the controlled jump process, the variance still surges.

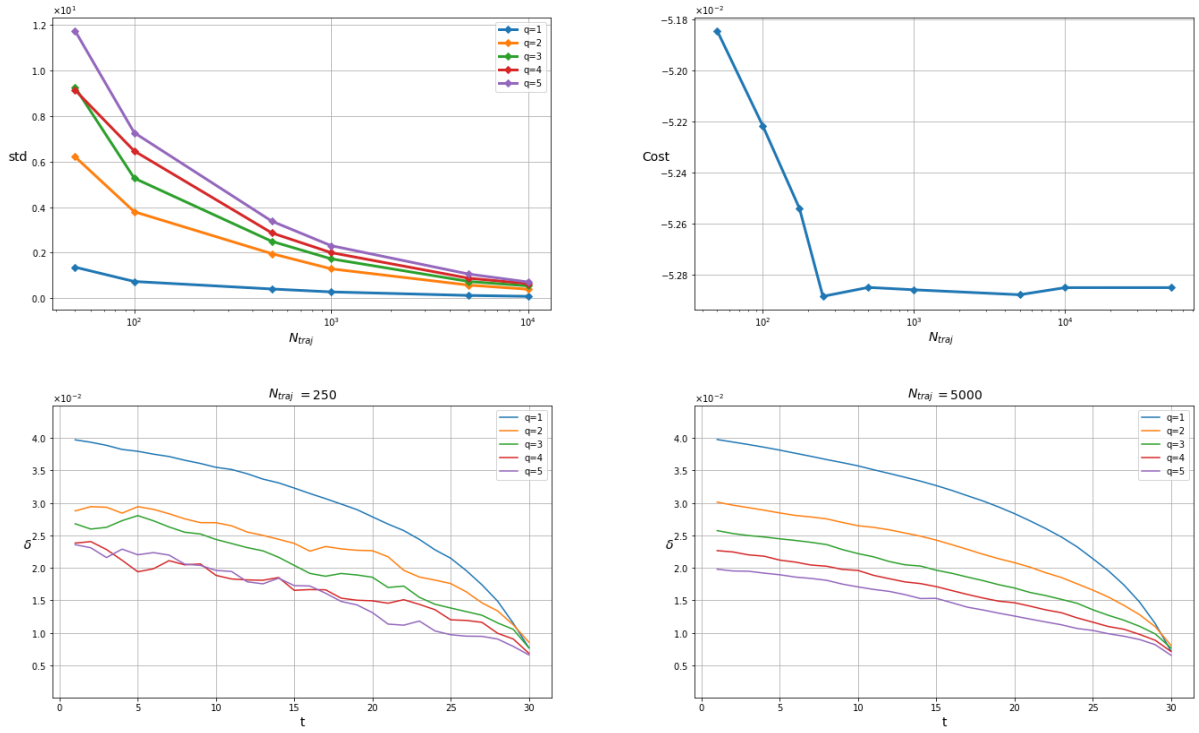


Figure 5.6: Top row: Standard deviation of gradient estimates (based on 10^3 gradient estimates) in dependence of number of samples per gradient estimate (left) and comparison of costs of learned policies with different number of samples per gradient estimate using 200 training steps in each period (right). Bottom row: Learned policies in dependence of time and inventory level with differing number of samples per gradient estimate.

finite difference estimation in r also propagates through to the performance of the zeroth-order optimization procedure.

As a final comment in this section, we will consider the impact of the diffusion component in the model on gradient estimation and our ability to learn a good performing policy. We vary the volatility parameter σ to determine the influence of the diffusion component. In Figure 5.8, we see the impact of the model volatility on the standard deviation of the gradient estimates with a fixed sample size. As one might expect, an increase in model volatility corresponds to an increase in the variance of the finite difference estimates, ceteris paribus. The second graph in Figure 5.8 reveals our capability to learn a good policy for changing σ since the expected cost should not change in dependence on σ for a fixed policy (see ansatz in Section 5.2.1). Observing that the cost increases as we increase the model volatility, we can presume that learning deteriorates for growing model volatility. We include this observation to show that the ability to learn depends on the model-inherent volatility.

5.2.4.2 Optimization procedure

After investigating the impact of noise in the gradient approximation, we turn our attention to the optimization method. In particular, we investigate whether our use of a backward induction scheme has a benefit compared to the simultaneous approximation approach. Furthermore, we would like to see how the number of gradient descent steps impacts the performance of a learned policy. Before considering these two aspects, we make a short comment on sampling efficiency. In total, the number of samples needed is determined by three components: the

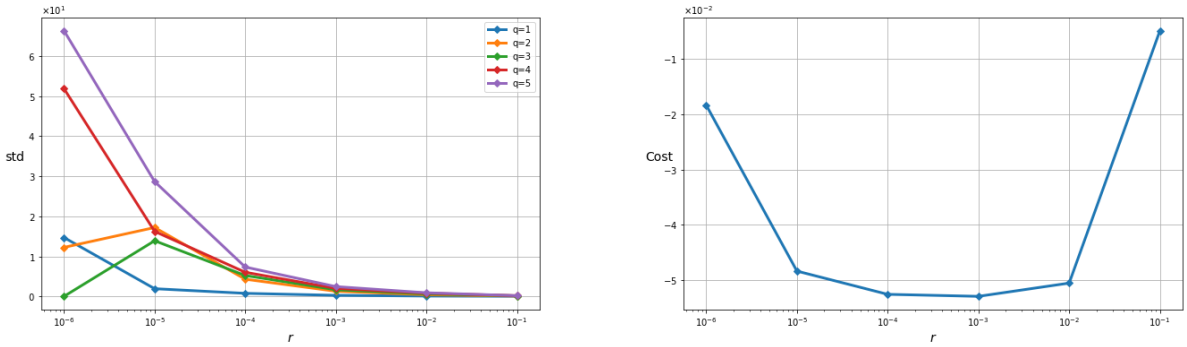


Figure 5.7: Standard deviation of gradient estimates (based on 10^3 gradient estimates and 10^3 samples per estimate) in dependence of the perturbation parameter r (left) and comparison of costs of learned policies with different perturbation parameters r (right).

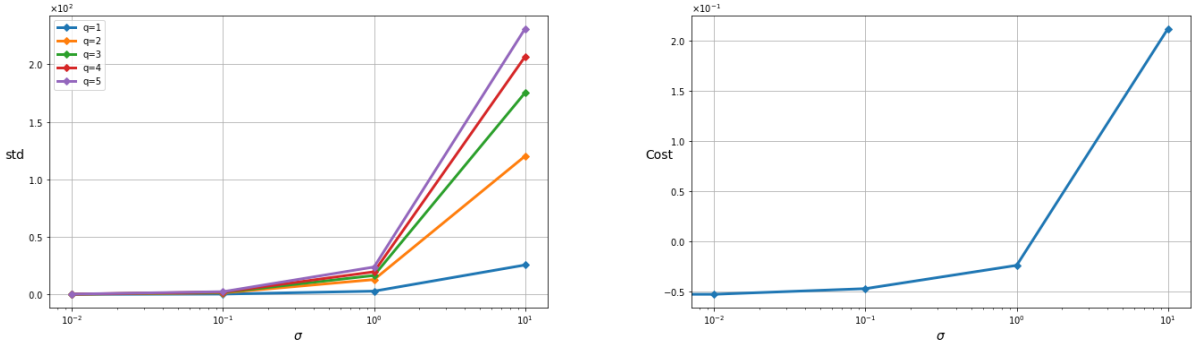


Figure 5.8: Standard deviation of gradient estimates (based on 10^3 gradient estimates and 10^3 samples per estimate) in dependence of the model volatility σ (left) and comparison of costs of learned policies with different model volatilities σ (right).

number of time-steps in the given problem, the number of gradient descent steps, and the number of samples for each gradient estimate. The total number of samples is the product of these three factors.

As mentioned above, our goal is to investigate the difference of local optimization used in backward scheme to a simultaneous global optimization. For that purpose, note that the calculation on the sample number is independent on whether one chooses to use a backward stepping algorithm as in Algorithm 5 or a simultaneously optimizing algorithm similar to the approach in [88]. Hence, we can easily choose the same number of samples for both procedures. In Figure 5.9, we plot the learned policy with a backward stepping optimization and a simultaneous optimization using the same number of samples (200 gradient steps with 10^3 samples for the finite difference estimate). Comparing the learned policies, one can easily see that the backward stepping optimization creates a policy that is more in line to what one would expect (i.e. ordering by inventory and monotonously decreasing over time). In contrast, for the policy learned by simultaneous optimization the ordering by inventory is not consistent, the overall level seems to be too low and the monotonicity is partially not satisfied. These differences are reflected in terms of costs with the backward stepping algorithm leading to a cost decrease by -0.02 units compared to the simultaneous optimization where the costs were evaluated along 5×10^5 paths.

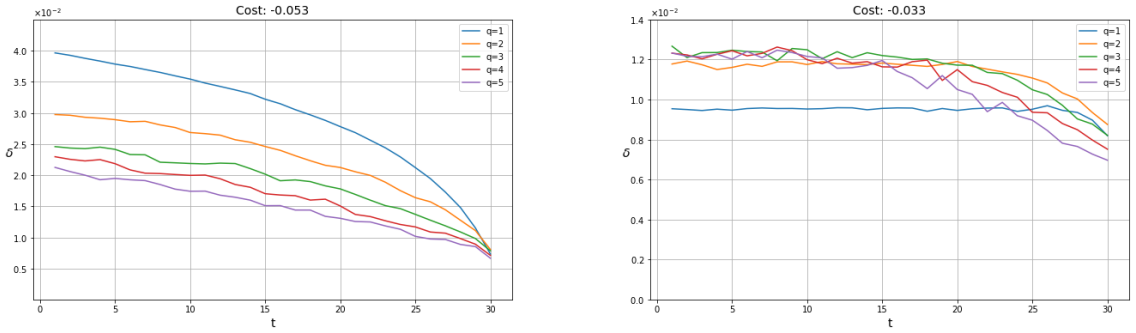


Figure 5.9: Learned policy with a backward stepping optimization (left) and a simultaneous optimization (right) using the same number of samples.

We conclude this section by discussing how the number of gradient steps effects the performance of the learned policy in the case of Algorithm 5. In Figure 5.10, we compare the costs for policies learned over a varying number of gradient steps taken at each point in time. For the standard algorithm given by the blue line, one can see a clear convergence behaviour with flattened improvements at around 100 gradient steps per time period. For the orange line, we take 200 gradient steps at the last time period while the previous time periods vary in the number of gradient iteration uniformly according to the axis. We are interested in this scheme since we reuse the learned parameters of the last period as a prior in the previous period. Hence, the consideration to be tested was whether an improved optimization in the last period can produce a good prior for the previous periods leading to a lower number of total samples needed. Indeed, for low numbers of gradient iterations, the improvement is quite significant. Thus, for situations with constraints on the number of samples one might want to consider such an adaptation. However, as to be expected, the benefit diminishes as the number of gradient steps grows larger.

5.3 Policy gradients for GAN LOB model

In the previous section, we considered a policy gradient method for a parametric model for execution in LOBs. While this investigation provided interesting results, especially on aspects of algorithm design, it has several limitations. First, since for this model an optimal policy can be derived in closed form, the informative content of the learned results is limited. Second, since the control parameters impacted the final profits from trading through a change in the sampling distributions for limit orders being filled, the objective function was not directly differentiable in the control parameter along paths. This required us to use the zeroth-order approximation of gradients rather than a path-wise approximation (cf. Chapter 2) and led to the difficulties encountered with variance in the gradient estimate. Finally, the parametric LOB model itself lacks several key features of real limit order book markets.

Some of the aspects not reflected in the parametric model are: First, the model does not take into account the actual shape of the order book as factor for fill probabilities. Neither does the model consider the LOB's matching mechanism, which can have significant impact on the functioning of the LOB market and asset execution. Furthermore, placing limit orders may have an impact on the market itself, which is also not considered in the simple LOB model. Finally, the actual control mechanism in a real LOB market may be markedly different. For example, in the CME's 3-months SOFR future market most of the trading happens at the touch, meaning only the best levels of the LOB have a chance of getting filled. Thus, controlling the depths in which a limit order is placed becomes much less relevant. Instead, either the time an order is placed in a first-in-first-out order book

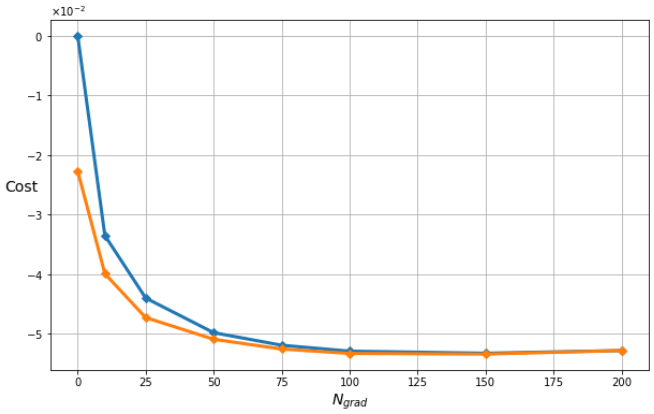


Figure 5.10: Costs evaluated along 5×10^6 paths for policies learned with a varying number of gradient steps and 10^3 samples for each gradient estimate. For the blue line, the number of gradient steps is uniform across time. For the orange line, 200 gradient steps are taken in the last time period while the previous vary uniformly according to the axis.

or the size of the order in a pro-rata LOB becomes significant for execution.

In this section, we aim to overcome some of the short-comings from the previous section. For this purpose, we consider a policy gradient method for optimizing execution strategies in a non-parametric LOB simulator trained with real world data. This LOB simulator is an extension to the GAN model for LOB simulation proposed in [48]. The usage of neural networks for simulating LOB dynamics makes it practically impossible to derive closed form solutions for optimal trading in this model. Furthermore, the extensions we propose to the LOB simulator make the trading objective function differentiable with respect to the control parameters, thus, allowing us to use path-wise derivatives for estimating gradients instead of relying on zeroth-order approximations. Lastly, the GAN model being directly trained on real LOB data provides an enhanced realism for the simulation of trading. We will discuss this point in more detail below.

This section will proceed as follows: First, we formulate the asset execution problem within the GAN LOB model for pro-rata execution. Then, we provide some additional implementation details for the GAN model which allow us to apply a policy gradient method with path-wise gradient estimation and give evidence for the quality of the LOB simulator. Then, we discuss algorithmic considerations for the application of policy gradients for learning within this framework. We conclude by presenting results of this learning approach.

5.3.1 Execution problem in GAN LOB model

To avoid repetition, we base this discussion on our exposition and notation on LOBs presented in Section 4.2 and highlight any differences. First, throughout this section, we assume that the state of the LOB is given by the volumes of the top three levels of the centered order book snapshot for each side, i.e.

$$Z_t = S^3_{p^*_t}(t).$$

The transition between LOB states will be given by the generator component of a trained GAN denoted by f_{GAN}. For $\Delta t > 0$, the transition function f_{GAN} satisfies the following mapping

$$(Z_{t+\Delta t}, \Delta p^*_{t+\Delta t}, \rho_t) = f_{\text{GAN}}(Z_t, \varepsilon_t)$$

where $Z_t \in \mathbb{N}_{>0}^6$ is the current order book state and ε_t is a real-valued random noise vector with independent and standard normally distributed components. If we chain multiple transitions together, the random noise vectors are iid. and constitute the latent noise vector of the GAN model. In addition to the next LOB state $Z_{t+\Delta t} \in \mathbb{N}_{>0}^6$, the GAN transition will also provide the change in mid-price $\Delta p_{t+\Delta t}^* \in \delta\mathbb{Z}$ where $\delta \in \mathbb{R}_{>0}$ is the tick size of the LOB. The current mid-price can be easily calculated using the following relation

$$p_{t+\Delta t}^* = \Delta p_{t+\Delta t}^* + p_t^*$$

and an initial mid-price p_0^*. Finally, $\rho_t = (\rho_t^b, \rho_t^a) \in [0,1]^2$ denotes the fill ratios during the transition of volume present on the best bid level ρ_t^b and best ask level ρ_t^a at time t.

The fill ratios ρ_t are used as a simplification for the pro-rata execution mechanisms introduced in Section 4.2. As explained in Section 4.2, to calculate the fraction executed of a given limit order in a pro-rata order book, we require for each arriving order which crosses the spread, the size of the order and the relative size of the given limit order to the total volume on the same order book level. To simplify the training of the GAN model, we make the assumption that it is sufficient to consider the ratio of the volume of arriving orders crossing the spread over a time period Δt with the current order book volume at that level to calculate the executed fraction of a given limit order. Thus, in the training data for the GAN, ρ_t^b is given as

$$\rho_t^b = \frac{\mathrm{MO}(\Delta t, t, b)}{V_{\lfloor p^* \rfloor}(t)}$$

where $\mathrm{MO}(\Delta t, t, b)$ is the volume of market sell orders arriving during the time interval $[t, t+\Delta t]$ at the best bid level at time t and $V_{\lfloor p^* \rfloor}(t)$ is the volume at the best bid level at time t. For a given limit order of size q on the best bid level, we assume that $q\rho_t^b$ of the limit order will be executed in the time interval $[t, t+\Delta t]$. Analogously, we calculate ρ_t^b and make the same assumption for limit orders on the best ask. We leave the investigation of simulating more realistic execution mechanisms using neural networks for future work.

We consider an execution problem where a trading agent seeks to trade out of an initial inventory of an asset using only limit orders until a terminal time T. The trading agent can place their limit orders at the best ask level if their inventory is positive and at the best bid level if having a short position. The control variable is the size of the trader's limit order position on the best ask and bid level of the LOB, since in pro-rata type markets traders compete for arriving market orders by their relative order size to the market, e.g.[59, 83]. We parametrize the control variable by a time-dependent family of linear feedback controls of the current inventory of the trading agent, i.e.

$$(u_a^\theta(t), u_b^\theta(t)) = (\theta_t q_t I_{q_t>0}, \theta_t q_t I_{q_t<0})$$

for $t \in [0,T]$ where q_t is the current inventory and $I_{\cdot}$ denotes an indicator function. Since the simulator considers discrete transitions between LOB states, we only update the position a finite number of times after each transition by the GAN and thus also only consider a finite family of parameters $\theta = (\theta_{i\Delta t})_i^n$ where $n\Delta t = T$ and $\theta_{i\Delta t} \in \mathbb{R}_+$. Furthermore, after each transition, we remove any outstanding limit orders and place new orders according to the control variable as explained in Section 4.2.

Given a trader's position $(u_a^\theta(t), u_b^\theta(t))$ and a transition's fill ratios (ρ_t^a, ρ_t^b), we assume that $(\rho_t^a u_a^\theta(t), \rho_t^b u_b^\theta(t))$ of the trader's limit orders will be filled. The trader's proceeds from trade in this time period thus correspond to

$$X_t^\theta = \rho_t^a u_a^\theta(t)(p_t^* - \delta/2) + \rho_t^b u_b^\theta(t)(p_t^* + \delta/2).$$

For optimization, we propose an analogous objective as in Chapter 8.2 of [39] and in the previous section. Namely, the objective is to maximize the proceeds from trade while penalizing remaining inventory at the terminal time with a linear market impact function with parameter $\alpha > 0$:

$$H(\theta) = \mathbb{E}\left[\sum_{i=1}^n X_{i\Delta t}^\theta + Q_T^\theta \left(p_T^* - \alpha Q_T^k\right) \mid p_0^* = p^*, Q_0^\theta = \mathfrak{N}\right],$$

where $\mathfrak{N}$ is the trader's initial inventory and Q_t^θ the current inventory under trading strategy θ.

Remark 11. *To our knowledge, we are the first to study an optimal execution problem over multiple time-steps in a pro-rata market with competition over the size of the limit orders. Thus, we would like to highlight how the trade-off present in this control problem differs from the one in the previous section. Namely, increasing the size of a limit order allows a trader to capture more of any incoming market order. This may lead to overquoting the actual trade target in order to increase the probability of being filled. However, overquoting carries the risk for the trader to overshoot their trading target and the need to unwind any remaining positions via market orders. Thus, the trader needs to trade-off the costs of a possible overshoot versus the cost of remaining inventory to find the optimal size of limit orders. Over multiple time steps, the trader can vary this consideration depending on the time left.*

5.3.2 Gumbel-LOB GAN

Next, we describe some implementation details for obtaining a good transition function f_{GAN}. The GAN market model we propose is an extension to the model from [48]. For the sake of brevity, we refrain from describing the general training procedure of a GAN and refer to [48] for more details. Instead, we focus on discussing the extensions needed to apply a policy gradient method with a path-wise gradient estimation.

In order to apply policy gradients in this environment, we need to ensure that the empirical version of the objective is sufficiently differentiable with respect to the controls. We achieve this by simulating additionally to the state of the order book, the fill ratios and the price transitions. In contrast, [48] only simulates the state of the LOB and keeps track of price changes implicitly by observing shifts in a centred order book. To chain transitions together, order books with observed price changes require reordering in their framework, which poses a challenge to gradient estimation via backward differentiation. Furthermore, [48] does not consider fulfilment of limit orders.

To consider fulfilment of limit orders, we simply add additional output nodes to the generator architecture associated to fill ratios and price moves. For the fill ratios, we project the unrestricted learned ratios onto a the interval $[0, 1]$ using a hard-tanh activator function on the nodes. The inclusion of price changes poses a more difficult problem since price changes are not continuous but discrete in LOBs. To enable an effective learning of price changes, we encode them as a categorical variable and limit ourselves to a maximum price change of two tick-sizes yielding 5 different categories. In our application, this choice of encoding is justified as larger price moves are not observed in our LOB training data set. Faster moving markets may require additional categories. To simulate the price change, we thus include 5 extra nodes in the output layer of the GAN generator and apply the Gumbel-softmax activation described in [35, 110] to the additional nodes. In particular, the Gumbel-softmax activation allows a hard encoding of the price change while still be usable for backward differentiation [110]. For more details on using Gumbel-softmax activation to simulate categorical data with GANs, see [35]. Since this adjustment enables us to use the GAN simulator for path-wise gradient estimation, we will refer to our implementation of the GAN LOB simulator as Gumbel-LOB GAN.

Also, note that we do not change the input of the generator neural network and only condition the simulator on an order book snapshot. For the LOB snapshot, we use an analogous transformation for the order volume V as in [48] by using $\sqrt{V}/100$ as input and output of the neural networks. Furthermore, we enforce a minimum order book volume of 1 on each order book level to ensure differentiability of the volume transformation. The discriminator of the GAN is modified to account for the additional output nodes of the generator. Finally, in comparison to [48] we adjust the architecture of the discriminator by adding an extra layer and using either 48 or 96 nodes per layer.

Data set and simulation results As data set for training the GAN, we use the same LOB data on 3-month Secured Overnight Financing Rate (SOFR) futures traded at CME as described in more detail in Section 4.4. Furthermore, we also use the same aggregation strategy in event time where one order book transition corresponds to 250 order book events. In addition, we also calculate the fill ratios as described in Section 5.3.1 and include the price moves explicitly in the training data set. The simulation results that we present

are analogous to evaluation methods presented in Section 4.5. We refer to this section for more details on the discussion of the evaluation strategies and focus here on interpreting the actual results for the Gumbel-LOB GAN.

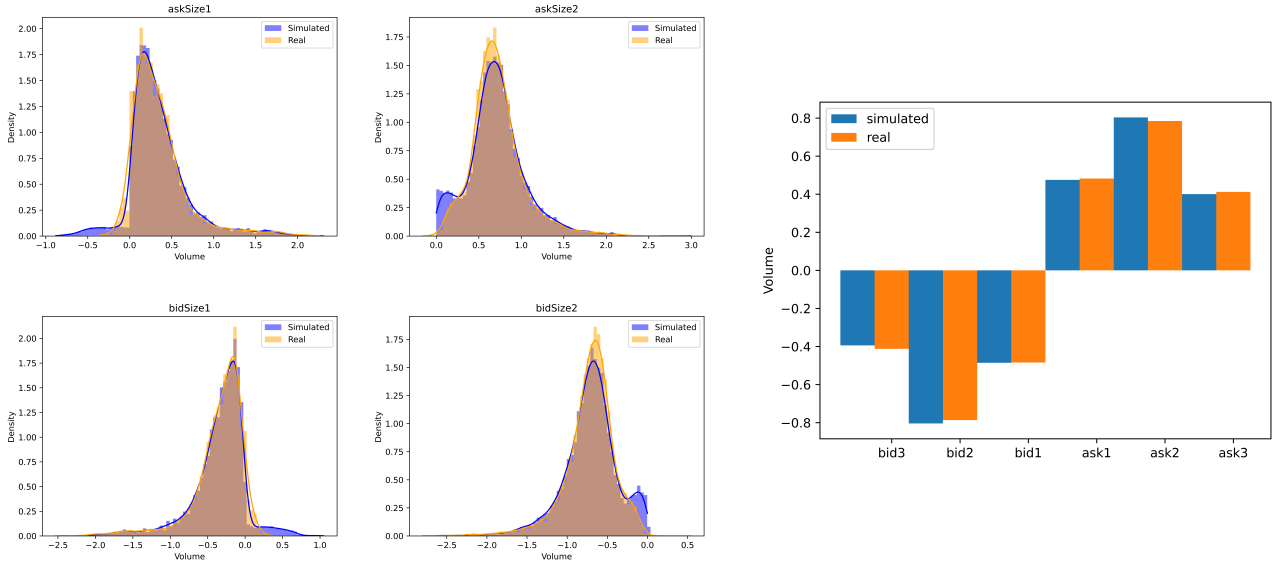


Figure 5.11: Volume comparison: Marginal volumes per order book level after one transition (left); Average LOB shape after one transition (right)

Figure 5.11 corresponds to Figure 4.6 in Section 4.5. In Figure 5.11, we observe that the Gumbel-LOB GAN reproduces the marginal volume distributions relatively well although we see some discrepancies in the tails. Namely, on the best ask and bid level, if a price move occurs the new volume on this level tends to be slightly larger compared to the real marginals. For the second best ask, the simulation produces more mass at small volumes. The average LOB volumes have a very similar shape with only minor deviations from the test data.

Figure 5.12 corresponds to Figure 4.7 in Section 4.5. In the first plot in Figure 5.12, we observe that the Gumbel-LOB GAN is effective in capturing the correlation structure for volume levels and only shows minor differences. The second plot in Figure 5.12 reveals an overall good fit in the correlation of volume changes pointing to the effectiveness in conditional sampling of the Gumbel-LOB GAN. For a few volume levels, we see some slightly larger differences in correlations, e.g. ask2-bid2.

The first plot in Figure 5.13 corresponds to Figure 4.8 in Section 4.5. The mid-price return series shows that the central quantiles of the simulation fit well with the central quantiles of the real returns time series. Even the tails seem too match reasonably well. The remaining plots in Figure 5.13 show histograms of fill ratios for the best ask level for the Gumbel-LOB GAN simulator and the real market as defined in Section 5.3.1 for 10^4 transitions. The top plot includes all the fill ratios for all transitions while for the bottom plot we excluded fill ratios smaller than 0.2 in order to better see the behavior of larger fill ratios. The first plot reveals that our simulator is quite good in capturing the large ratio of transitions with zero fills. However, we also observe that the GAN proposes more small positive fill ratios compared to the real market instead of setting them to zero. In the bottom plot, fill ratios match relatively well for ratios below one. While the simulator shows some accumulating behavior at a fill ratio of one, the accumulation is significantly smaller compared to the real market.

Overall, we showed that the Gumbel-LOB GAN is capable of extracting the main features of LOB transitions in a pro-rata market and provides a good basis for analysing the execution problem described in Section 5.3.1. To extract actionable trading strategies in the actual underlying market, we believe that further effort should be

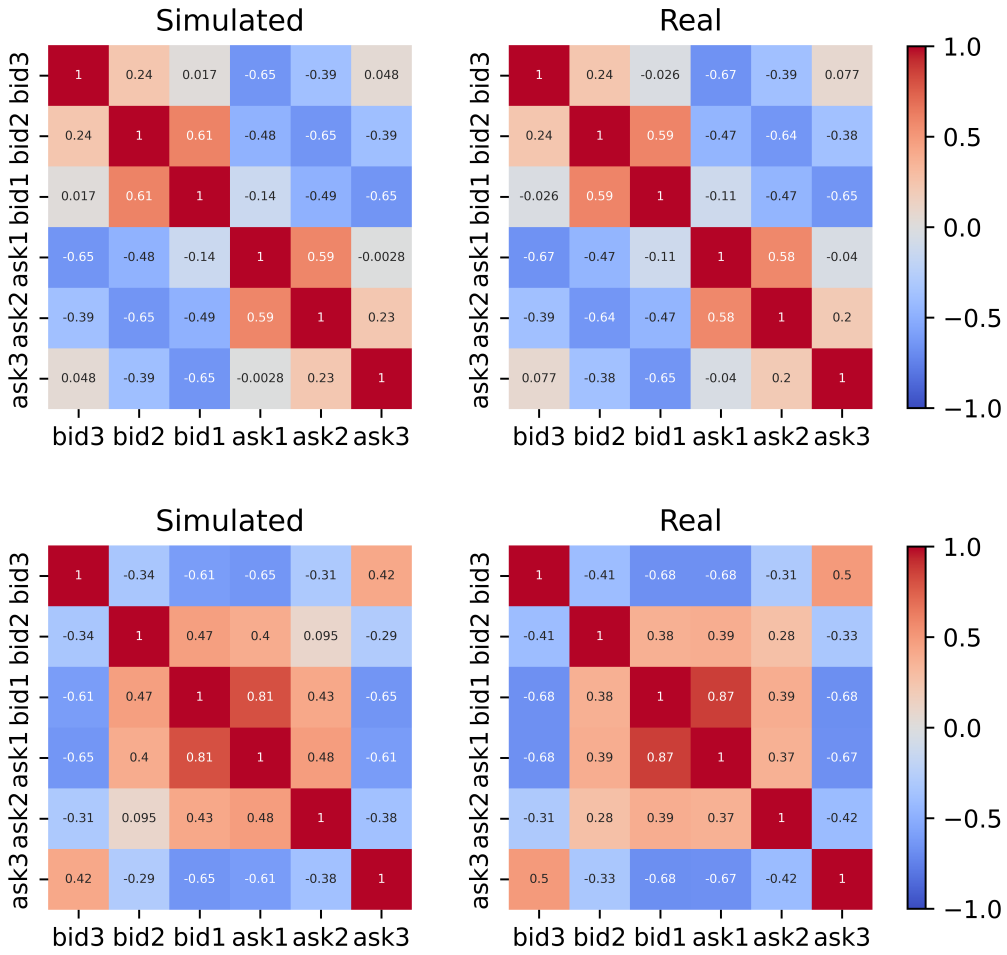


Figure 5.12: Correlation: Absolute volumes (first row); Volume differences (second row)

put in calibrating the Gumbel-LOB GAN to the market data as for example the discrepancy in fill ratios may understate the risks of over-quoting. We leave these improvements in calibration of the Gumbel-LOB GAN or any other differentiable LOB simulator for future work.

5.3.3 Algorithmic considerations for policy gradients

With the previous work of training the Gumbel-LOB GAN as a LOB transition function that is sufficiently differentiable in the control parameter, we can now simulate LOB paths and differentiate with respect to the control parameters along the paths using standard automatic differentiation libraries such as [169]. To approximate the objective for optimal execution, we consider the empirical version along simulated paths, i.e.

$$\widehat{H}(\theta) = \sum_{j=1}^{N} \left(\sum_{i=1}^{n} X^{\theta}_{i\Delta t,j} + Q^{\theta}_{T,j} \left(p^{*}_{T,j} - \alpha Q^{k}_{T,j} \right) \right)$$

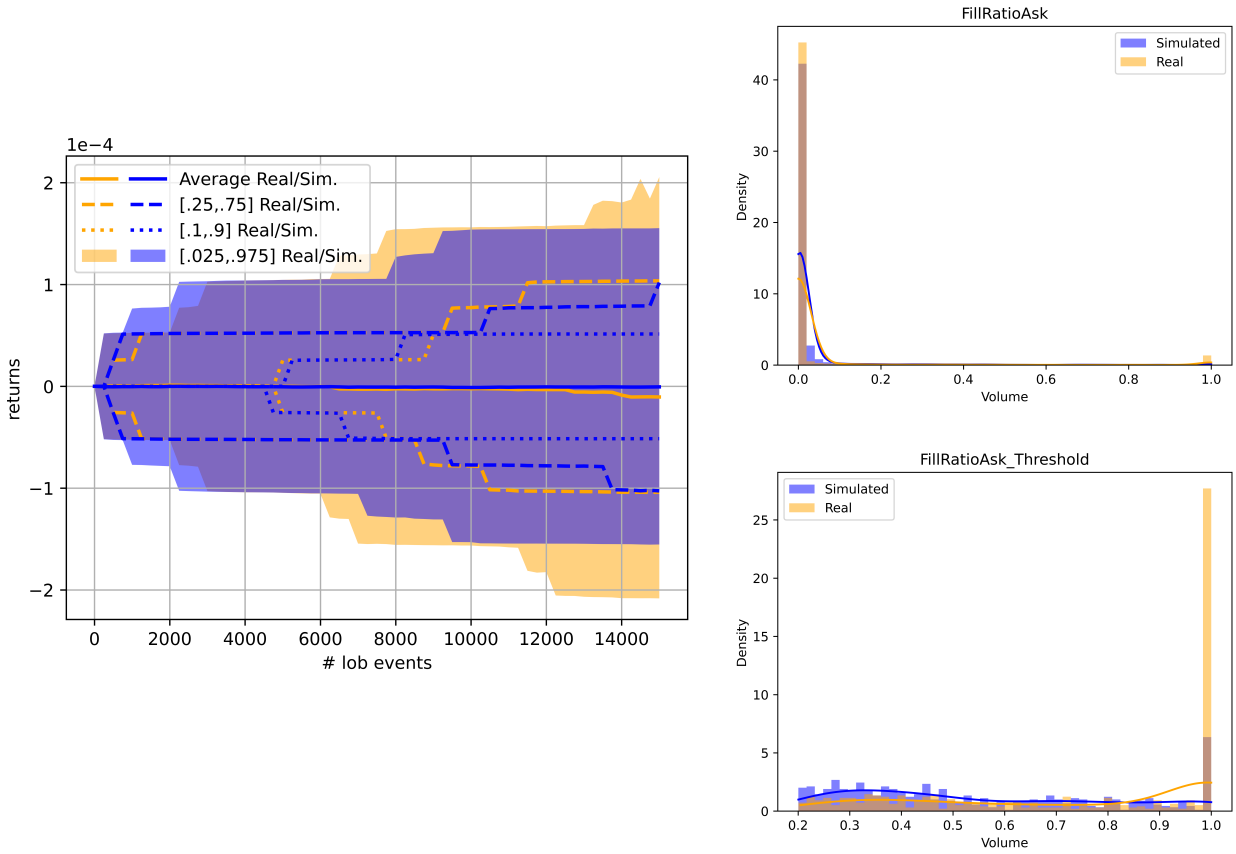


Figure 5.13: On the left, comparison of real mid-price return time series with simulations by Gumbel-LOB GAN; On the right, histogram of fill ratios for the best ask level including all transitions (top) and the transitions with fill ratios large than 0.2 (bottom)

where $p^*_{0,j} = p^*$ and $Q^\delta_{0,j} = \mathfrak{N}$ for all $j = 1, \dots, N$. For each j, the processes $(X^\theta_{i\Delta t,j}, Q^\theta_{i\Delta t,j}, p_{i\Delta t,j})$ are simulated using the Gumbel-LOB GAN transition f_{GAN} where the latent noise inputs are iid. sampled for each transition and the initial LOB snapshot is sampled from the data set. Furthermore, with automatic differentiation, we can calculate the gradient

$$\nabla_\theta \left(\sum_{i=1}^{n} X^\theta_{i\Delta t,j} + Q^\theta_{T,j} \left(p^*_{T,j} - \alpha Q^k_{T,j} \right) \right) \tag{5.2}$$

for each j and, thus, also the gradient $\nabla_\theta \widehat{H}(\theta)$.

Remark 12. *1. In the previous section, we were required to use finite difference approximations to calculate derivatives since the objective was not easily differentiable with respect to the control in a path-wise sense. The usage of a neural network as a transition function easily overcomes this issue since neural networks are usually required to be sufficiently differentiable for their own training. Exploiting this fact allows us not to concern ourselves with issues of using finite difference estimators such as the trade-off for choosing the parameter for the size of the difference.*

2. *Note that usage of neural networks for the transitions generally imply that the objective function is not concave in the control parameters and may exhibit multiple local maxima and saddle points.*

To improve the numerical results, we modify the standard policy gradient algorithm as described in Section 2.4.4. With a standard policy gradient approach, we observe a strong dependence of the optimization results on initialization of the control parameter especially for small α. We propose two modifications to overcomer this issue, which we will describe in more detail below: First, as in the previous section, we employ a backward stepping scheme. However, after a first run of the backward stepping scheme, we run additional global optimization steps. Second, as an initial optimization step in the backward stepping scheme we employ a quasi-Newton method before using standard stochastic gradient descent for the subsequent optimization steps. Another numerical issue that we observe is that parameter θ fluctuates strongly over time and optimization may even become unstable. To limit this behavior, we include a regularization term on the curvature of θ in time.

As in the previous section, the backward stepping scheme implies that we start to optimize the parameter θ_T. In order to do so, we need to initialize the LOB simulator for one transition. While for the price and LOB snapshot initialization is as described above, the initial inventory is drawn for each simulated path from a uniform distribution over $[-100, 100]$. We do this to ensure enough variation in the inventory leading to more robust optimization results. Once θ_T is sufficiently optimized, we consider $\theta_{T-\Delta t}$ and optimize this parameter while keeping θ_T fixed. To initialize the optimization, we set $\theta_{T-\Delta t} = \theta_T$. We repeat this procedure until the start of the time-horizon is reached. After the backward stepping optimization has concluded, we apply standard stochastic gradient descent globally for several iterations. In contrast to the previous section, this is needed since the state space is more complex and future LOB states of the simulations depend on previous quoting decisions. This means, that in the backward stepping optimization, we cannot accurately sample the distribution of starting states for later times as they would depend on previous trading decisions. In other words, the backward stepping optimization attempts to solve an inexact dynamic programming problem to approximate the actual dynamic programming problem. The global optimization is used to overcome the artefacts of the inexact greedy algorithm and improve trading strategies.

Additionally, to overcome the dependence on initialization, we use a quasi-Newton method as an initial gradient step in the backward stepping scheme before using stochastic gradient descent for the following steps and the global optimization. Namely, we use the L-BFGS algorithm as introduced in [139]. The motivation for this modification is based on the analysis of the optimization landscape in Figure 5.14. In the first graph of Figure 5.14, we plot the value of the (unregularized) objective function with $\alpha = 0.001$, $N = 10^5$ and $t = 25$ as a function of the parameter θ_{25}, which shows a relatively smooth and seemingly quadratic objective. Zooming in on the optimization landscape in the second plot reveals that the objective function for θ_{25} has many local minima and saddle points and is highly irregular. The third plot in Figure 5.14 shows the value of the objective function for the same hyper-parameters as the previous plots but in dependence of θ_0 instead of θ_{25}. For θ_0, the unregularized objective seems to be rougher most likely due to the accumulation of noise by the chaining of transitions even on a more macroscopic level. However, the objective has still some resemblance to a convex function but less compared to the objective in dependence on θ_{25}. A quasi-Newton method such as L-BFGS implicitly assumes a quadratic approximation to the objective. Overall, we select L-BFGS since the method often chooses a good Hessian that allows us to escape bad local minima and reach superior parameter regions. Repetitions of stochastic gradient ascent are then used to optimize within the region.

To address the deterioration of the objective function for earlier θ and to improve stability of the optimization, we impose a regularization on the curvature of θ. In the backward optimizing phase, for the optimization of $\theta_{k\Delta t}$ where $k \in \mathbb{N}$ and $(k+2)\Delta t \leq T$, we add the term $\lambda(\theta_{k\Delta t} - 2\theta_{(k+1)\Delta t} + \theta_{(k+2)\Delta t})^2$ to the objective function where $\lambda > 0$ is a regularization parameter and, for $\theta_{T-\Delta t}$, we add $\lambda(\theta_{T-\Delta t} - \theta_T)^2$. For the global optimization, we add the term

$$\lambda \sum_{i=0}^{n-2} (\theta_{i\Delta t} - 2\theta_{(i+1)\Delta t} + \theta_{(i+2)\Delta t})^2$$

to the empirical loss function for regularization. We note that regularizations such as this are common in

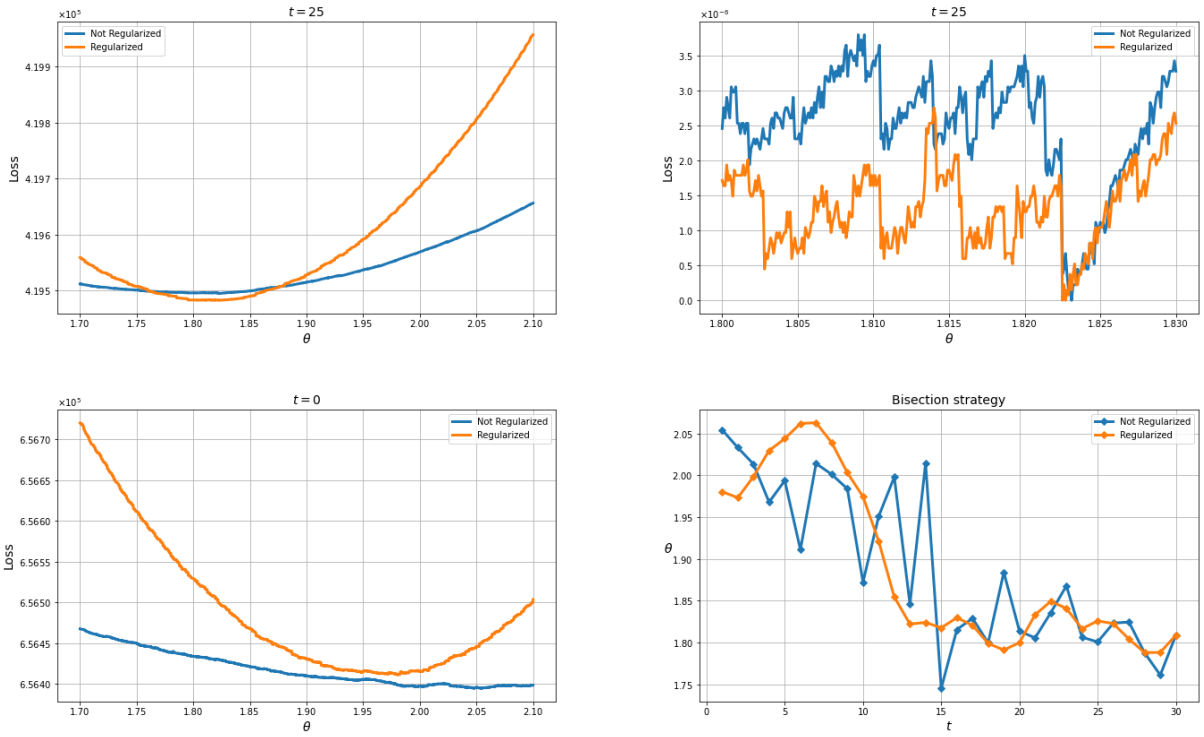


Figure 5.14: Top row: Loss landscape at $t = 25$ (left) Zoomed-in normalized loss landscape at $t = 25$ (right). Bottom row: Loss landscape at $t = 0$ (left) Strategy obtained by bisection for regularized and not regularized loss (right).

non-parametric curve fitting, see e.g. [96]. Furthermore, the top left and bottom left plot in Figure 5.14 reveal that regularization is effective in making the objective more convex enabling an improved performance of the L-BFGS optimization step. The bottom right plot of Figure 5.14 also shows that regularization smoothens the optimal policy over time for the backward stepping scheme getting rid of large fluctuations in strategy. However, zooming in on the loss landscape in the top right plot reveals that even the regularized loss is rough on a microscopic scale.

Remark 13. *We obtained the optimal strategies in the bottom right plot of Figure 5.14 using a bisection method during the backward stepping procedure. However, this method requires prior knowledge on the region of the optimal strategy and the scale of the objective function. Without such knowledge, bisection may be ineffective for solving the backward stepping optimization. In this case, we leveraged results obtained from our policy gradient approach to identify the region and resolution of the bisection method. Furthermore, bisection cannot be used for the global optimization due to the curse of dimensionality.*

5.3.4 Results

Figure 5.15 shows the resulting execution strategies of the optimization procedure explained in the previous section for different market order impact parameters α for a time horizon of 30 LOB transitions. As regularization parameter, we choose $\lambda = 0.001$ and we use 50 stochastic gradient steps for within the backward stepping scheme and the global optimization. For each gradient calculation we use 4×10^4 sample paths. As learning rate, we

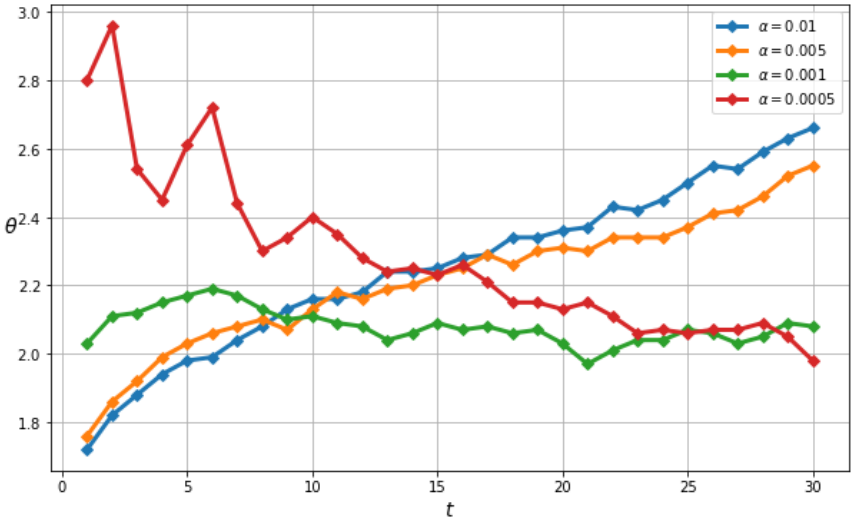


Figure 5.15: Strategy obtained by policy gradient method for different values of α.

choose 0.1 for L-BFGS steps and 10^{-5} for stochastic gradient steps. We will use these hyperparameters unless mentioned otherwise. We see that for all choices of α the strategy parameter θ is larger than 1 for all t. This means that the learned strategies suggests overquoting the actual execution target, which is consistent with the results in Chapter 4 and related literature, e.g. [59, 83].

Furthermore, the policy gradient method reveals a dependence of the functional form of the strategy in time on the cost of liquidating assets via market orders. Figure 5.15 shows that larger penalizations of final inventory are associated with an increasing order multiplier. Furthermore, the steepness of the slope is increasing with the size of α. Financially, this strategy implies that the trading agent starts out with attempting to liquidate their position slowly. However, as time progresses, the agent will place relatively more aggressive limit orders to execute their remaining position in order to avoid having to pay the final inventory penalty. By doing so, the agent trades off the additional risk of overfilling with the costs associated of liquidating their portfolio via market orders. Figure 5.16 shows the quantiles and averages of inventories over time for a strategy learned with $\alpha = 0.01$ and $\alpha = 0.0005$ for 10^5 paths. Overall, for $\alpha = 0.01$, the average inventory vanishes fast in the start mainly driven by a left tailed distribution of inventories. However, after some time the average inventory curve flattens and the mass of distribution concentrates more around the median which remains slightly above 0 at the final point in time.

For a decreasing penalization of final inventory, Figure 5.15 shows that the order multiplier further flattens in time and even inverts to a decreasing function for very small penalties. This means that for small costs of market orders, the agent vastly overquotes their inventory to trade out of their position faster. This can be seen in Figure 5.16 where for both quantiles and average the inventory initially decreases faster for strategies learned with $\alpha = 0.0005$ compared to strategies with $\alpha = 0.01$. As a drawback, the agent with low final inventory costs accepts a higher risk of initially being overfilled. To counteract the wider distribution of inventories and concentrate it around the median, the agent decreases the order multiplier. Compared to the strategy produced by $\alpha = 0.01$, the slightly wider distribution seems to be acceptable to reach a slightly lower average inventory for $\alpha = 0.0005$ at the final point in time.

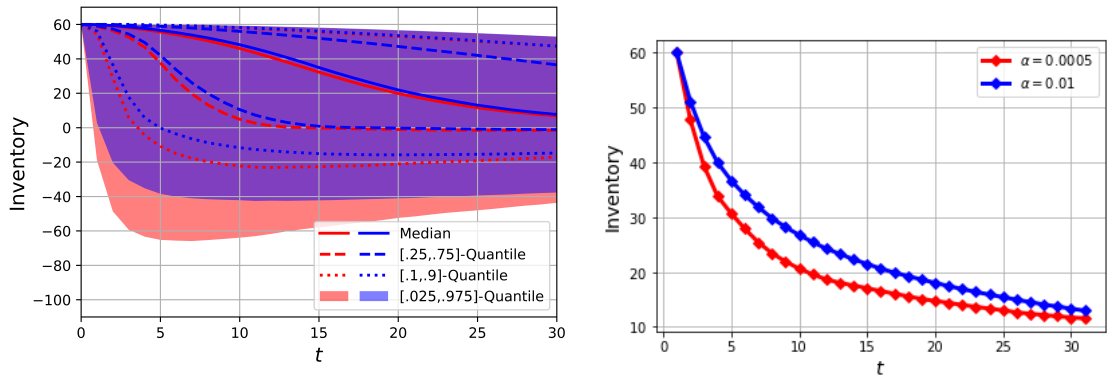


Figure 5.16: On the left, comparison of inventory quantiles for $\alpha = 0.0005$ (red) and $\alpha = 0.01$ (blue); On the right, comparison of average inventory for $\alpha = 0.0005$ (red) and $\alpha = 0.01$ (blue)

Regularization and global optimization Lastly, we further discuss some numerical aspects in the optimization procedure. As mentioned before, we use regularization with respect to the curvature of the parameters to smooth the learned strategy in time. The first plot in Figure 5.17 shows learned strategies for different regularization for $\alpha = 0.001$ and using global optimization. For this parametrization, we see that without regularization ($\lambda = 0$) the learned strategy is relatively jagged across time. Using regularization smooths the strategy curve but may lead to outliers at the start of the time frame. In practice, the level of regularization may be chosen in such a way that the learned strategy leads to the highest (unregularized) empirical trading profits among all regularizations. Considering the third plot in Figure 5.17 reveals that with this chosen parametrization the profits from trade are maximal if $\lambda = 0$. For the previous results, we still decide to use regularization as for some choices of α the policy gradient method may diverge. We apply the same regularization for all α to ensure comparability.

In the second plot in Figure 5.17, we show that adding global optimization steps after the backward stepping scheme has a qualitative impact on the learned strategy. The graph contains the strategy obtained after only using the backward stepping scheme and the strategy learned when combining backward stepping and ex-post global optimization. Both strategies are learned without regularization. At the beginning of the time horizon both strategies almost coincide. However, while the strategy without the global optimization steps decreases, the strategy with global optimization remains relatively constant. Comparing the profits of trade per asset in the third plot in Figure 5.17 reveals that this qualitative change in strategy due to global optimization leads to an improved outcome for all choices of regularization. Furthermore, the learned strategy with global optimization also outperforms the strategy learned with a bisection method and backward stepping (cf. bottom left plot in Figure 5.14).

5.4 Conclusion

This chapter introduces two applications of policy gradient methods to the problem of asset execution using only limit orders. We overcome issues with differentiability by either using zeroth order approximations for gradients or introducing a differentiable GAN simulator for LOBs and using path-wise gradient estimates from this simulator. In both cases, the learned strategies are effective in solving the asset execution problem. Furthermore, in the second part, we introduce a LOB simulator trained with actual LOB data adding realism. We further believe that there are several extensions that may be of interest. First, improvements in the simulator leading to an even better calibration to the data may further enhance realism and make learned strategies applicable within markets.

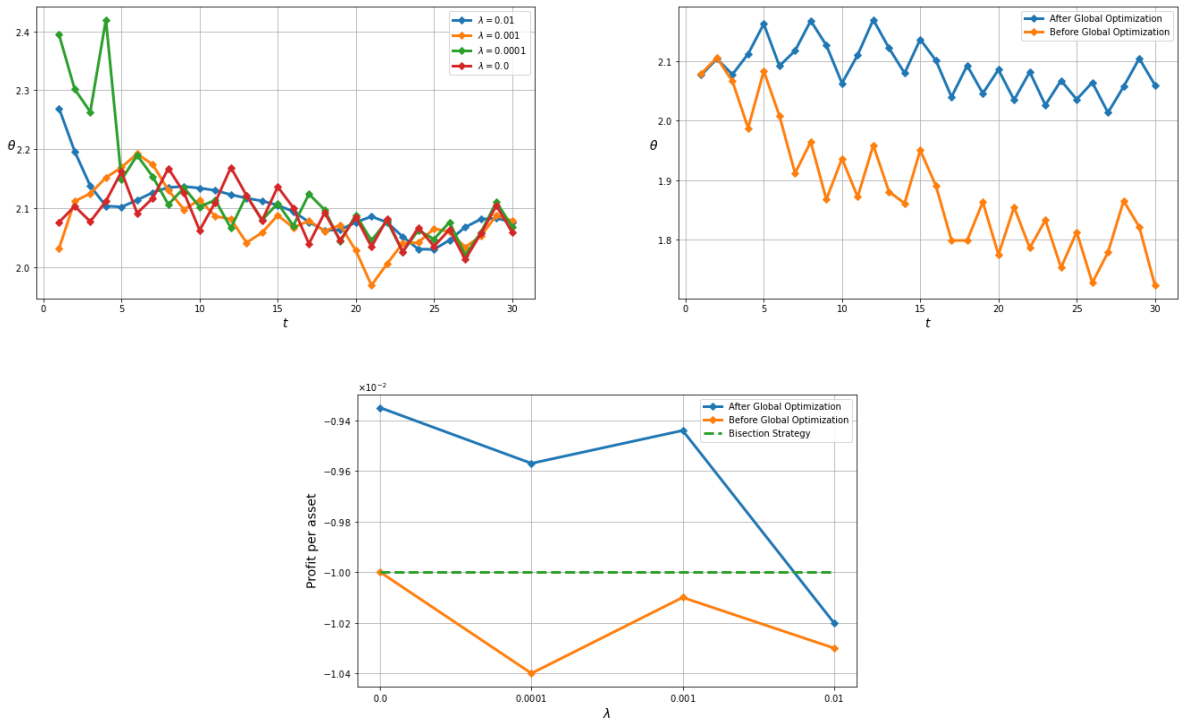


Figure 5.17: Top row: Strategies obtained by policy gradient method for different values of λ for $\alpha = 0.001$ and strategy with and without global optimization step using $\alpha = 0.001$ and $\lambda = 0$ (left); Bottom row: Profit per asset for strategies with different regularization with and without global optimization.

For example, such improvements may be achievable by exploring other generative neural network architectures such as diffusion models and considering different training approaches. Furthermore, introducing differentiable LOB simulators allowing for evaluations of limit orders using other LOB mechanisms such as first-in-first-out may extend the applicability to other markets such as equities. Second, more complex parametrizations of strategies (i.e. neural networks) may allow inclusion of trading signals for execution strategies. However, such parametrizations pose a difficult challenge for optimization with either zeroth-order methods or GAN simulators. From a financial perspective, it would also be of interest to include controls for market orders and investigate the related market making problem.

Chapter 6

Policy Gradient Methods for Continuous-time Linear Quadratic Control

6.1 Introduction

As described in Chapter 2, policy gradient methods have become a widely used tool to find good policies for a variety of stochastic control problems. Using a parametrization of the policy in the current state of the system, the policy gradient method exploits the gradient of the objective in the direction of the policy parameters to update these parameters. Recently, the theoretical analysis of policy gradients has gained considerable attention with a main focus on discrete-time systems described by Markov decision processes (MDPs)(see e.g., [57, 90, 119, 148, 225]). However, the dynamical systems underlying real-world control problems are often seen as being set in continuous time. One example is the trading in limit order books where the emergence of modern electronic markets allowed interactions at the frequency of milliseconds (e.g. [39]). Thus, explaining the need to investigate learning techniques such as policy gradient methods in continuous time settings.

In this chapter, we consider a stochastic finite-horizon linear-quadratic control (LQC) model in continuous time. In a LQC problem, the aim is to minimize a quadratic cost function of the state and the control subject to a stochastic differential equation linear in the state and the control. As explained in Chapter 2, LQC problems are often seen as an approximation of more complex non-linear systems and are used in applications such as portfolio optimisation [216, 228] or algorithmic trading [40]. Furthermore, LQC problems are analytically tractable and optimal policies can written as linear functions of the current state where the coefficients may depend on time. From the perspective of a policy gradient method it is thus sufficient to parametrize the policy by a time-dependent function linear in the state and optimize over the coefficients. Thus, in analogue to discrete-time LQC problems (e.g. [90, 179]), continuous-time LQC problems are an important model class to understand theoretical behaviour of learning approaches to control and evaluate their practical effectiveness.

A priori, it is unclear how the algorithm and results from discrete control systems relate to the time-continuous case. Furthermore, applying a time-discrete learning algorithm to a time-continuous control problem poses the question how robust the algorithm is with respect to the chosen discretization. Indeed, in numerical experiments in [154, 167], the authors find that an increased action frequency is related to worse learning outcomes for policy gradient methods. We reproduce this observation in our numerical experiments (see Section 6.5). Recently, there has been an effort to overcome these issues and design algorithms that are robust with respect to different action frequencies (e.g. [112, 113, 206]). However, this literature does not provide theoretical convergence guarantees.

The theoretical analysis of policy gradient methods in LQC problems is difficult even in discrete-time since

the control objective is usually not convex in the parametrization of the policy (e.g. [57, 225]). This is the case as well in the time-continuous LQC problem (see Section 6.4, Proposition 6). It is well known in optimization theory that proving convergence of gradient-based algorithm to a global optimum is particularly challenging for non-convex optimization problems. Additionally, considering a finite horizon LQC problem in a continuous time setting implies that optimal controls are time-dependent. Thus, the parametrization of the optimal solution to the LQC problem is function-valued and hence an optimization over an infinite dimensional parameter space is required. This is in contrast to the time-discrete case, where the optimization is over a finite dimensional space, where the dimension is given by the number of time-steps. Considering an infinite-dimensional optimization instead of a finite one, leads to significant differences in the loss-landscape. Namely, in the time-continuous case, the cost function is *noncoercive* meaning the sublevel sets of the cost function may not be bounded (see Section 6.4, Proposition 6). In the time-discrete case, the cost function has bounded sublevel sets (or is coercive) which implies the stability of the optimization along the iterates for sufficiently small step sizes. The lack of coercivity in the time-continuous LQC problem implies that iterates of gradient-based algorithm may not remain bounded and hence a more careful analysis of the loss-landscape is required.

In this chapter, we introduce a finite-horizon exploratory LQC problem, a generalisation of the standard time-continuous LQC that allows for entropy regularization in the cost function (see [215]). We then formulate the exploratory LQC problem as an optimization problem over Gaussian policies parametrized by the time-dependent functions (K, V). The mean of the Gaussian policy at a point in time is given as a linear function of the state where the coefficient is given by K and the covariance by V. We then use the Pontryagin optimality principle to describe the gradient of the cost function in the direction of the policy parameters and using this characterization we show that the control objective satisfies a non-uniform Łojasiewicz condition (Proposition 4) and smoothness condition (Proposition 5). These conditions on the loss-landscape are then sufficient to prove the linear convergence of a geometry-aware policy gradient descent method (see Section 6.4 Theorem 4). Furthermore, we conduct two numerical experiments for an exploratory LQC problem arising from mean-variance portfolio construction: First, we give evidence for the linear convergence of the policy gradient method. Second, we investigate how the discretization of policies in time impact the policy gradient method. We find that the number of optimization steps necessary to reach a certain level of loss increases linearly with the action frequency and show that this issue can be overcome by linearly rescaling the time-discrete gradients by the action frequency. Indeed, we then observe that the number of steps required remains constant in action frequency which implies robustness with respect to action frequency for the policy gradient method. This second experiment confirms another theoretical result from [71] which postulates the robustness with respect to action frequency for a rescaled policy gradient method (see Section 6.4 Theorem 5).

The results presented in this chapter correspond to the author's contribution to [71]. For completeness we will state additional results from [71] without proof (see Section 6.4). We organize this chapter as follows: We will start by discussing related literature and notation. The next section will contain a description of the exploratory LQC problem. After that, we will introduce the loss-landscape analysis without proofs, before providing the additional results from our co-authors. Then, we will discuss the numerical experiments and conclude with the proofs for the loss-landscape analysis.

6.1.1 Related work

The previous literature on theoretical properties of policy gradient methods for LQC problems mostly focused on settings with stationary policies for infinite time horizons and deterministic dynamics (see e.g., [32, 57]). For the stochastic case, studies consider discrete-time problems with additive noise and either finite (e.g. [90]) or infinite time-horizon (e.g. [115, 225]). [79] discusses multiplicative noise for infinite horzions. Policy gradient methods for linear quadratic games were presented in [88, 217, 226]. Note that in all the previous settings a finite dimensional parametrization of the policy was considered.

In contrast to previous work, we need to analyse the loss-landscape of the optimization problem over time-continuous policies in an infinite dimensional space. Thus, the analysis requires the usage of continuous-time

control theory. For example, instead of using algebraic equations as in the discrete-time setting, we require Lyapunov differential equations to describe the gradient. Conditions on the optimization objective we derive require the usage of techniques from partial differential equations. Secondly, we allow for degeneracies of matrices in the cost function and, thus, require sharper estimates for the regularity conditions of the costs. Previous works consider a running cost of the form $x^\top Qx + a^\top Ra$ where Q and R are positive definite matrices and use the existence of positive eigenvalues for their loss-landscape analysis. In applications of stochastic LQC problems, one often encounters costs that include a product of state and control [40] or with an indefinite matrix R [216, 228]. Using the solutions of Lyapunov equations, we derive tighter bounds on the conditions for the costs. With this, we can consider more general settings where both drift and diffusion coefficients of the state process are controlled and all matrices in the running costs may be negative definite.

6.1.2 Notation

By $\langle \cdot, \cdot \rangle$ we denote the standard inner product for an Euclidean space E, and by $|\cdot|$ the norm induced by $\langle \cdot, \cdot \rangle$. For a matrix $A \in \mathbb{R}^{n\times m}$, we write $A^\top$ for the transpose, $\operatorname{tr}(A)$ for the trace, and $\|A\|_2$ for the spectral norm. For $n \in \mathbb{N}$, I_n is the $n \times n$ identity matrix. $\mathbb{S}^n$, $\overline{\mathbb{S}^n_+}$ and $\mathbb{S}^n_+$ are the spaces of symmetric, symmetric positive semidefinite, and symmetric positive definite matrices, respectively. $\lambda_{\max}(A)$ and $\lambda_{\min}(A)$ is the largest and smallest eigenvalues of $A \in \mathbb{S}^n$. For each $A, B \in \mathbb{S}^n$, we may write $A \succeq B$ if $A - B \in \overline{\mathbb{S}^n_+}$. This equips $\mathbb{S}^n$ with the Loewner (partial) order. For measurable functions $F, G : [0,T] \to \mathbb{S}^n$, we write $F \succeq G$ if $F(t) - G(t) \in \overline{\mathbb{S}^n_+}$ for a.e. $t \in [0,T]$.

For any $T > 0$, let $(\Omega, \mathcal{F}, \mathbb{F}, \mathbb{P})$ be a filtered probability space satisfying right continuity and completeness. For a Euclidean space $(E, |\cdot|)$, $\mathcal{B}(0,T;E)$ denotes the space of Borel measurable functions $\phi : [0,T] \to E$. $L^p(0,T;E) \subseteq \mathcal{B}(0,T;E)$ with $p \in [1,\infty]$ is the space of functions $\phi : [0,T] \to E$ satisfying $\|\phi\|_{L^p} = (\int_0^T |\phi_t|^p \,\mathrm{d}t)^{1/p} < \infty$ if $p \in [1,\infty)$ and $\|\phi\|_{L^\infty} = \operatorname{ess\,sup}_{t\in[0,T]} |\phi_t| < \infty$. $C([0,T];E)$ is the space of continuous functions with the norm $\|\cdot\|_{L^\infty}$. $\mathcal{S}^2(0,T;E)$ denotes the space of $\mathbb{F}$-progressively measurable càdlàg processes $X : \Omega \times [0,T] \to E$ where $\|X\|_{\mathcal{S}^2} = \mathbb{E}[\operatorname{ess\,sup}_{t\in[0,T]} |X_t|^2]^{1/2} < \infty$. $\mathcal{M}(E)$ is the set of measures on E, $\mathcal{P}(E)$ the set of probability measures, and $\mathcal{P}_2(E)$ the set of square integrable probability measures endowed with the 2–Wasserstein distance. For each $\mu \in \mathbb{R}^n$ and $\Sigma \in \overline{\mathbb{S}^n_+}$, $\mathcal{N}(\mu, \Sigma)$ denotes the Gaussian measure on $\mathbb{R}^n$ with mean μ and covariance matrix Σ. For simplicity, we may write $\mathbb{N}_0 = \mathbb{N} \cup \{0\}$.

6.2 Regularised stochastic LQ control problems with indefinite costs

At this point, we introduce the regularised LQC problem and recall the optimal feedback controls for them. For $T > 0$, consider a d-dimensional standard Brownian motion $W = (W_t)_{t\geq 0}$ on a probability space $(\Omega, \mathcal{F}, \mathbb{P})$ where $\mathbb{F} = (\mathcal{F}_t)_{t\geq 0}$ is the natural filtration of W augmented by an independent σ-algebra $\mathcal{F}_0$. $\xi_0 \in L^2(\Omega; \mathbb{R}^d)$ is a $\mathcal{F}_0$-measurable random variable, and $\mathcal{A}$ denotes the set of (relaxed) controls $\mathfrak{m} : \Omega \to \mathcal{M}([0,T] \times \mathbb{R}^k)$ such that $\mathfrak{m}_t(\mathrm{d}t, \mathrm{d}a) = \mathfrak{m}_t(\mathrm{d}a)\mathrm{d}t$ for a.e. $t \in [0,T]$, where $\mathfrak{m}_t : \Omega \to \mathcal{P}(\mathbb{R}^k)$ is $\mathcal{F}_t$-measurable for all $t \in [0,T]$ and $\mathbb{E}[\int_0^T \int_{\mathbb{R}^k} |a|^2 \mathfrak{m}_t(\mathrm{d}a)\mathrm{d}t] < \infty$. For any $\mathfrak{m} \in \mathcal{A}$, the dynamics are given by

$$\mathrm{d}X_t = \Phi_t(X_t, \mathfrak{m}_t)\,\mathrm{d}t + \Gamma_t(X_t, \mathfrak{m}_t)\,\mathrm{d}W_t, \quad t \in [0,T]; \quad X_0 = \xi_0. \tag{6.1}$$

The functions $\Phi : [0,T] \times \mathbb{R}^d \times \mathcal{P}_2(\mathbb{R}^k) \to \mathbb{R}^d$ and $\Gamma : [0,T] \times \mathbb{R}^d \times \mathcal{P}_2(\mathbb{R}^k) \to \overline{\mathbb{S}^d_+}$ can be written for all $(t,x,m) \in [0,T] \times \mathbb{R}^d \times \mathcal{P}_2(\mathbb{R}^k)$, as

$$\Phi_t(x,m) = \int_{\mathbb{R}^k} (A_t x + B_t a)\, m(\mathrm{d}a), \quad \Gamma_t(x,m) = \left(\int_{\mathbb{R}^k} (C_t x + D_t a)(C_t x + D_t a)^\top m(\mathrm{d}a) \right)^{\frac{1}{2}}. \tag{6.2}$$

The operator $(\cdot)^{\frac{1}{2}} : \overline{\mathbb{S}^d_+} \to \overline{\mathbb{S}^d_+}$ denotes the matrix square root such that $M^{\frac{1}{2}}(M^{\frac{1}{2}})^\top = M$ for all $M \in \overline{\mathbb{S}^d_+}$, and A, B, C, D are measurable functions such that (6.1) has a unique and strong solution $X^{\mathfrak{m}} \in \mathcal{S}^2(0,T;\mathbb{R}^d)$. The precise conditions on these functions are given in (H.3).

We can introduce the objective for the control problem. For this consider $\rho \geq 0$ and $\overline{\mathfrak{m}} : [0,T] \to \mathcal{M}(\mathbb{R}^k)$ a measurable function. Then, we would like to minimise the cost functional

$$\mathbb{E}\Big[\int_0^T \int_{\mathbb{R}^k} \Big(\frac{1}{2}\Big\langle \begin{pmatrix} Q_t & S_t^\top \\ S_t & R_t \end{pmatrix} \begin{pmatrix} X_t^{\mathfrak{m}} \\ a \end{pmatrix}, \begin{pmatrix} X_t^{\mathfrak{m}} \\ a \end{pmatrix} \Big\rangle \mathfrak{m}_t(\mathrm{d}a) + \rho \mathcal{H}(\mathfrak{m}_t \| \overline{\mathfrak{m}}_t)\Big)\,\mathrm{d}t + \frac{1}{2}(X_T^{\mathfrak{m}})^\top G X_T^{\mathfrak{m}}\Big], \tag{6.3}$$

over the set of all relaxed controls $\mathfrak{m} \in \mathcal{A}$, where Q, S, R are matrix-valued functions and $G \in \mathbb{R}^{d\times d}$ parametrizes the terminal costs. Furthermore, for all $t \in [0,T]$, $\mathcal{H}(\cdot\|\overline{\mathfrak{m}}_t) : \mathcal{P}(\mathbb{R}^k) \to [0,\infty]$ denotes the relative entropy comparing controls to $\overline{\mathfrak{m}}_t$ and is given by

$$\mathcal{H}(m\|\overline{\mathfrak{m}}_t) = \begin{cases} \int_{\mathbb{R}} \ln\big(\frac{m(\mathrm{d}a)}{\overline{\mathfrak{m}}_t(\mathrm{d}a)}\big)\, m(\mathrm{d}a), & m \text{ is absolutely continuous with respect to } \overline{\mathfrak{m}}_t, \\ \infty, & \text{otherwise.} \end{cases}$$

The control problem defined by (6.1)-(6.3) is known as an exploratory stochastic LQC problem and was, for example, discussed in [60, 196, 215, 216]. In this context, the relaxed controls given by measures are used to model randomised interaction of the controller with the system and the entropy term $\rho\mathcal{H}(\cdot\|\overline{\mathfrak{m}}_t)$ is added for regularization. A more thorough discussion on how exploratory and randomised policies relate can be found in [205] and in Section 6.5. Note that if we set $\rho = 0$ in (6.3) and restrict $\mathcal{A}$ to Dirac-measure valued controls, i.e. $\mathfrak{m}_t(\mathrm{d}t, \mathrm{d}a) = \boldsymbol{\delta}_{\alpha_t}(\mathrm{d}a)\mathrm{d}t$ for some $\alpha_t : \Omega \times [0,T] \to \mathbb{R}^k$ where $\boldsymbol{\delta}_a$ is the Dirac measure on $a \in \mathbb{R}^k$, the reduced problem corresponds to the ordinary stochastic LQC case.

For the remainder of this Chapter, we will make the following assumptions on the parameters of the control problem:

Assumption 3. *(1) $T > 0$, $\xi_0 \in L^2(\Omega;\mathbb{R}^d)$, $A \in L^1(0,T;\mathbb{R}^{d\times d})$, $B \in L^2(0,T;\mathbb{R}^{d\times k})$, $C \in L^2(0,T;\mathbb{R}^{d\times d})$, $D \in L^\infty(0,T;\mathbb{R}^{d\times k})$, $Q \in L^1(0.T;\mathbb{S}^d)$, $S \in L^2(0.T;\mathbb{R}^{k\times d})$, $R \in L^\infty(0,T;\mathbb{S}^k)$ and $G \in \mathbb{S}^d$.*

(2) $\rho > 0$, $\overline{\mathfrak{m}}_t = \mathcal{N}(0,\bar{V}_t)$ for all $t \in [0,T]$, $\bar{V} \in L^\infty(0,T;\mathbb{S}_+^k)$ and $\bar{V} \succeq \delta I_k$ for some $\delta > 0$.

With the conditions on A, B, C, D in Assumption 3(1) one can show that for all $\mathfrak{m} \in \mathcal{A}$, underlying SDE is uniquely and strongly solvable and that the optimization objective is well-defined [71]. The coefficient functions Q, S, R and G may be indefinite according to Assumption 3(1). Thus, our analysis covers indefinite stochastic LQ problem (see e.g. [200]) that may be used to model optimal liquidation problems [40] or mean-variance portfolio construction [228]. Furthermore, Assumption 3(2) restricts the LQC problem to settings where $\rho > 0$ and the reference measure $\overline{\mathfrak{m}}_t$ in the entropy term is Gaussian with nondegenerate covariance. This assumption allows us to rewrite the control problem as an equivalent minimization problem over nondegenerate Gaussian policies. However, the convergence analysis of the policy gradient algorithm does not exploit the nondegeneracy of the reference measures. Hence, to extend the results to cases where the reference measure is the Lebesgue measure [60, 215, 216] or the standard stochastic LQC problem, only minor adjustments in the loss-landscape analysis are needed. We omit these cases for the sake of brevity.

To ensure that our LQC problem allows a unique optimal feedback control (meaning the LQC problem is closed-loop solvable) we make the following assumption on the solvability of the corresponding Riccati equation:

Assumption 4. *There exists $P^\star \in C([0,T];\mathbb{S}^d)$ satisfing the following Riccati equation: for a.e. $t \in [0,T]$,*

$$\begin{cases} (\frac{\mathrm{d}}{\mathrm{d}t}P)_t + A_t^\top P_t + P_t A_t + C_t^\top P_t C_t + Q_t \\ \quad - (B_t^\top P_t + D_t^\top P_t C_t + S_t)^\top (D_t^\top P_t D_t + R_t + \rho \bar{V}_t^{-1})^{-1}(B_t^\top P_t + D_t^\top P_t C_t + S_t) = 0; \\ P_T = G, \end{cases} \tag{6.4}$$

and $D^\top P^\star D + R + \rho\bar{V}^{-1} \succeq \widetilde{\delta} I_k$ for some $\widetilde{\delta} > 0$.

This assumption corresponds to the strongly regular solvability condition for Riccati equations introduced in [200] and implies the unique existence of an optimal feedback control for the underlying LQC problem. If the unregularized version (i.e. where $\rho = 0$) of (6.4) has a strongly regular solution $P^{\star,0} \in C([0,T];\mathbb{S}^n)$ and $D^\top P^{\star,0} D + R \succeq \widetilde{\delta} I_k$, then also the regularized version with $\rho > 0$ satisfies Assumption 4. This holds since $P^\star \succeq P^{\star,0}$ (see Proof of [200, Theorem 5.3]), and hence $D^\top P^\star D + R + \rho \bar{V}^{-1} \succeq D^\top P^{\star,0} D + R$ by Assumption3(2).

Assumption 4 may still hold for a regularized LQC problem ($\rho > 0$) even if the unregualised version of the LQC problem ($\rho = 0$) does no longer admit a closed loop solution due to the additional regularizing term $\rho \bar{V}^{-1}$. This implies that the entropy term has indeed a regularizing effect on the loss landscape. However, this observation relies on choosing a Gaussian reference measures with non-degenerate covariance. For example, for the Lebesgue measure as reference, this will not be the case as [60, 215, 216] have shown that the loss landscape remains unchanged by entropy regularization and the closed loop solvability of the regularised and unregularised LQC problem is equivalent.

With Assumption 3 and Assumption 4 a standard verification argument (see, e.g., [224]) can be used to show that the optimal control $\mathfrak{m}^\star \in \mathcal{A}$ of (6.1)-(6.3) is given by $\mathfrak{m}^\star_t = \nu^\star_t(X^{\mathfrak{m}^\star}_t)$, where for all $(t,x) \in [0,T] \times \mathbb{R}$, $\nu^\star_t(x) = \mathcal{N}(K^\star_t x, V^\star_t)$ and

$$\begin{aligned} K^\star_t &= -(D_t^\top P^\star_t D_t + R_t + \rho \bar{V}_t^{-1})^{-1}(B_t^\top P^\star_t + D_t^\top P^\star_t C_t + S_t), \\ V^\star_t &= \rho (D_t^\top P^\star_t D_t + R_t + \rho \bar{V}_t^{-1})^{-1}. \end{aligned} \tag{6.5}$$

Furthermore, with Assumption 3 and Assumption 4, we have that $K^\star \in L^2(0,T;\mathbb{R}^{k\times d})$, $V^\star \in L^\infty(0,T;\mathbb{S}^k_+)$ and $V^\star \succeq \varepsilon I_k$ for some $\varepsilon > 0$.

6.3 Optimisation over policy parameters and loss landscape analysis

In this section, we reformulate the LQC problem (6.1)-(6.3) as an equivalent minimisation problem over a parametrized policies where the choice of parametrization is motivated by the shape of the optimal policy $\nu^\star$. Furthermore, we state important characteristics of the loss-landscape of the parametrized optimisation problem. The proofs of these statements can be found in Section 6.6.

6.3.1 Parametrization and parametrized optimisation

We define the parameter space as

$$\Theta \coloneqq \left\{ \theta = (K,V) \in \mathcal{B}(0,T;\mathbb{R}^{k\times d} \times \mathbb{S}^k_+) \;\middle|\; \|K\|_{L^2} < \infty,\ \varepsilon I_k \preceq V \preceq \tfrac{1}{\varepsilon} I_k \text{ for some } \varepsilon > 0 \right\},$$

and the space of Gaussian policies parameterised by Θ by

$$\mathcal{V} \coloneqq \left\{ \nu^\theta : [0,T] \times \mathbb{R}^d \ni (t,x) \mapsto \mathcal{N}(K_t x, V_t) \in \mathcal{P}(\mathbb{R}^k) \;\middle|\; \theta = (K,V) \in \Theta \right\}.^1 \tag{6.6}$$

For simplicity we identify the Gaussian policy $\nu^\theta \in \mathcal{V}$ with its parameter $\theta = (K,V) \in \Theta$. Furthermore, for any $\nu^\theta \in \mathcal{V}$, we introduce the controlled dynamics (cf. (6.1)):

$$\mathrm{d}X_t = \Phi_t(X_t, \nu^\theta_t(X_t))\,\mathrm{d}t + \Gamma_t(X_t, \nu^\theta_t(X_t))\,\mathrm{d}W_t, \quad t \in [0,T]; \quad X_0 = \xi_0, \tag{6.7}$$

where Φ and Γ as in (6.2). This SDE admits a unique solution (see [71]) we denote by $X^\theta \in \mathcal{S}^2(0,T;\mathbb{R}^d)$.

[1] Note that since we assume $\rho > 0$, Gaussian policies in $\mathcal{V}$ need to have nondegenerate covariances. If instead we consider $\rho = 0$, we can choose the admissible policies as $\nu^\theta_t(x) = \mathcal{N}(K_t x, 0)$. As mentioned before our results and analysis naturally extend to this case.

Given the parametrization and the parametrized dynamics we can state the corresponding cost functional:

$$\begin{aligned}\mathcal{C}(\theta) \coloneqq \mathbb{E}\Bigg[\int_0^T \int_{\mathbb{R}^k} \left(\frac{1}{2} \left\langle \begin{pmatrix} Q_t & S_t^\top \\ S_t & R_t \end{pmatrix} \begin{pmatrix} X_t^\theta \\ a \end{pmatrix}, \begin{pmatrix} X_t^\theta \\ a \end{pmatrix} \right\rangle \nu_t^\theta(X_t^\theta; \mathrm{d}a) + \rho \mathcal{H}(\nu_t^\theta(X_t^\theta) \| \overline{\mathfrak{m}}_t) \right) \mathrm{d}t \\ + \frac{1}{2} (X_T^\theta)^\top G X_T^\theta \Bigg]\end{aligned} \tag{6.8}$$

The objective of the optimization is to minimize (6.8) over Θ or equivalently over $\mathcal{V}$. Comparing the parametrized optimization given by (6.7)-(6.8) with control formulation in (6.1)-(6.3), reveals that the cost (6.8) is minimal at $\theta^\star = (K^\star, V^\star)$ where $(K^\star, V^\star)$ as in (6.5) and that the minimal costs for the parametrized problem $\inf_{\theta \in \Theta} \mathcal{C}(\theta)$ is the same as the minimum reached in the control problem (6.1)-(6.3).

6.3.2 Loss landscape analysis

In this section, we analyse the loss landscape of the parametrized costs $\mathcal{C} : \Theta \to \mathbb{R}$. For this purpose, we need to introduce the following Lyapunov equations corresponding to the problem and their solutions. Thus, for any $\theta = (K, V) \in \Theta$, let $P^\theta \in C([0,T]; \mathbb{S}^d)$ be the solution of

$$\begin{aligned}(\tfrac{\mathrm{d}}{\mathrm{d}t} P)_t + (A_t + B_t K_t)^\top P_t + P_t^\top (A_t + B_t K_t) + (C_t + D_t K_t)^\top P_t (C_t + D_t K_t) \\ + K_t^\top (R_t + \rho \bar{V}_t^{-1}) K_t + S_t^\top K_t + K_t^\top S_t + Q_t = 0, \quad \text{a.e. } t \in [0,T]; \quad P_T = G,\end{aligned} \tag{6.9}$$

and let $\Sigma^\theta \in C([0,T]; \overline{\mathbb{S}_+^d})$ be the solution of: for a.e. $t \in [0,T]$,

$$\begin{aligned}(\tfrac{\mathrm{d}}{\mathrm{d}t} \Sigma)_t &= (A_t + B_t K_t) \Sigma_t + \Sigma_t (A_t + B_t K_t)^\top + (C_t + D_t K_t) \Sigma_t (C_t + D_t K_t)^\top + D_t V_t D_t^\top, \\ \Sigma_0 &= \mathbb{E}[\xi_0 \xi_0^\top].\end{aligned} \tag{6.10}$$

Given the assumptions on the coefficients of the LQC problem (Assumption 3), P^θ and Σ^θ exist and are unique by standard well-posedness results for linear differential equations. Furthermore, note that P^θ is independent of the covariance parameter V and only depends on the linear parameter K. Furthermore, by applying Itô's formula to $t \to X_t^\theta (X_t^\theta)^\top$ where X^θ is the solution of (6.7) and using the results from Lemma 2, one can see that $\Sigma_t^\theta = \mathbb{E}[X_t^\theta (X_t^\theta)^\top]$ for any $t \in [0,T]$.

Using these quantities, we can characterize the Gateaux derivatives of $\mathcal{C}$ in the direction of the parameters in the next proposition. In order to proof this proposition, we write the optimization problem as a deterministic control problem over the covariances of the state process Σ^θ and then apply the Pontryagin optimality principle to this problem.

Proposition 3. *Suppose Assumption 3 holds. For any $\theta \in \Theta$, let $P^\theta \in C([0,T]; \mathbb{S}^d)$ be the solution of* (6.9)*, and $\Sigma^\theta \in C([0,T]; \overline{\mathbb{S}_+^d})$ the solution of* (6.10)*. Then for all $\theta, \theta' \in \Theta$,*

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}\varepsilon} \mathcal{C}(K + \varepsilon K', V)\Big|_{\varepsilon=0} &= \int_0^T \langle \mathcal{D}_K(\theta)_t \Sigma_t^\theta, K_t' \rangle \, \mathrm{d}t, \\ \frac{\mathrm{d}}{\mathrm{d}\varepsilon} \mathcal{C}(K, V + \varepsilon(V' - V))\Big|_{\varepsilon=0} &= \int_0^T \langle \mathcal{D}_V(\theta)_t, V' - V \rangle \, \mathrm{d}t,\end{aligned}$$

where for a.e. $t \in [0,T]$,

$$\mathcal{D}_K(\theta)_t \coloneqq B_t^\top P_t^\theta + D_t^\top P_t^\theta (C_t + D_t K_t) + S_t + (R_t + \rho \bar{V}_t^{-1}) K_t, \tag{6.11}$$

$$\mathcal{D}_V(\theta)_t \coloneqq \frac{1}{2} (D_t^\top P_t^\theta D_t + R_t + \rho(\bar{V}_t^{-1} - V_t^{-1})). \tag{6.12}$$

With this characterization of the gradients $\mathcal{D}_K(\theta)$ and $\mathcal{D}_V(\theta)$, we can derive the conditions on the cost function $\mathcal{C} : \Theta \to \mathbb{R}$ that are crucial to show the convergence of certain policy gradient algorithms. First, we state a Łojasiewicz condition (or gradient domination condition) which depends locally on the parameters θ, i.e. the condition is non-uniform. Such a Łojasiewicz condition is needed since $\mathcal{C}$ is usually nonconvex in K (see Section 6.4 and, thus, crucial for the global convergence of gradient-based algorithms.

Proposition 4. *Suppose Assumption 3 and Assumption 4 hold. Let $\theta^\star \in \Theta$ be defined by the optimal solution to the control problem* (6.5)*. For any $\theta \in \Theta$, let $P^\theta \in C([0,T];\mathbb{S}^d)$ be the solution of* (6.9)*, and $\Sigma^\theta \in C([0,T];\overline{\mathbb{S}^d_+})$ the solution of* (6.10)*. Let $\mathcal{D}_K(\theta)$ and $\mathcal{D}_V(\theta)$ be defined as in* (6.11) *and* (6.12)*, respectively. Then for all $\theta \in \Theta$,*

$$\begin{aligned}\mathcal{C}(\theta) - \mathcal{C}(\theta^\star) \leq \int_0^T \Big(&\frac{1}{2}\langle (D_t^\top P_t^\theta D_t + R_t + \rho \bar{V}_t^{-1})^{-1} \mathcal{D}_K(\theta)_t, \mathcal{D}_K(\theta)_t \Sigma_t^{\theta^\star} \rangle \\ &+ \frac{1}{\rho} \max(\|V_t^\star\|_2^2, \|V_t\|_2^2) |\mathcal{D}_V(\theta)_t|^2 \Big) \, \mathrm{d}t.\end{aligned} \tag{6.13}$$

Next, we upper-bound the cost difference $\mathcal{C}(\theta') - \mathcal{C}(\theta)$ for any $\theta, \theta' \in \Theta$ by the first and second order terms of $\theta' - \theta$. In the context of policy gradient methods, such a condition is referred to as a "almost smoothness" condition (see e.g., [57, 90, 225]).

Proposition 5. *Suppose Assumption 3 holds. For any $\theta \in \Theta$, let $P^\theta \in C([0,T];\mathbb{S}^d)$ be the solution of* (6.9)*, and $\Sigma^\theta \in C([0,T];\overline{\mathbb{S}^d_+})$ the solution of* (6.10)*. Let $\mathcal{D}_K(\theta)$ and $\mathcal{D}_V(\theta)$ be defined as in* (6.11) *and* (6.12)*, respectively. Then for all $\theta, \theta' \in \Theta$,*

$$\begin{aligned}\mathcal{C}(\theta') - \mathcal{C}(\theta) \leq \int_0^T \Big(&\langle K_t' - K_t, \mathcal{D}_K(\theta)_t \Sigma_t^{\theta'} \rangle + \frac{1}{2} \langle K_t' - K_t, (D_t^\top P_t^\theta D_t + R_t + \rho \bar{V}_t^{-1})(K_t' - K_t) \Sigma_t^{\theta'} \rangle \\ &+ \langle \mathcal{D}_V(\theta)_t, V_t' - V_t \rangle + \frac{\rho}{4} \frac{|V_t' - V_t|^2}{\min(\lambda_{\min}^2(V_t), \lambda_{\min}^2(V_t'))} \Big) \, \mathrm{d}t.\end{aligned}$$

We note that the conditions derived in Proposition 4 and Proposition 5 are local properties, i.e. the bounds depend explicitly on P^θ and Σ^θ and do not allow for uniform bounds over the unbounded parameter set Θ. For policy gradient methods with finite-dimensional parameter spaces, such as in discrete time or stationary settings, one may derive global bounds from local bounds by observing that the sublevel set $\{\theta \in \Theta \mid \mathcal{C}(\theta) < \beta\}$ is bounded for any $\beta > 0$ and then propose algorithms whose iterates remain within a certain sublevel set (see e.g., [57, 79, 90]). Due to the non-coerciveness in the continuous time setting, the algorithm design becomes more involved and it needs to be shown that the bounds hold uniformly along the iterates.

6.4 Additional results

This section contains additional results obtained by co-authors from [71]. We provide a shortened version of these results for context and refer to [71] for more detail.

6.4.1 Noncoerciveness of costs

A main difference between the optimization landscape with finite dimensional parameter space versus the infinite case is the noncoerciveness of costs. We say a function on a normed space $(X, \|\cdot\|)$ is coercive if $\lim_{\|x\| \to \infty} f(x) = \infty$. The following proposition gives an example of a noncoercive and nonconvex cost function for a time-continuous setting.

Proposition 6 (cf. Proposition 2.4 in [71]). *Let $\mathcal{C} : L^2(0,1;\mathbb{R}) \to \mathbb{R}$ be such that for all $K \in L^2(0,1;\mathbb{R})$,*

$$\mathcal{C}(K) \coloneqq \int_0^1 (K_t X_t)^2 \,\mathrm{d}t, \quad \text{with } X_t = 1 + \int_0^t K_s X_s \,\mathrm{d}s,\ t \in [0,1]. \tag{6.14}$$

Then $\mathcal{C} : L^2(0,1;\mathbb{R}) \to \mathbb{R}$ is neither coercive nor quasiconvex. In particular, let $K^\varepsilon \in L^2(0,1;\mathbb{R})$, $\varepsilon > 0$, be such that $K^\varepsilon_t = -(1+\varepsilon-t)^{-1}$ for all $t \in [0,1]$. Then $\lim_{\varepsilon\to 0} \|K^\varepsilon\|_{L^1} = \infty$ and $\sup_{\varepsilon>0} \mathcal{C}(K^\varepsilon) = 1$. Moreover, there exists $\varepsilon_0 > 0$ such that for all $\varepsilon \in (0,\varepsilon_0]$, $\mathcal{C}(0.5K^\varepsilon) > \max\{\mathcal{C}(\mathbf{0}), \mathcal{C}(K^\varepsilon)\}$, with $\mathbf{0}$ being the zero function.

6.4.2 Geometry-aware policy gradient method

We present a variant of policy gradient methods that takes into account the geometry of the underlying problem to preserve a-priori bounds. Thus, consider any initial guess $\theta^0 = (K^0, V^0) \in \Theta$ and a stepsize $\tau > 0$, then the iterates of parameters $(\theta^n)_{n\in\mathbb{N}} \subset \mathcal{B}(0,T;\mathbb{R}^{k\times d} \times \mathbb{S}^k)$ are defined by the following gradient descent algorithm:

$$K^{n+1}_t = K^n_t - \tau \mathcal{D}_K(\theta^n)_t, \quad V^{n+1}_t = V^n_t - \tau \mathcal{D}^{\mathrm{bw}}_V(\theta^n)_t, \quad \text{a.e. } t \in [0,T], \tag{6.15}$$

with

$$\mathcal{D}^{\mathrm{bw}}_V(\theta^n)_t = \mathcal{D}_V(\theta^n)_t V^n_t + V^n_t \mathcal{D}_V(\theta^n)_t, \tag{6.16}$$

where $\mathcal{D}_K(\theta)$ and $\mathcal{D}_V(\theta)$ are as in (6.11) and (6.12), respectively. The policy gradient algorithm (6.15) normalises the gradients of θ^n (cf. Proposition 3) by incorporating the local geometry of the parameter space. For $(K^n)_{n\in\mathbb{N}}$, the algorithm uses the steepest descent direction on the manifold of Gaussian policies endowed with the Fisher information metric. This approach is also known as natural gradient descent (see [116]). For $(V^n)_{n\in\mathbb{N}}$, the steepest descent is on the matrix manifold $\mathbb{S}^k_+$ endowed with the Bures-Wasserstein metric [91] which is the geometry induced by the 2-Wasserstein metric over the space of centered nondegenerate Gaussian measures. In particular, this choice of gradient preserves pointwise upper and lower bound on the eigenvalues of $(V^n)_{n\in\mathbb{N}}$ without the use of projections. Using the geometry-aware gradient and a sufficiently small learning rate in the descent algorithm it can be proven that the parameters remain in the set of admissible parameters along the iterates. This observation can be found in the convergence analysis in [71].

6.4.3 Convergence analysis

Based on characteristics of the loss landscape in Section 6.3, the global linear convergence of Algorithm (6.15) can be established. Exploiting the local geometry on the parameters in Algorithm (6.15) allows to establish uniform-bounds along the iterates without the use of computationally expensive projection steps. The convergence result is summarized in the following theorem:

Theorem 4 (cf. Theorem 2.6 in [71]). *Suppose Assumption 3 and Assumption 4 hold, and $\mathbb{E}[\xi_0\xi_0^\top] \succ 0$. Let $\theta^0 \in \Theta$, and for each $\tau > 0$, let $(\theta^n)_{n\in\mathbb{N}} \subset \mathcal{B}(0,T;\mathbb{R}^{k\times d} \times \mathbb{S}^k)$ be defined in (6.15). Then there exists $\tau_0, C_1, C_2 > 0$ such that for all $\tau \in (0,\tau_0]$ and $n \in \mathbb{N}_0$,*

(1) $\mathcal{C}(\theta^{n+1}) \le \mathcal{C}(\theta^n)$ and $\mathcal{C}(\theta^{n+1}) - \mathcal{C}(\theta^\star) \le (1-\tau C_1)(\mathcal{C}(\theta^n) - \mathcal{C}(\theta^\star))$, with $\theta^\star$ defined in (6.5),

(2) $\|K^n - K^\star\|^2_{L^2} + \|V^n - V^\star\|^2_{L^2} \le C_2(1-\tau C_1)^n$.

6.4.4 Mesh-independent linear convergence with discrete-time policies

Next, we consider a policy gradient method for actions taken in discrete time and present results that show that proper rescaling of the gradient by mesh size leads to robust convergence behaviour across different mesh sizes. For that purpose, consider $\mathscr{P}_{[0,T]}$ the collection of all partitions of $[0,T]$. For any partition

$\pi = \{0 = t_0 < \cdots < t_N = T\} \in \mathscr{P}_{[0,T]}$, we define the mesh size of π by $|\pi| = \max_{i=0,\dots,N-1}(t_{i+1} - t_i)$. Furthermore, $\Theta^\pi \subset \Theta$ denotes the set of piecewise constant policies on the partition π:

$$\Theta^\pi = \{\theta \in \Theta \mid \theta_t = \theta_{t_i},\ \text{a.e. } t \in [t_i, t_{i+1}) \text{ and all } i \in \{0, \dots, N-1\}\}. \tag{6.17}$$

With this define the minimization problem over piecewise constant policies for a partition π:

$$\mathcal{C}_\pi^\star = \inf_{\theta \in \Theta^\pi} \mathcal{C}(\theta). \tag{6.18}$$

Note that since Θ^π is a subset of Θ, $\mathcal{C}_\pi^\star \geq \inf_{\theta\in\Theta} \mathcal{C}(\theta) = \mathcal{C}(\theta^\star) > -\infty$.

For any $\pi = \{0 = t_0 < \cdots < t_N = T\} \in \mathscr{P}_{[0,T]}$ we identify the parameter space Θ^π with $(\mathbb{R}^{k\times d} \times \mathbb{S}_+^k)^N$ using the parameterisation:

$$(\mathbb{R}^{k\times d} \times \mathbb{S}_+^k)^N \ni (K_i, V_i)_{i=1}^{N-1} \mapsto \left(\sum_{i=0}^{N-1} K_i \mathbb{1}_{[t_i, t_{i+1})}(t),\ \sum_{i=0}^{N-1} V_i \mathbb{1}_{[t_i, t_{i+1})}(t) \right)_{t\in[0,T]} \in \Theta^\pi, \tag{6.19}$$

and in a slight abuse of notation, denotes $\mathcal{C} : (\mathbb{R}^{k\times d} \times \mathbb{S}_+^k)^N \to \mathbb{R}$ the cost of the Gaussian policy induced by the parameterisation (6.6) and (6.19). With $\tau > 0$ and starting from an initial $\theta^{\pi,0} \in \Theta^\pi$, the sequence $(\theta^{\pi,n})_{n\in\mathbb{N}_0} \subset \Theta^\pi$ is updated for all $n \in \mathbb{N}_0$ and $i \in \{0, \dots, N-1\}$ where $\theta_t^{\pi,n+1} = (K_i^{\pi,n+1}, V_i^{\pi,n+1})$ for all $t \in [t_i, t_{i+1})$, by the scaled policy gradient algorithm

$$\begin{aligned} K_i^{\pi,n+1} &= K_i^{\pi,n} - \frac{\tau}{\Delta_i} \nabla_{K_i} \mathcal{C}(\theta^{\pi,n}) \left(\Sigma_{t_i}^{\theta^{\pi,n}}\right)^{-1}, \\ V_i^{\pi,n+1} &= V_i^{\pi,n} - \frac{\tau}{\Delta_i} \left(V_i^{\pi,n} \nabla_{V_i} \mathcal{C}(\theta^{\pi,n}) + \nabla_{V_i} \mathcal{C}(\theta^{\pi,n}) V_i^{\pi,n}\right), \end{aligned} \tag{6.20}$$

where $\Delta_i = t_{i+1} - t_i$, $\Sigma_{t_i}^{\theta^{\pi,n}} = \mathbb{E}[X_{t_i}^{\theta^{\pi,n}} (X_{t_i}^{\theta^{\pi,n}})^\top]$, and $\nabla_{K_i}\mathcal{C}$ (resp. $\nabla_{V_i}\mathcal{C}$) is the partial derivative of $\mathcal{C}$ with respect to the matrix K_i (resp. V_i). Note that for implementation the state covariances and policy gradients may be estimated by a Monte-Carlo method (see Section 6.5 and also [57, 90]).

The convergence behaviour of this rescaled policy gradient algorithm is quantified by the number of required iterations $N^\pi(\varepsilon)$ necessary to achieve the accuracy $\varepsilon > 0$. More specifically consider the sequence $(\theta^{\pi,n})_{n\in\mathbb{N}_0}$ generated by (6.20) where $\theta^{\pi,0} \in \Theta^\pi$ and $\tau > 0$, then for any $\varepsilon > 0$, define

$$N^\pi(\varepsilon) \coloneqq \min\left\{ n \in \mathbb{N}_0 \,\middle|\, \mathcal{C}(\theta^{\pi,n}) - \inf_{\theta\in\Theta^{\pi_m}} \mathcal{C}(\theta) < \varepsilon \right\} \in \mathbb{N}_0 \cup \{\infty\}. \tag{6.21}$$

Note that N^π assumes a fixed mesh π and that the residual is measured relative to the minimum cost $\mathcal{C}_\pi^\star$ over piecewise constant policies Θ^π. To compare the speed of convergence, we introduce the analogous quantity for the time-continuous policies. Therefore, consider the sequence $(\theta^n)_{n\in\mathbb{N}_0}$ generated by (6.15) starting from $\theta^0 \in \Theta$ and using a stepsize $\tau > 0$. Then for any $\varepsilon > 0$, define

$$N(\varepsilon) \coloneqq \min\left\{ n \in \mathbb{N}_0 \,\middle|\, \mathcal{C}(\theta^n) - \inf_{\theta\in\Theta} \mathcal{C}(\theta) < \varepsilon \right\} \in \mathbb{N}_0 \cup \{\infty\}. \tag{6.22}$$

The next Theorem shows that for a sufficiently fine grid $N^\pi(\varepsilon)$ is essentially equal to $N(\varepsilon)$. Together with Theorem 4, this implies that the rescaled policy gradient method allows for linear convergence uniformly across different timescales.

Theorem 5 (cf. Theorem 2.7 and Corollary 2.8 in [71])**.** *Suppose Assumption 3 and Assumption 4 hold, $\mathbb{E}[\xi_0\xi_0^\top] \succ 0$, $D \in C([0,T]; \mathbb{R}^{d\times k})$, $R \in C([0,T]; \mathbb{S}^k)$ and $\bar{V} \in C([0,T]; \mathbb{S}_+^k)$. Let $\theta^0 \in L^2(0,T; \mathbb{R}^{k\times d}) \times C([0,T]; \mathbb{S}_+^k)$, let $(\pi_m)_{m\in\mathbb{N}} \subset \mathscr{P}_{[0,T]}$ be such that $\lim_{m\to\infty} |\pi_m| = 0$, and let $(\theta^{\pi_m,0})_{m\in\mathbb{N}} \subset \Theta$ be such that $\theta^{\pi_m,0} \in \Theta^{\pi_m}$ for all*

$m \in \mathbb{N}$ *and* $\lim_{m\to\infty} \|\theta^{\pi_m,0} - \theta^0\|_{L^2\times L^\infty} = 0$. *Then there exists* $\tau_0 > 0$ *such that for all* $\tau \in (0,\tau_0)$ *and* $\varepsilon > 0$, *there exists* $\mathfrak{m} \in \mathbb{N}$ *such that*

$$N(\varepsilon) - 1 \le N^{\pi_m}(\varepsilon) \le N(\varepsilon), \quad \forall m \in \mathbb{N} \cap [\mathfrak{m}, \infty), \tag{6.23}$$

where $N(\varepsilon) = \mathcal{O}\big(\tau^{-1}\log(\varepsilon^{-1})\big)$ *for all small* τ *and* ε.

Remark 14 (**Scaling hyper-parameters with timescales**). *Note that in* (6.20) *it is crucial to scale the the stepsize* τ *by the time mesh* Δ_i *to achieve robustness for all small mesh sizes. A standard discrete-time policy gradient method would set* $\Delta_i = 1$ *in* (6.20) *for any time mesh. For sufficiently fine grids, this implies a vanishing stepsize* $\tau\Delta_i$ *in* (6.15), *since* $\nabla_{K_i}\mathcal{C}(\theta)\left(\Sigma^\theta_{t_i}\right)^{-1} \approx \mathcal{D}_K(\theta)_{t_i}\Delta_i$ *and* $\nabla_{V_i}\mathcal{C}(\theta^{\pi,n}) \approx \mathcal{D}_V(\theta)_{t_i}\Delta_i$ *(cf. Proposition 3). This vanishing stepsize explains the deteriorating performance of conventional discrete-time policy gradient methods on tasks with small mesh sizes. Normalising the stepsize with* Δ_i, *on the other hand, makes sure that the time-continuous limit* (6.15) *is reached as* $|\pi| \to 0$ *and allowing for mesh-independent convergence. These observations are confirmed in numerical experiments in Section 6.5.*

6.5 Numerical experiments

In this section, we test the theoretical findings through a numerical experiment on an exploratory LQC problem arising from mean-variance portfolio selection. Our experiments confirm that the proposed algorithm (6.20) converges linearly to the optimal policy. They also show that conventional policy gradient methods exhibit a degraded performance for small timesteps in the policy updates, while the rescaled algorithm (6.20) demonstrates robustness across different step sizes.

Problem setup We minimise the following cost $\mathcal{C} : \Theta \to \mathbb{R}$ (cf. (6.8)):

$$\mathcal{C}(\theta) = \mathbb{E}\left[\frac{1}{2}\mu\,(X^\theta_T)^2 + \rho\int_0^T \mathcal{H}(\nu^\theta_t(X^\theta_t)\|\overline{\mathfrak{m}}_t)\,\mathrm{d}t\right], \tag{6.24}$$

where $\overline{\mathfrak{m}}_t = \mathcal{N}(0,\bar{V})$ with $\bar{V} \in \mathbb{S}^3_+$, and for each $\theta \in \Theta$, $X^\theta \in \mathcal{S}^2(0,T;\mathbb{R})$ satisfies for all $t \in [0,T]$,

$$\mathrm{d}X_t = \int_{\mathbb{R}^3}\big(B_t a\,\nu^\theta_t(X_t;\mathrm{d}a)\big)\,\mathrm{d}t + \left(\int_{\mathbb{R}^3}\sum_{j=1}^3\left(D^{(j)}a\right)^2\nu^\theta_t(X_t;\mathrm{d}a)\right)^{\frac{1}{2}}\mathrm{d}W_t, \quad X_0 = \xi_0, \tag{6.25}$$

for some $B : [0,T] \to \mathbb{R}^{1\times 3}$ and $D^{(j)} \in \mathbb{R}^{1\times 3}$, $j = 1,2,3$. The coefficients are chosen as follows: $T = 1$, $\mu = 0.5$, $\rho = 0.01$, $\bar{V} = 0.1I_3$, $\xi_0 \sim \mathcal{N}(0.5, 0.01)$, $B_t = (0.4, 0.8, 0.4) + 0.2\sin(2\pi t)\mathbf{1}_3$ for all $t \in [0,T]$, and $D = \begin{pmatrix} D^{(1)} \\ D^{(2)} \\ D^{(3)} \end{pmatrix}$ with $D^\top D = \begin{pmatrix} 0.5 & 0.25 & -0.125 \\ 0.25 & 1 & -0.25 \\ -0.125 & -0.25 & 0.5 \end{pmatrix}$. Note that $D^\top D \in \mathbb{S}^3_+$, and hence Assumption 4 holds for all $\rho \ge 0$ (see [228]).

The problem (6.24)-(6.25) arises from an exploratory mean-variance portfolio selection problem, where the agent allocates their wealth among three risky assets by sampling from the policy ν^θ (see [112, 216]). Indeed, as shown in [205], for each $\theta = (K,V) \in \Theta$, $\mathcal{C}(\theta)$ can be approximated by replacing (6.25) with the following dynamics: $X_0 = \xi_0$, and for all $t \in [0,T]$,

$$\mathrm{d}X_t = B_t\left(K_tX_t + V_t^{\frac{1}{2}}\xi_t\right)\mathrm{d}t + \sum_{j=1}^3 D^{(j)}\left(K_tX_t + V_t^{\frac{1}{2}}\xi_t\right)\mathrm{d}W^{(j)}_t \tag{6.26}$$

with $\xi_t = \sum_{i=1}^n \zeta_i \mathbb{1}_{[t_i,t_{i+1})}(t)$, where $(W^{(j)})_{j=1}^3$ are independent Brownian motions, $(\zeta_i)_{i=1}^n$ are independent standard normal random vectors, and $(t_i)_{i=1}^n$ is a sufficiently fine time mesh.

Linear convergence We first implement (6.20) on the uniform time mesh π_c with mesh size $1/128$, and examine its convergence. The scheme is initialised with $K^0 \equiv (1/3, 1/3, 1/3)$ and $V^0 \equiv 0.1D^\top D$. For each $n \in \mathbb{N}_0$, given $\theta^n \subset \Theta^{\pi_c}$, we simulate 10^5 independent trajectories of (6.26) (with $\theta = \theta^n$) using the Euler–Maruyama method on the mesh π_c, evaluate the approximate value $\widehat{\mathcal{C}}(\theta^n)$ and state covariance $\widehat{\Sigma}^n$ using the empirical distribution of these sample paths, and compute an approximate gradient $(\widehat{\nabla_{\theta_i^n}\mathcal{C}})_{i=0}^{127}$ using automatic differentiation (cf. Chapter 5). The iterate θ^n is updated by (6.20) with $\widehat{\Sigma}^n$, $\widehat{\nabla_{\theta^n}\mathcal{C}}$ and the stepsize $\tau = 0.01$. The performance of the scheme is measured by the errors $(\widehat{\mathcal{C}}(\theta^n) - \mathcal{C}^\star)_{n\in\mathbb{N}_0}$, where $\mathcal{C}^\star$ is the optimal cost of (6.24), (6.25) obtained by Riccati equations. Further implementation details are given at the end of this section.

Figure 6.1 (left) exhibits the decay of $(\widehat{\mathcal{C}}(\theta^n) - \mathcal{C}^\star)_{n\in\mathbb{N}_0}$ with respect to the number of iterations, where the solid line and the shaded area indicate the sample mean and the spread over 10 repeated experiments, respectively. It clearly shows the linear convergence of (6.20), as indicated in Theorems 4 and 5. The seemingly higher noise for larger iteration numbers results from the small errors in this case, so that the fluctuations appear larger on the log scale. The variance could be reduced by increasing the number of samples.

Robustness in action frequency We then compare the performance of (6.20) with a standard policy gradient method for different policy discretisation timescales. The former (termed "scaled PG") scales the gradients with the discretisation mesh size, while the latter (termed "unscaled PG") updates the policy with unscaled gradients. More precisely, let $\theta^0 = (K^0, V^0)$ be a fixed initial guess given as above, and $\pi_m = \{i\frac{1}{m}\}_{i=0}^m$, $m \in \{8, 16, 32, 64, 128\}$ be a family of time meshes. For each $m \in \{8, 16, 32, 64, 128\}$, the scaled PG method generates the iterates $(\theta^{\pi_m, n})_{n\in\mathbb{N}_0} \subset \Theta^{\pi_m}$ according to (6.20) with $\tau = 0.01$ and $\Delta_i = 1/m$, where the required gradients for each iteration are computed as above. The unscaled PG method follows (6.20) with $\tau = 0.08$ and $\Delta_i = 1$ for all m. Here, a larger stepsize has been adopted for the unscaled PG method so that the two algorithms coincide for the coarsest mesh π_8.

Figure 6.1 (right) compares, for different discretisation timescales, the numbers of required iterations $N^{\pi_m}(0.01)$ for both schemes to achieve an accuracy of $\epsilon = 0.01$ (cf. (6.21)). One can observe clearly that the number of required iterations for the unscaled PG method exhibits a linear growth in the number of action time points. In contrast, the number of iterations for the scaled PG method remains constant for all meshes. This confirms the theoretical result in Theorem 5, and shows that the scaled PG method outperforms conventional PG methods for fine meshes.

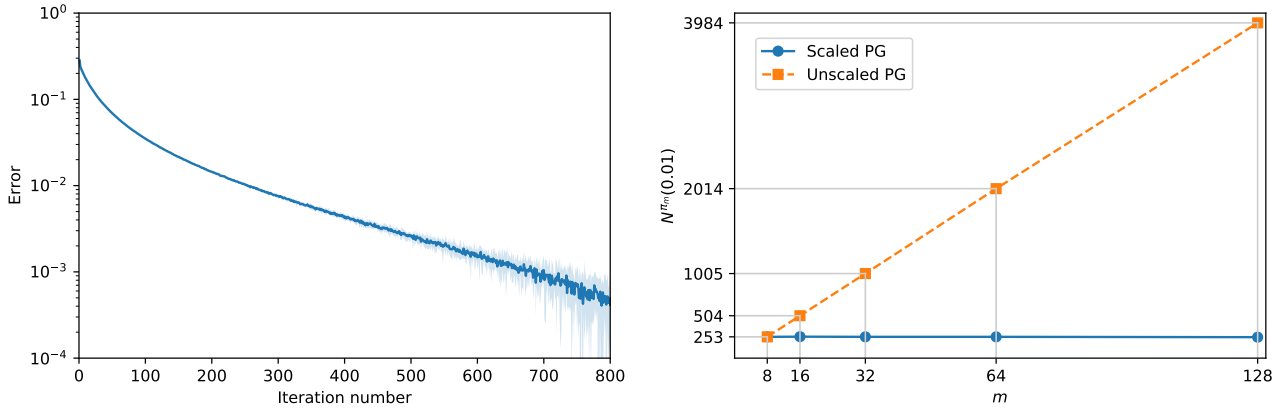


Figure 6.1: Convergence and robustness of the PG method (6.20).

6.5.1 Experiment details

This section presents additional details for the numerical experiments.

Calculation of optimal cost $\mathcal{C}^\star$ Let $P^\star \in C([0,T];\mathbb{R})$ solve the following Riccati equation: for all $t \in [0,T]$,

$$\left(\tfrac{\mathrm{d}P}{\mathrm{d}t}\right)_t - B_t \left(P_t \textstyle\sum_{j=1}^3 (D^{(j)})^\top D^{(j)} + \rho \bar{V}^{-1}\right)^{-1} B_t^\top P_t^2 = 0; \quad P_T = \tfrac{\mu}{2}. \tag{6.27}$$

Then the optimal policy of (6.24)-(6.25) satisfies $\nu_t^\star(x) = \mathcal{N}(K_t^\star x, V_t^\star)$ for all $(t,x) \in [0,T] \times \mathbb{R}$, where

$$K_t^\star = -\left(P_t^\star \textstyle\sum_{j=1}^3 (D^{(j)})^\top D^{(j)} + \rho \bar{V}^{-1}\right)^{-1} B_t^\top P_t^\star, \quad V_t^\star = \rho \left(P_t^\star \textstyle\sum_{j=1}^3 (D^{(j)})^\top D^{(j)} + \rho \bar{V}^{-1}\right)^{-1}.$$

Moreover, let $\varphi^\star \in C([0,T];\mathbb{R})$ satisfy for all $t \in [0,T]$,

$$(\tfrac{\mathrm{d}}{\mathrm{d}t}\varphi)_t + \tfrac{1}{2}\mathrm{tr}\left(\left(P_t^\star \textstyle\sum_{j=1}^3 (D^{(j)})^\top D^{(j)} + \rho \bar{V}^{-1}\right) V_t^\star\right) + \tfrac{\rho}{2}\left(-3 + \ln\left(\tfrac{\det(\bar{V})}{\det(V_t)}\right)\right) = 0; \quad \varphi_T = 0, \tag{6.28}$$

Then the optimal cost of (6.24)-(6.25) is given by $\mathcal{C}^\star = \frac{1}{2}\mathbb{E}[\xi_0^\top \xi_0] P_0^\star + \varphi_0^\star$.

Implementation details The numerical experiments are implemented by using Tensorflow. To examine the linear convergence, the scheme (6.20) is implemented on the uniform time grid π_c with mesh size $\Delta t = 1/128$. Indeed, let $K^0 \equiv (1/3, 1/3, 1/3)$ and $V^0 \equiv 0.1 D^\top D$ be the initial guess. For each $n \in \mathbb{N}_0$, given $\theta^n = (K_i^n, V_i^n)_{i=0}^{127}$, consider the Euler–Maruyama discretisation of (6.26): $X_0 = \xi_0$ and for all $i = 0, \ldots, 127$,

$$X_{i+1} = X_i + B_{i\Delta t}\left(K_i^n X_i + (V_i^n)^{\frac{1}{2}}\zeta_i\right)\Delta t + \sum_{j=1}^3 D^{(j)}\left(K_i^n X_i + (V_i^n)^{\frac{1}{2}}\zeta_i\right)\Delta W_i^{(j)}, \tag{6.29}$$

where $(\Delta W_i^{(j)})_{i=0,\ldots,127, j=1,\ldots 3}$ are independent normal random variables with mean zero and variance $1/128$, and $(\zeta_i)_{i=0}^{127}$ are independent standard normal random vectors in $\mathbb{R}^3$. We simulate $N_{\mathrm{MC}} = 10^5$ independent trajectories of (6.29) and approximate $\mathcal{C}(\theta^n)$ as follows (cf. (6.35)):

$$\widehat{\mathcal{C}}(\theta^n) \coloneqq \frac{1}{N_{\mathrm{MC}}}\sum_{l=1}^{N_{\mathrm{MC}}}\frac{1}{2}\left(\mu X_{128,l}^2 + \rho \sum_{i=0}^{127}\left((K_i^n)^\top \bar{V}^{-1} K_i^n X_{i,l}^2 + \mathrm{tr}(\bar{V}^{-1}V_i^n) - 3 + \ln\left(\tfrac{\det(\bar{V})}{\det(V_i^n)}\right)\right)\Delta t\right),$$

where $(X_{i,l})_{i=0}^{128}$, $l = 1, \ldots, N_{\mathrm{MC}}$, represents the l-th trajectory of (6.29). The required gradients $(\widehat{\nabla_{K_i^n}\mathcal{C}}, \widehat{\nabla_{V_i^n}\mathcal{C}})_{i=1}^{127}$ are computed using automatic differentiation along these paths, and for each $i = 0, \ldots, 127$, the state covariance $\Sigma_{i\Delta t}^{\theta^n}$ is estimated by $\widehat{\Sigma}_i^n \coloneqq \frac{1}{N_{\mathrm{MC}}}\sum_{l=1}^{N_{\mathrm{MC}}} X_{i,l}^2$. The policy is then updated as follows (cf. (6.20)): for all $i = 0, \cdots, 127$,

$$K_i^{n+1} = K_i^n - \frac{\tau}{\Delta t \widehat{\Sigma}_i^n}\widehat{\nabla_{K_i^n}\mathcal{C}}, \quad V_i^{n+1} = V_i^n - \frac{\tau}{\Delta t}\left(\widehat{\nabla_{V_i^n}\mathcal{C}}\, V_i^n + V_i^n\, \widehat{\nabla_{V_i^n}\mathcal{C}}\right).$$

The optimal cost of (6.24)-(6.25) is computed by solving (6.27) and (6.28) with the explicit Euler scheme on π_c, which leads to the value $\mathcal{C}^\star = 0.0402$.

To examine the robustness of (6.20) in time discretisation, a family of coarser time grids $\pi_m = \{i\frac{1}{m}\}_{i=0}^m \subset \pi_c$, $m \in \{8, 16, 32, 64, 128\}$, have been introduced. The PG scheme only updates policy parameters at the grid points of these coarser grids. However, to mimic a continuous-time environment, the performance of each policy iterate is still evaluated by simulating (6.29) on the fine grid π_c (with mesh size $\Delta t = 1/128$). In particular, let (K^0, V^0) be given as above. For each $m \in \{8, 16, 32, 64, 128\}$ and $n \in \mathbb{N}_0$, given $\theta^n = (K_j^n, V_j^n)_{j=0}^{m-1}$, consider the following Euler-Maruyama discretisation of (6.26): $X_0 = \xi_0$ and for all $j = 0, \ldots, m-1$, and all $i = 0, \ldots, 127$ such that $\frac{j}{m} \le i\Delta t < \frac{j+1}{m}$,

$$X_{i+1} = X_i + B_{i\Delta t}\left(K_j^n X_i + (V_j^n)^{\frac{1}{2}}\zeta_i\right)\Delta t + \sum_{j=1}^3 D^{(j)}\left(K_j^n X_i + (V_j^n)^{\frac{1}{2}}\zeta_i\right)\Delta W_i^{(j)}, \tag{6.30}$$

where $(\Delta W_i^{(j)})_{i=0,\dots,127,j=1,\dots 3}$ and $(\zeta_i)_{i=0}^{127}$ are independent random variables as in (6.29). We shall sample 10^5 independent trajectories of (6.30), and use them to approximate the gradients in $(K_j^n, V_j^n)_{j=0}^{m-1}$ and the state covariance $(\Sigma_{j/m}^{\theta^n})_{j=0}^{m-1}$ with similar methods as above. The scaled PG method (6.20) then updates the parameters by: for all $j = 0, \dots, m-1$,

$$K_j^{n+1} = K_j^n - \frac{m\tau}{\widehat{\Sigma}_j^n}\widehat{\nabla_{K_j^n}\mathcal{C}}, \quad V_j^{n+1} = V_j^n - m\tau\left(\widehat{\nabla_{V_j^n}\mathcal{C}}\, V_j^n + V_j^n\, \widehat{\nabla_{V_j^n}\mathcal{C}}\right), \quad \text{with } \tau = 0.01, \tag{6.31}$$

while the unscaled PG method updates the parameters by: for all $j = 0, \dots, m-1$,

$$K_j^{n+1} = K_j^n - \frac{\tau}{\widehat{\Sigma}_j^n}\widehat{\nabla_{K_j^n}\mathcal{C}}, \quad V_j^{n+1} = V_j^n - \tau\left(\widehat{\nabla_{V_j^n}\mathcal{C}}\, V_j^n + V_j^n\, \widehat{\nabla_{V_j^n}\mathcal{C}}\right), \quad \text{with } \tau = 0.08. \tag{6.32}$$

Let $(\theta^{\pi_m,n})_{n\in\mathbb{N}_0}$ be the policy iterate generated by (6.31), define $N^{\pi_m}(0.01)$ by

$$N^{\pi_m}(0.01) \coloneqq \min\left\{n \in \mathbb{N}_0 \,|\, \widehat{\mathcal{C}}(\theta^{\pi_m,n}) - \mathcal{C}^\star_{\pi_m}) < 0.01\right\},$$

where $\mathcal{C}^\star_{\pi_m} \coloneqq \frac{1}{50}\sum_{n=951}^{1000}\widehat{\mathcal{C}}(\theta^{\pi_m,n})$ approximates the optimal cost among all piecewise constant polices on π_m. The quantity $N^{\pi_m}(0.01)$ is defined similarly for the iterates generated by (6.32).

6.6 Proofs: Analysis of optimisation landscape

This section gives the proofs for the results stated in Section 6.3.

Before proving the main propositions, several technical lemmas are presented. The first lemma represents the coefficients of (6.7) and the cost function (6.8) in terms of $\theta = (K, V)$. As this result is based on elemental calculation, the proof is omitted.

Lemma 2. *Suppose Assumption 3 holds. Then for all $\nu^\theta \in \mathcal{V}$ and $(t,x) \in [0,T]\times\mathbb{R}^d$,*

$$\begin{aligned}
\Phi_t(x,\nu_t^\theta(x)) &= (A_t + B_tK_t)x,\\
\Gamma_t(x,\nu_t^\theta(x)) &= \left((C_t+D_tK_t)xx^\top(C_t+D_tK_t)^\top + D_tV_tD_t^\top\right)^{\frac{1}{2}},\\
\int_{\mathbb{R}^k}\left\langle\begin{pmatrix}Q_t & S_t^\top\\ S_t & R_t\end{pmatrix}\begin{pmatrix}x\\a\end{pmatrix},\begin{pmatrix}x\\a\end{pmatrix}\right\rangle \nu_t^\theta(x;\mathrm{d}a) &= \left\langle\begin{pmatrix}Q_t & S_t^\top\\ S_t & R_t\end{pmatrix}\begin{pmatrix}x\\K_tx\end{pmatrix},\begin{pmatrix}x\\K_tx\end{pmatrix}\right\rangle + \mathrm{tr}(R_tV_t),\\
\mathcal{H}(\nu_t^\theta(x)\|\overline{\mathfrak{m}}_t) &= \frac{1}{2}\left((K_tx)^\top\bar{V}_t^{-1}K_tx + \mathrm{tr}(\bar{V}_t^{-1}V_t) - k + \ln\left(\frac{\det(\bar{V}_t)}{\det(V_t)}\right)\right).
\end{aligned}$$

In the next Lemma we use Itô's formula to represent the cost of the parametrized optimization problem as the solution of a PDE depending on the solution of the Lyapunov equation P^θ (see (6.9)).

Lemma 3. *Suppose Assumption 3 holds. For each $\theta \in \Theta$, $P^\theta \in C([0,T];\mathbb{S}^d)$ is the solution to* (6.9)*, and $\varphi^\theta \in C([0,T];\mathbb{R}^d)$ the solution for a.e. $t \in [0,T]$ of*

$$(\tfrac{\mathrm{d}}{\mathrm{d}t}\varphi)_t + \tfrac{1}{2}\mathrm{tr}\left((D_t^\top P_t^\theta D_t + R_t + \rho\bar{V}_t^{-1})V_t\right) + \tfrac{\rho}{2}\left(-k + \ln\left(\tfrac{\det(\bar{V}_t)}{\det(V_t)}\right)\right) = 0; \quad \varphi_T = 0,$$

and let $u^\theta : [0,T]\times\mathbb{R}^d \to \mathbb{R}$ be such that $u_t^\theta(x) = \frac{1}{2}x^\top P_t^\theta x + \varphi_t^\theta$ for all $(t,x) \in [0,T]\times\mathbb{R}^d$. Then for all $\theta \in \Theta$ and $x \in \mathbb{R}^d$, u^θ solves

$$\begin{aligned}
(\tfrac{\mathrm{d}}{\mathrm{d}t}u^\theta)_t &+ \frac{1}{2}\mathrm{tr}\left(\Gamma_t(x,\nu_t^\theta(x))\Gamma_t(x,\nu_t^\theta(x))^\top(\nabla_x^2u^\theta)_t(x)\right) + \Phi_t(x,\nu_t^\theta(x))^\top(\nabla_xu^\theta)_t(x)\\
&+ \frac{1}{2}\left(x^\top Q_tx + x^\top S_t^\top K_tx + (K_tx)^\top S_tx + (K_tx)^\top R_tK_tx + \mathrm{tr}(R_tV_t)\right)\\
&+ \frac{\rho}{2}\left((K_tx)^\top\bar{V}_t^{-1}K_tx + \mathrm{tr}(\bar{V}_t^{-1}V_t) - k + \ln\left(\frac{\det(\bar{V}_t)}{\det(V_t)}\right)\right) = 0, \quad \text{a.e. } t \in [0,T],
\end{aligned} \tag{6.33}$$

and $u_T^\theta(x) = \frac{1}{2}x^\top Gx$*, where* $\nabla_x u^\theta$ *and* $\nabla_x^2 u^\theta$ *are the gradient and Hessian of* u^θ *in* x*, respectively. Moreover, it holds that* $\mathcal{C}(\theta) = \mathbb{E}[u_0^\theta(\xi_0)]$*.*

Proof. Let $X^\theta \in \mathcal{S}^2(0,T;\mathbb{R}^d)$ be the solution to the state dynamics (6.7). To simplify notation, the θ in the superscripts of all variables will be omitted.

With Lemma 2 and the definition of u, direct calculation yield for all $(t,x) \in [0,T]$,

$$\begin{aligned}\Phi_t(x,\nu_t^\theta(x))^\top(\nabla_x u^\theta)_t(x) &= \tfrac{1}{2}x^\top\left((A_t+B_tK_t)^\top P_t + P_t(A_t+B_tK_t)\right)x,\\ \operatorname{tr}\left(\Gamma_t(x,\nu_t^\theta(x))\Gamma_t(x,\nu_t^\theta(x))^\top(\nabla_x^2 u^\theta)_t(x)\right) &= \operatorname{tr}\left(\left((C_t+D_tK_t)xx^\top(C_t+D_tK_t)^\top + DV_tD^\top\right)P_t\right).\end{aligned}$$

Using these representation and the definitions of P and φ shows that u solves (6.33) for a.e. $t \in [0,T]$ and all $x \in \mathbb{R}^d$, and satisfies the terminal condition $u_T(x) = \frac{1}{2}x^\top Gx$.

Applying Itô's formula to $t \mapsto u_t(X_t)$ yields

$$\begin{aligned}u_T(X_T) = u_0(X_0) + \int_0^T \Big(&(\tfrac{\mathrm{d}}{\mathrm{d}t}u)_t(X_t) + \frac{1}{2}\operatorname{tr}\left(\Gamma_t(X_t,\nu_t(X_t))\Gamma_t(X_t,\nu_t(X_t))^\top(\nabla_x^2 u)_t(X_t)\right)\\ &+ \Phi_t(X_t,\nu_t(X_t))^\top(\nabla_x u)_t(X_t)\Big)\,\mathrm{d}t + \int_0^T (\nabla_x u)_t(X_t)^\top\Gamma_t(X_t,\nu_t(X_t))\,\mathrm{d}W_t.\end{aligned} \tag{6.34}$$

Since $\nabla_x u_t = P_t x$ and by the integrability of C, D, θ and X, the diffusion term $\int_0^\cdot (\nabla_x u)_t(X_t)^\top\Gamma_t(X_t,\nu_t(X_t))\,\mathrm{d}W_t$ is a martingale. Thus, taking expectations on both sides of (6.34) and using the PDE (6.33) yields that

$$\begin{aligned}\mathbb{E}[u_0(\xi_0)] = \mathbb{E}\left[\frac{1}{2}X_T^\top GX_T\right] + \mathbb{E}\Bigg[\int_0^T \Bigg\{\frac{1}{2}\Bigg(&\left\langle\begin{pmatrix}Q_t & S_t^\top\\ S_t & R_t\end{pmatrix}\begin{pmatrix}X_t\\ K_tX_t\end{pmatrix},\begin{pmatrix}X_t\\ K_tX_t\end{pmatrix}\right\rangle + \operatorname{tr}(R_tV_t)\Bigg)\\ &+ \frac{\rho}{2}\left((K_tX_t)^\top\bar{V}_t^{-1}K_tX_t + \operatorname{tr}(\bar{V}_t^{-1}V_t) - k + \ln\left(\frac{\det(\bar{V}_t)}{\det(V_t)}\right)\right)\Bigg\}\,\mathrm{d}t\Bigg].\end{aligned} \tag{6.35}$$

Together with Lemma 2 this gives us $\mathcal{C}(\theta) = \mathbb{E}[u_0(\xi_0)]$. □

Using the previous result allows us to find a representation for the difference in costs $\mathcal{C}(\theta') - \mathcal{C}(\theta)$ for any parameters $\theta, \theta' \in \Theta$.

Lemma 4. *Suppose Assumption 3 holds. For any* $\theta \in \Theta$*, let* $P^\theta \in C([0,T];\mathbb{S}^d)$ *be a solution to* (6.9)*, and* $\Sigma^\theta \in C([0,T];\overline{\mathbb{S}_+^d})$ *to* (6.10)*. Then for all* $\theta, \theta' \in \Theta$*,*

$$\begin{aligned}\mathcal{C}(\theta') - \mathcal{C}(\theta) = \int_0^T \Big(&\langle K_t' - K_t, \mathcal{D}_K(\theta)_t\Sigma_t^{\theta'}\rangle + \frac{1}{2}\langle K_t' - K_t, (D_t^\top P_t^\theta D_t + R_t + \rho\bar{V}_t^{-1})(K_t' - K_t)\Sigma_t^{\theta'}\rangle\\ &+ \ell_t(V_t', P_t^\theta) - \ell_t(V_t, P_t^\theta)\Big)\,\mathrm{d}t,\end{aligned}$$

where $\mathcal{D}_K(\theta)_t$ *as in* (6.11)*, and* $\ell : [0,T]\times\mathbb{S}_+^k\times\mathbb{R}^{d\times d} \to \mathbb{R}$ *is given by*

$$\ell_t(V,Z) = \frac{1}{2}\left(\langle D_t^\top ZD_t + R_t + \rho\bar{V}_t^{-1}, V\rangle - \rho\ln(\det(V))\right) \quad \forall (t,V,Z) \in [0,T]\times\mathbb{S}_+^k\times\mathbb{R}^{d\times d}. \tag{6.36}$$

Proof. For this proof, we fix $\theta, \theta' \in \Theta$, and write for simplicity $(P,\Sigma) = (P^\theta, \Sigma^\theta)$, $(P',\Sigma') = (P^{\theta'}, \Sigma^{\theta'})$, $u = u^\theta$ and $u' = u^{\theta'}$, where for each $\theta \in \Theta$, $u^\theta : [0,T]\times\mathbb{R}^d \to \mathbb{R}$ is defined as in Lemma 3. Using equation (6.33), yields for the difference, for all $x \in \mathbb{R}^d$, $(u'-u)_T(x) = 0$,

$$\begin{aligned}(\tfrac{\mathrm{d}}{\mathrm{d}t}(u'-u))_t + \frac{1}{2}\operatorname{tr}\Big(&\Gamma_t(x,\nu_t^{\theta'}(x))\Gamma_t(x,\nu_t^{\theta'}(x))^\top(\nabla_x^2(u'-u))_t(x)\Big)\\ &+ \Phi_t(x,\nu_t^{\theta'}(x))^\top(\nabla_x(u'-u))_t(x) + F_t(x) = 0, \quad \text{a.e. } t \in [0,T],\end{aligned} \tag{6.37}$$

where $F : [0,T] \times \mathbb{R} \to \mathbb{R}$ is given by

$$\begin{aligned}
F_t(x) = \frac{1}{2}\mathrm{tr}\left(\Gamma_t(x, \nu_t^{\theta'}(x))\Gamma_t(x, \nu_t^{\theta'}(x))^\top (\nabla_x^2 u)_t(x)\right) + \Phi_t(x, \nu_t^{\theta'}(x))^\top (\nabla_x u)_t(x) \\
+ \frac{1}{2}\Big[\left(x^\top Q_t x + x^\top S_t^\top K_t' x + (K_t' x)^\top S_t x + (K_t' x)^\top R_t K_t' x + \mathrm{tr}(R_t V_t')\right) \\
- \left(x^\top Q_t x + x^\top S_t^\top K_t x + (K_t x)^\top S_t x + (K_t x)^\top R_t K_t x + \mathrm{tr}(R_t V_t)\right) \Big] \\
+ \frac{\rho}{2}\Big[\left((K_t' x)^\top \bar{V}_t^{-1} K_t' x + \mathrm{tr}(\bar{V}_t^{-1} V_t') - \ln\left(\det(V_t')\right)\right) \\
- \left((K_t x)^\top \bar{V}_t^{-1} K_t x + \mathrm{tr}(\bar{V}_t^{-1} V_t) - \ln\left(\det(V_t)\right)\right) \Big].
\end{aligned}$$

Applying Itô's formula to the difference $t \mapsto (u' - u)_t(X_t^{\theta'})$ and using equation (6.37) shows that

$$\mathbb{E}[(u' - u)_T(X_T^{\theta'})] - \mathbb{E}[(u' - u)_0(X_0^{\theta'})] = \mathbb{E}\left[\int_0^T -F_t(X_t^{\theta'})\,\mathrm{d}t\right].$$

With the representation $\mathcal{C}(\theta) = \mathbb{E}[u_0^\theta(\xi_0)]$ from Lemma 3 and the terminal condition $(u' - u)_T = 0$ we have

$$\mathcal{C}(\theta') - \mathcal{C}(\theta) = \mathbb{E}\left[\int_0^T F_t(X_t^{\theta'})\,\mathrm{d}t\right]. \tag{6.38}$$

To make the expression for $F_t(x)$ more manageable we rewrite it in terms of a modified Hamiltonian and the function ℓ. For that purpose, let $\overline{H} : [0,T] \times \mathbb{R}^d \times \mathbb{R}^k \times \mathbb{R}^d \times \mathbb{R}^{d\times d} \to \mathbb{R}$ denote the modified Hamiltonian where for $(t,x,a,y,z) \in [0,T] \times \mathbb{R}^d \times \mathbb{R}^k \times \mathbb{R}^d \times \mathbb{R}^{d\times d}$

$$\begin{aligned}
\overline{H}_t(x,a,y,z) = \tfrac{1}{2}\mathrm{tr}\left((C_t x + D_t a)(C_t x + D_t a)^\top z\right) + \langle A_t x + B_t a, y\rangle \\
+ \tfrac{1}{2}\left(x^\top Q_t x + x^\top S_t^\top a + a^\top S_t x + a^\top (R_t + \rho \bar{V}_t^{-1}) a\right).
\end{aligned}$$

Consider $\ell : [0,T] \times \mathbb{S}_+^k \times \mathbb{R}^{d\times d} \to \mathbb{R}$ as defined in (6.36). Since $(\nabla_x u)_t(x) = P_t x$ and $(\nabla_x^2 u)_t(x) = P_t$, Lemma 2 yields

$$F_t(x) = \overline{H}_t(x, K_t' x, P_t x, P_t) - \overline{H}_t(x, K_t x, P_t x, P_t) + \ell_t(V_t', P_t) - \ell_t(V_t, P_t). \tag{6.39}$$

Note that for all $(t,x,y,z) \in [0,T] \times \mathbb{R}^k \times \mathbb{R}^d \times \mathbb{S}^d$, $a \mapsto \overline{H}_t(x,a,y,z)$ is quadratic. This means the difference between to modified Hamiltonian can be expressed by a second-order Taylor's approximation: for any $a, a \in \mathbb{R}^k$,

$$\overline{H}_t(x,a',y,z) - \overline{H}_t(x,a,y,z) = \langle a' - a, \partial_a \overline{H}_t(x,a,y,z)\rangle + \frac{1}{2}\langle a' - a, \partial_a^2 \overline{H}_t(x,a,y,z)(a' - a)\rangle,$$

The derivatives $\partial_a \overline{H}_t(x,a,y,z)$ and $\partial_a^2 \overline{H}_t(x,a,y,z)$ are given by

$$\begin{aligned}
\partial_a \overline{H}_t(x,a,y,z) &= D_t^\top z(C_t x + D_t a) + B_t^\top y + S_t x + (R_t + \rho \bar{V}_t^{-1}) a, \\
\partial_a^2 \overline{H}_t(x,a,y,z) &= D_t^\top z D_t + R_t + \rho \bar{V}_t^{-1}.
\end{aligned}$$

Replacing the difference in (6.39) with the second-order Taylor approximation yields

$$\begin{aligned}
F_t(x) = \ell_t(V_t', P_t) - \ell_t(V_t, P_t) \\
+ \langle (K_t' - K_t)x, D_t^\top P_t (C_t x + D_t K_t x) + B_t^\top P_t x + S_t x + (R_t + \rho \bar{V}_t^{-1}) K_t x\rangle \\
+ \frac{1}{2}\langle (K_t' - K_t)x, (D_t^\top P_t D_t + R_t + \rho \bar{V}_t^{-1})(K_t' - K_t)x\rangle.
\end{aligned}$$

Combining this result with (6.38), the definition of $\mathcal{D}_K(\theta)$ in (6.11), and $\Sigma'_t = \mathbb{E}[X_t^{\theta'}(X_t^{\theta'})^\top]$ concludes the proof. □

With this work we may prove Proposition 3 from Section 6.3.

Proof of Proposition 3. Using the representation (6.35) we have for each $\theta \in \Theta$ that

$$\begin{aligned}\mathcal{C}(\theta) =& \frac{1}{2}\int_0^T \Big(\operatorname{tr}\left((Q_t + K_t^\top S_t + S_t^\top K_t + K_t^\top(R_t + \rho\bar{V}_t^{-1})K_t)\Sigma_t^\theta\right) \\ &+ \operatorname{tr}(R_t V_t) + \rho\left(\operatorname{tr}(\bar{V}_t^{-1}V_t) - k + \ln\left(\frac{\det(\bar{V}_t)}{\det(V_t)}\right)\right)\Big)\mathrm{d}t + \frac{1}{2}\operatorname{tr}(G\Sigma_T^\theta),\end{aligned} \tag{6.40}$$

where $\Sigma^\theta \in C([0,T];\overline{\mathbb{S}_+^d})$ is the solution to (6.10). Using the result from [36, Corollary 4.11] we can characterise the Gateaux derivatives of the cost function by the corresponding Hamiltonian and adjoint process. For that purpose consider the Hamiltonian for (6.40)-(6.10) $H : [0,T] \times \overline{\mathbb{S}_+^d} \times \mathbb{R}^{k\times d} \times \mathbb{S}_+^k \times \mathbb{R}^d \to \mathbb{R}$ where for all $(t,\Sigma,K,V,Y) \in [0,T] \times \overline{\mathbb{S}_+^d} \times \mathbb{R}^{k\times d} \times \mathbb{S}_+^k \times \mathbb{R}^{d\times d}$,

$$\begin{aligned}H_t(\Sigma,K,V,Y) =& \langle (A_t + B_tK)\Sigma + \Sigma(A_t + B_tK)^\top + (C_t + D_tK)\Sigma(C_t + D_tK)^\top + D_tVD_t^\top, Y\rangle \\ &+ \frac{1}{2}\Big\{\operatorname{tr}\left((Q_t + K^\top S_t + S_t^\top K + K^\top(R_t + \rho\bar{V}_t^{-1})K)\Sigma\right) + \operatorname{tr}(R_tV) \\ &+ \rho\left(\operatorname{tr}(\bar{V}_t^{-1}V) - k + \ln\left(\frac{\det(\bar{V}_t)}{\det(V)}\right)\right)\Big\},\end{aligned}$$

Furthermore, for each $\theta \in \Theta$, the adjoint process $Y^\theta \in C([0,T];\mathbb{R}^{d\times d})$ satisfies

$$(\tfrac{\mathrm{d}}{\mathrm{d}t}Y)_t = -\partial_\Sigma H_t(\Sigma_t^\theta, K_t, V_t, Y_t), \quad \text{a.e. } t \in [0,T]; \quad Y_T = \tfrac{1}{2}G.$$

For any $\theta \in \Theta$, [36, Corollary 4.11] then yields

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}\varepsilon}\mathcal{C}(K + \varepsilon K', V)\Big|_{\varepsilon=0} &= \int_0^T \langle \partial_K H_t(\Sigma_t^\theta, K_t, V_t, Y_t^\theta), K_t'\rangle\,\mathrm{d}t, \\ \frac{\mathrm{d}}{\mathrm{d}\varepsilon}\mathcal{C}(K, V + \varepsilon(V' - V))\Big|_{\varepsilon=0} &= \int_0^T \langle \partial_V H_t(\Sigma_t^\theta, K_t, V_t, Y_t^\theta), V_t' - V_t\rangle\,\mathrm{d}t.\end{aligned}$$

Direct calculation gives us for all $(t,\Sigma,K,V,Y) \in [0,T] \times \overline{\mathbb{S}_+^d} \times \mathbb{R}^{k\times d} \times \mathbb{S}_+^k \times \mathbb{S}^d$, that

$$\begin{aligned}\partial_K H_t(\Sigma,K,V,Y) &= \left(2B_t^\top Y + 2D_t^\top Y(C_t + D_tK) + S_t + (R_t + \rho\bar{V}_t^{-1})K\right)\Sigma, \\ \partial_V H_t(\Sigma,K,V,Y) &= D_t^\top Y D_t + \tfrac{1}{2}(R_t + \rho(\bar{V}_t^{-1} - V^{-1})).\end{aligned}$$

Since $Y^\theta = \frac{1}{2}P^\theta \in C([0,T];\mathbb{S}^d)$, the proof is complete. □

Using the characterisation of the gradients from Proposition 3 and the representation of the differences in cost (Lemma 4) we can derive the Łojasiewicz condition for the optimization problem (Proposition 4).

Proof of Proposition 4. First, we introduce a scalar product identity that can be obtained from direct computations. For that purpose, let $Z,\Gamma \in \mathbb{R}^{k\times d}$, $\Sigma \in \overline{\mathbb{S}_+^k}$ and $M \in \mathbb{S}_+^k$,

$$\begin{aligned}\langle Z,\Gamma\Sigma\rangle + \frac{1}{2}\langle Z, MZ\Sigma\rangle &= \frac{1}{2}\left\langle Z + M^{-1}\Gamma, M\left(Z + M^{-1}\Gamma\right)\Sigma\right\rangle - \frac{1}{2}\langle M^{-1}\Gamma,\Gamma\Sigma\rangle \\ &\geq -\frac{1}{2}\langle M^{-1}\Gamma,\Gamma\Sigma\rangle.\end{aligned} \tag{6.41}$$

The last inequality holds since $M, \Sigma \in \overline{\mathbb{S}^d_+}$ and $\operatorname{tr}(AB) \geq 0$ if $A, B \in \overline{\mathbb{S}^d_+}$. This inequality can be applied as a bound for parts of the cost difference from Lemma 4. For that purpose, for any $\theta \in \Theta$ and $t \in [0,T]$, use (6.41) with $Z = K_t^\star - K_t$, $\Gamma = \mathcal{D}_K(\theta)_t$, $\Sigma = \Sigma_t^{\theta^\star}$ and $M = D_t^\top P_t^\theta D_t + R_t + \rho \bar{V}_t^{-1}$. This yields that

$$\begin{aligned}
&\int_0^T \left(\langle K_t^\star - K_t, \mathcal{D}_K(\theta)_t \Sigma_t^{\theta^\star} \rangle + \frac{1}{2} \langle K_t^\star - K_t, (D_t^\top P_t^\theta D_t + R_t + \rho \bar{V}_t^{-1})(K_t^\star - K_t) \Sigma_t^{\theta^\star} \rangle \right) \mathrm{d}t \\
&\quad \geq -\frac{1}{2} \int_0^T \langle (D_t^\top P_t^\theta D_t + R_t + \rho \bar{V}_t^{-1})^{-1} \mathcal{D}_K(\theta)_t, \mathcal{D}_K(\theta)_t \Sigma_t^{\theta^\star} \rangle \, \mathrm{d}t.
\end{aligned} \tag{6.42}$$

Applying (6.42) to the full cost difference gives

$$\begin{aligned}
&\mathcal{C}(\theta^\star) - \mathcal{C}(\theta) \\
&\quad = \int_0^T \Big(\langle K_t^\star - K_t, \mathcal{D}_K(\theta)_t \Sigma_t^{\theta^\star} \rangle + \frac{1}{2} \langle K_t^\star - K_t, (D_t^\top P_t^\theta D_t + R_t + \rho \bar{V}_t^{-1})(K_t^\star - K_t) \Sigma_t^{\theta^\star} \rangle \\
&\qquad + \ell_t(V_t^\star, P_t^\theta) - \ell_t(V_t, P_t^\theta) \Big) \, \mathrm{d}t \\
&\quad \geq \int_0^T \left(-\frac{1}{2} \langle (D_t^\top P_t^\theta D_t + R_t + \rho \bar{V}_t^{-1})^{-1} \mathcal{D}_K(\theta)_t, \mathcal{D}_K(\theta)_t \Sigma_t^{\theta^\star} \rangle + \ell_t(V_t^\star, P_t^\theta) - \ell_t(V_t, P_t^\theta) \right) \mathrm{d}t.
\end{aligned} \tag{6.43}$$

Furthermore, with (6.36), for any $(t, Z) \in [0,T] \times \mathbb{R}^{d \times d}$ and $V, V' \in \mathbb{S}^k_+$, it holds that

$$\begin{aligned}
&\ell_t(V', Z) - \ell_t(V, Z) \\
&\quad = \langle \partial_V \ell_t(V, Z), V' - V \rangle + \int_0^1 \left(\tfrac{\mathrm{d}}{\mathrm{d}s} \ell_t(V + s(V' - V), Z) - \langle \partial_V \ell_t(V, Z), V' - V \rangle \right) \mathrm{d}s \\
&\quad = \langle \partial_V \ell_t(V, Z), V' - V \rangle + \int_0^1 \langle \partial_V \ell_t(V + s(V' - V), Z) - \partial_V \ell_t(V, Z), V' - V \rangle \, \mathrm{d}s.
\end{aligned}$$

Recall that we also have that $\partial_V \ell_t(V, Z) = \frac{1}{2}(D_t^\top Z D_t + R_t + \rho \bar{V}_t^{-1} - \rho V^{-1})$, and by direct calculation $A^{-1} - B^{-1} = B^{-1}(B - A)A^{-1}$ for all $A, B \in \mathbb{S}^k_+$. Combining these for any $(t, Z) \in [0,T] \times \mathbb{R}^{d \times d}$ and $V, V' \in \mathbb{S}^k_+$, yields

$$\begin{aligned}
&\ell_t(V', Z) - \ell_t(V, Z) \\
&\quad = \langle \partial_V \ell_t(V, Z), V' - V \rangle + \frac{\rho}{2} \int_0^1 \langle V^{-1} \big(s(V' - V) \big) (V + s(V' - V))^{-1}, V' - V \rangle \, \mathrm{d}s.
\end{aligned} \tag{6.44}$$

Next, for any $\theta, \theta' \in \Theta$ and $t \in [0,T]$, we have that

$$\begin{aligned}
&\ell_t(V_t', P_t^\theta) - \ell_t(V_t, P_t^\theta) \\
&\quad = \langle \partial_V \ell_t(V_t, P_t^\theta), V_t' - V_t \rangle + \frac{\rho}{2} \int_0^1 \langle V_t^{-1} \big(s(V_t' - V_t) \big) (V_t + s(V_t' - V_t))^{-1}, V_t' - V_t \rangle \, \mathrm{d}s \\
&\quad \geq \langle \mathcal{D}_V(\theta)_t, V_t' - V_t \rangle + \frac{\rho}{4} \Lambda(V_t', V_t)^2 \langle V_t' - V_t, V_t' - V_t \rangle \geq -\frac{1}{\rho \Lambda(V_t', V_t)^2} |\mathcal{D}_V(\theta)_t|^2,
\end{aligned} \tag{6.45}$$

where the first inequality is due to (6.12), the fact that $\operatorname{tr}(AB) \geq 0$ for all $A, B \in \overline{\mathbb{S}^d_+}$ and since $\Lambda(V_t', V_t) > 0$ defined as

$$\begin{aligned}
\Lambda(V_t', V_t) &\coloneqq \min_{s \in [0,1]} \lambda_{\min} \left((V_t + s(V_t' - V_t))^{-1} \right) = \frac{1}{\max_{s \in [0,1]} \lambda_{\max} \left(V_t + s(V_t' - V_t) \right)} \\
&= \frac{1}{\max(\lambda_{\max}(V_t), \lambda_{\max}(V_t'))} = \frac{1}{\max(\|V_t\|_2, \|V_t'\|_2)},
\end{aligned}$$

where we used the convexity of $[0,1] \ni s \mapsto \lambda_{\max}\left(V_t + s(V_t' - V_t)\right) \in \mathbb{R}$, and that $\|V\|_2 = \lambda_{\max}(V)$ for all $V \in \overline{\mathbb{S}_+^k}$. The second inequality in (6.45) is an application of (6.41) with $Z = V_t' - V_t$, $\Gamma = \mathcal{D}_V(\theta)_t$, $\Sigma = I_k$ and $M = \frac{\rho}{2}\Lambda(V_t', V_t)^2 I_k$. Considering (6.45) with $V' = V^\star$ and substituting in (6.43) yields the Łojasiewicz condition (6.13). □

Finally, we proof the almost smoothness condition by using Lemma 4 and use a similar strategy for ℓ as in the previous proof.

Proof of Proposition 5. With (6.12) and (6.44), we have for any $\theta, \theta' \in \Theta$ and $t \in [0,T]$ that

$$\begin{aligned}
&\ell_t(V_t', P_t^\theta) - \ell_t(V_t, P_t^\theta) \\
&\quad = \langle \partial_V \ell_t(V_t, P_t^\theta), V_t' - V_t\rangle + \frac{\rho}{2}\int_0^1 \langle V_t^{-1}\big(s(V_t' - V_t)\big)(V_t + s(V_t' - V_t))^{-1}, V_t' - V_t\rangle \,\mathrm{d}s \\
&\quad \leq \langle \mathcal{D}_V(\theta)_t, V_t' - V_t\rangle + \frac{\rho}{4}\overline{\Lambda}(V_t', V_t)^2 \langle V_t' - V_t, V_t' - V_t\rangle,
\end{aligned}$$

where $\overline{\Lambda}(V_t', V_t) > 0$ is given by

$$\begin{aligned}
\overline{\Lambda}(V_t', V_t) &\coloneqq \max_{s\in[0,1]} \lambda_{\max}\left((V_t + s(V_t' - V_t))^{-1}\right) = \frac{1}{\min_{s\in[0,1]} \lambda_{\min}\left(V_t + s(V_t' - V_t)\right)} \\
&= \frac{1}{\min(\lambda_{\min}(V_t), \lambda_{\min}(V_t'))}
\end{aligned}$$

using the same considerations as in the proof of Proposition 4. Combining this inequality with Lemma 4 concludes the proof. □

Chapter 7

Summary and outlook

In this thesis, we presented several advances for the use of reinforcement learning in LOBs. Chapter 3 was concerned with providing a novel data-driven simulation approach for off-policy evaluation and provided convergence guarantees for the value function in a general stochastic control setting. In Chapter 4, we applied this algorithm to simulate limit order books and provided evidence for its effectiveness by considering several stylized facts and comparing the results to other simulation approaches. We showed how our method may be used for evaluating trading strategies in a pro-rata LOB and how it can be used to calibrate an execution strategy. Chapter 5 explores the usage of policy gradient methods for optimal execution with limit orders and discusses several challenges. First, we applied a zeroth-order gradient scheme for a parametric model of a LOB. Second, we introduced a generalization of a realistic LOB GAN simulator that allows for strategy evaluation with limit orders and is differentiable with respect to the trading strategy. In this realistic LOB simulator, we presented a policy gradient algorithm using path-wise gradient estimates obtained by automated differentiation and showed the results of the learned strategy. In Chapter 6, we analysed the loss-landscape of a continuous-time, episodic and stochastic LQC problem to provide sufficient conditions for the convergence of a policy gradient method and considered the impact of action frequency on the effectiveness of policy gradients. We provided numerical evidence that rescaling of gradients enhances the algorithms robustness against different discretizations.

Outlook While we discussed some possible extensions in the individual chapters, we believe that this thesis highlights several future research directions and overarching themes in connection to applications of reinforcement learning in limit order books.

One of these themes is the role of causality in dynamical systems and the usage of observational data to extract causal information. For example, one can view a multiperiod asset execution strategy in a LOB as a multi-stage treatment within a dynamical system. Taking this view allows the usage of causal techniques such as A/B-testing in this context [219]. Furthermore, by fitting a LOB model, either parametric or non-parametric, to data, one implicitly assumes to extract causal relationships from observational data. We believe that there is value in examining these assumptions more explicitly from the perspective of causal analysis. For example, it would be of interest how A/B-testing and other causal approaches can be used to validate generative LOB models. Furthermore, systematically approaching trading signals from a causal perspective (by e.g. building DAG-models [170]) could improve the understanding for investment and trading decisions.

Related to the problem of causality, is the challenge to find the correct state representation for a high-dimensional dynamical system. From the perspective of generative models, neglecting important state variables may lead to a systematic error in the model while including unnecessary ones may increase noise. Thus, we believe that methodical approaches to identifying a suitable state space could be highly beneficial for generative models of real-world dynamical systems. From the perspective of reinforcement learning, it would be particularly important to identify a Markovian representation due to the Markovian assumptions for many algorithms. In

the case of LOBs this could, for example, imply to include statistics on the book's recent history.

Given a state representation for a real-world dynamical system, it is often unclear what is the right metric on the state space. For example, some variables in the state representation may contain more relevant information on the system's future evolution. Hence, from the perspective of algorithms relying on metric information (such as in Chapter 3), it would be beneficial to be able to learn metrics that take this into account. Furthermore, including information on the metric of the state/parameter space in algorithm design may improve learning outcomes and stabilize the learning procedure (as in Chapter 6).

Finally, from a practitioners perspective, we believe that there are several extensions of interest for the application of reinforcement learning to LOBs. First, one could work on further improving the quality of LOB simulators and extending the evaluating capacities to other LOB execution mechanisms such as first-in-first-out order books. Second, a systematic study of different reinforcement learning algorithms such as actor-critic methods or value-based methods on realistic LOB simulators may lead to improved trading strategies. Finally, we believe that there are many more possible problem formulations for trading in limit order book markets that are worth investigating under the paradigm of using data-driven simulators and reinforcement learning. For example, one could consider different cost objectives such as in the market making problem, the inclusion of α-signals, the usage of both market and limit orders, or execution in a cross-asset context for related products.

Bibliography

[1] Frédéric Abergel and Aymen Jedidi. "A mathematical approach to order book modeling". In: *International Journal of Theoretical and Applied Finance* 16.05 (2013), pp. 1–40.

[2] Jacob Abernethy and Satyen Kale. "Adaptive market making via online learning". In: *Advances in Neural Information Processing Systems* 26 (2013).

[3] Robert Almgren and Neil Chriss. "Optimal execution of portfolio transactions". In: *Journal of Risk* 3 (2001), pp. 5–40.

[4] Hans Wilhelm Alt. "Linear functional analysis". In: *An Application-oriented Introduction* (2016).

[5] Sigrún Andradóttir, Daniel P Heyman, and Teunis J Ott. "On the choice of alternative measures in importance sampling with Markov chains". In: *Operations research* 43.3 (1995), pp. 509–519.

[6] Ludwig Arnold. "Random dynamical systems". In: *Dynamical systems* (1995), pp. 1–43.

[7] Christopher G Atkeson, Andrew W Moore, and Stefan Schaal. "Locally weighted learning". In: *Artificial Intelligence Review 11* (1997), pp. 11–73.

[8] Christopher G Atkeson, Andrew W Moore, and Stefan Schaal. "Locally weighted learning for control". In: *Artificial Intelligence Review 11* (1997), pp. 75–113.

[9] Marco Avellaneda and Sasha Stoikov. "High-frequency trading in a limit order book". In: *Quantitative Finance* 8.3 (2008), pp. 217–224.

[10] Emmanuel Bacry, Adrian Iuga, Matthieu Lasnier, and Charles-Albert Lehalle. "Market impacts and the life cycle of investors orders". In: *Market Microstructure and Liquidity* 1.02 (2015), p. 1550009.

[11] Emmanuel Bacry, Iacopo Mastromatteo, and Jean-François Muzy. "Hawkes processes in finance". In: *Market Microstructure and Liquidity* 1.01 (2015), p. 1550005.

[12] Leemon C Baird and A Harry Klopf. "Reinforcement learning with high-dimensional continuous actions". In: *Wright Laboratory, Wright-Patterson Air Force Base, Tech. Rep. WL-TR-93-1147* 15 (1993).

[13] Bharathan Balaji, Jordan Bell-Masterson, Enes Bilgin, Andreas Damianou, Pablo Moreno Garcia, Arpit Jain, Runfei Luo, Alvaro Maggiar, Balakrishnan Narayanaswamy, and Chun Ye. "Orl: Reinforcement learning benchmarks for online stochastic optimization problems". In: *arXiv preprint arXiv:1911.10641* (2019).

[14] Robert L Barron. "Adaptive flight control systems". In: *NATO AGARD Bionics Symp., Brussels, Belgium, Sept. 1968.* 1970.

[15] Emilio Barucci, Adrien Mathieu, and Leandro Sánchez-Betancourt. "Market Making with Fads, Informed, and Uninformed Traders". In: *arXiv preprint arXiv:2501.03658* (2025).

[16] Christopher Bayliss, Christine SM Currie, Julia A Bennell, and Antonio Martinez-Sykora. "Dynamic pricing for vehicle ferries: using packing and simulation to optimize revenues". In: *European Journal of Operational Research* 273.1 (2019), pp. 288–304.

[17] Erhan Bayraktar and Michael Ludkovski. "Liquidation in limit order books with controlled intensity". In: *Mathematical Finance* 24.4 (2014), pp. 627–650.

[18] Peter Belcak, Jan-Peter Calliess, and Stefan Zohren. "Fast agent-based simulation framework of limit order books with applications to pro-rata markets and the study of latency effects". In: *arXiv preprint arXiv:2008.07871* (2020).

[19] Claudio Bellani, Damiano Brigo, Mikko S Pakkanen, and Leandro Sánchez-Betancourt. "Price Impact Without Averaging". In: *Applied Mathematical Finance* 30.4 (2023), pp. 175–206.

[20] Jon Louis Bentley. "Multidimensional binary search trees used for associative searching". In: *Communications of the ACM* 18.9 (1975), pp. 509–517.

[21] Dimitri Bertsekas. *Dynamic programming and optimal control: Volume I*. Vol. 1. Athena scientific, 2012.

[22] Dimitri Bertsekas. *Reinforcement learning and optimal control*. Athena Scientific, 2019.

[23] Dimitri Bertsekas and Steven E Shreve. *Stochastic optimal control: the discrete-time case*. Vol. 5. Athena Scientific, 1996.

[24] Hans-Georg Beyer and Hans-Paul Schwefel. "Evolution strategies – a comprehensive introduction". In: *Natural Computing* 1.1 (2002), pp. 3–52.

[25] Jalaj Bhandari and Daniel Russo. "Global optimality guarantees for policy gradient methods". In: *arXiv preprint arXiv:1906.01786* (2019).

[26] Rabi Bhattacharya and Mukul Majumdar. "Random dynamical systems: a review". In: *Economic Theory* 23.1 (2003), pp. 13–38.

[27] Rabi N. Bhattacharya and Edward C. Waymire. *Stochastic Processes with Applications*. Society for Industrial and Applied Mathematics, 2009. DOI: 10.1137/1.9780898718997. eprint: https://epubs.siam.org/doi/pdf/10.1137/1.9780898718997. URL: https://epubs.siam.org/doi/abs/10.1137/1.9780898718997.

[28] Bruno Biais, Pierre Hillion, and Chester Spatt. "An empirical analysis of the limit order book and the order flow in the Paris Bourse". In: *The Journal of Finance* 50.5 (1995), pp. 1655–1689.

[29] Raghu Bollapragada and Stefan M Wild. "Adaptive Sampling Quasi-Newton Methods for Zeroth-Order Stochastic Optimization". In: *arXiv preprint arXiv:2109.12213* (2021).

[30] Léon Bottou, Frank E Curtis, and Jorge Nocedal. "Optimization methods for large-scale machine learning". In: *SIAM review* 60.2 (2018), pp. 223–311.

[31] Jean-Philippe Bouchaud, Marc Mézard, and Marc Potters. "Statistical properties of stock order books: empirical results and models". In: *Quantitative Finance* 2.4 (2002), p. 251.

[32] Jingjing Bu, Afshin Mesbahi, and Mehran Mesbahi. "Policy gradient-based algorithms for continuous-time linear quadratic control". In: *arXiv preprint arXiv:2006.09178* (2020).

[33] Hans Buehler, Lukas Gonon, Josef Teichmann, and Ben Wood. "Deep hedging". In: *Quantitative Finance* 19.8 (2019), pp. 1271–1291.

[34] David Byrd, Maria Hybinette, and Tucker Hybinette Balch. "Abides: Towards high-fidelity market simulation for AI research". In: *arXiv preprint arXiv:1904.12066* (2019).

[35] Ramiro Camino, Christian Hammerschmidt, and Radu State. "Generating multi-categorical samples with generative adversarial networks". In: *arXiv preprint arXiv:1807.01202* (2018).

[36] René Carmona. *Lectures on BSDEs, stochastic control, and stochastic differential games with financial applications*. SIAM, 2016.

[37] Alvaro Cartea, Ryan Donnelly, and Sebastian Jaimungal. "Enhancing trading strategies with order book signals". In: *Applied Mathematical Finance* 25.1 (2018), pp. 1–35.

[38] Alvaro Cartea and Sebastian Jaimungal. "Optimal execution with limit and market orders". In: *Quantitative Finance* 15.8 (2015), pp. 1279–1291.

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

[40] Alvaro Cartea, Sebastian Jaimungal, and Jason Ricci. "Algorithmic trading, stochastic control, and mutually exciting processes". In: *SIAM Review* 60.3 (2018), pp. 673–703.

[41] Álvaro Cartea, Sebastian Jaimungal, and Jason Ricci. "Buy low, sell high: A high frequency trading perspective". In: *SIAM Journal on Financial Mathematics* 5.1 (2014), pp. 415–444.

[42] Tolga Cenesizoglu and Gunnar Grass. "Bid-and ask-side liquidity in the NYSE limit order book". In: *Journal of Financial Markets* 38 (2018), pp. 14–38.

[43] Arthur Charpentier, Romuald Elie, and Carl Remlinger. "Reinforcement learning in economics and finance". In: *Computational Economics* (2021), pp. 1–38.

[44] Yue Chen, Ana Bušić, and Sean P Meyn. "State estimation for the individual and the population in mean field control with application to demand dispatch". In: *IEEE Transactions on Automatic Control* 62.3 (2016), pp. 1138–1149.

[45] CME. *Three-Month SOFR*. `https://www.cmegroup.com/markets/interest-rates/stirs/three-month-sofr.html`. Accessed: 2024-02-23. 2024.

[46] Andrea Coletta, Joseph Jerome, Rahul Savani, and Svitlana Vyetrenko. "Conditional Generators for Limit Order Book Environments: Explainability, Challenges, and Robustness". In: *arXiv preprint arXiv:2306.12806* (2023).

[47] Andrew R Conn, Katya Scheinberg, and Luis N Vicente. *Introduction to derivative-free optimization*. SIAM, 2009.

[48] Rama Cont, Mihai Cucuringu, Jonathan Kochems, and Felix Prenzel. "Limit Order Book Simulation with Generative Adversarial Networks". In: *Available at SSRN 4512356* (2023).

[49] Rama Cont and Adrien De Larrard. "Price dynamics in a Markovian limit order market". In: *SIAM Journal on Financial Mathematics* 4.1 (2013), pp. 1–25.

[50] Rama Cont and Marvin S Muller. "A stochastic partial differential equation model for limit order book dynamics". In: *SIAM Journal on Financial Mathematics* 12.2 (2021), pp. 744–787.

[51] Rama Cont, Sasha Stoikov, and Rishi Talreja. "A stochastic model for order book dynamics". In: *Operations Research* 58.3 (2010), pp. 549–563.

[52] Robert Dadashi, Shideh Rezaeifar, Nino Vieillard, Léonard Hussenot, Olivier Pietquin, and Matthieu Geist. "Offline reinforcement learning with pseudometric learning". In: *International Conference on Machine Learning*. PMLR. 2021, pp. 2307–2318.

[53] Christoph Dann, Gerhard Neumann, Jan Peters, et al. "Policy evaluation with temporal differences: A survey and comparison". In: *Journal of Machine Learning Research* 15 (2014), pp. 809–883.

[54] Miroslav Dudík, John Langford, and Lihong Li. "Doubly robust policy evaluation and learning". In: *Proceedings of the 28th International Conference on International Conference on Machine Learning*. 2011, pp. 1097–1104.

[55] Zoltan Eisler, Jean-Philippe Bouchaud, and Julien Kockelkoren. "The price impact of order book events: market orders, limit orders and cancellations". In: *Quantitative Finance* 12.9 (2012), pp. 1395–1419.

[56] *Eurex Market Models*. `https://www.eurex.com/ex-en/trade/market-models`. Accessed: 2024-12-04.

[57] Maryam Fazel, Rong Ge, Sham Kakade, and Mehran Mesbahi. "Global convergence of policy gradient methods for the linear quadratic regulator". In: *International Conference on Machine Learning*. PMLR. 2018, pp. 1467–1476.

[58] Yihao Feng, Lihong Li, and Qiang Liu. "A kernel loss for solving the bellman equation". In: *Advances in Neural Information Processing Systems* 32 (2019).

[59] Jonathan Field and J Large. "Pro-rata matching and one-tick markets". In: *Unpublished paper: Oxford-Man Institute* (2008).

[60] Dena Firoozi and Sebastian Jaimungal. "Exploratory LQG mean field games with entropy regularization". In: *Automatica* 139 (2022), p. 110177.

[61] Abraham D Flaxman, Adam Tauman Kalai, and H Brendan McMahan. "Online convex optimization in the bandit setting: gradient descent without a gradient". In: *arXiv preprint cs/0408007* (2004).

[62] Raphael Fonteneau, Susan Murphy, Louis Wehenkel, and Damien Ernst. "Model-free monte carlo-like policy evaluation". In: *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*. JMLR Workshop and Conference Proceedings. 2010, pp. 217–224.

[63] Raphael Fonteneau, Susan A Murphy, Louis Wehenkel, and Damien Ernst. "Batch mode reinforcement learning based on the synthesis of artificial trajectories". In: *Annals of Operations Research* 208 (2013), pp. 383–416.

[64] Justin Fu et al. "Benchmarks for Deep Off-Policy Evaluation". In: *International Conference on Learning Representations*. 2021. URL: https://openreview.net/forum?id=kWSeGEeHvF8.

[65] Guillermo Gallego and Garrett Van Ryzin. "Optimal dynamic pricing of inventories with stochastic demand over finite horizons". In: *Management Science* 40.8 (1994), pp. 999–1020.

[66] Bruno Gašperov and Zvonko Kostanjčar. "Market making with signals through deep reinforcement learning". In: *IEEE access* 9 (2021), pp. 61611–61622.

[67] Jim Gatheral. "No-dynamic-arbitrage and market impact". In: *Quantitative Finance* 10.7 (2010), pp. 749–759.

[68] Maximilien Germain, Huyen Pham, and Xavier Warin. "Approximation error analysis of some deep backward schemes for nonlinear PDEs". In: *SIAM Journal on Scientific Computing* 44.1 (2022), A28–A56.

[69] Michael Giegrich, Roel Oomen, and Christoph Reisinger. "K-Nearest-Neighbor Resampling for Off-Policy Evaluation in Stochastic Control". In: *arXiv preprint arXiv:2306.04836* (2023).

[70] Michael Giegrich, Roel Oomen, and Christoph Reisinger. "Limit Order Book Simulation and Trade Evaluation with K-Nearest-Neighbor Resampling". In: *arXiv preprint arXiv:2409.06514* (2024).

[71] Michael Giegrich, Christoph Reisinger, and Yufei Zhang. "Convergence of policy gradient methods for finite-horizon exploratory linear-quadratic control problems". In: *SIAM Journal on Control and Optimization* 62.2 (2024), pp. 1060–1092.

[72] Mike Giles and Paul Glasserman. "Smoking adjoints: Fast Monte Carlo greeks". In: *Risk* 19.1 (2006), pp. 88–92.

[73] Paul Glasserman. *Monte Carlo methods in financial engineering*. Vol. 53. Springer, 2004.

[74] Xavier Glorot and Yoshua Bengio. "Understanding the difficulty of training deep feedforward neural networks". In: *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings. 2010, pp. 249–256.

[75] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. "Generative adversarial networks". In: *Communications of the ACM* 63.11 (2020), pp. 139–144.

[76] Omer Gottesman, Yao Liu, Scott Sussex, Emma Brunskill, and Finale Doshi-Velez. "Combining parametric and nonparametric models for off-policy evaluation". In: *International Conference on Machine Learning*. PMLR. 2019, pp. 2366–2375.

[77] Martin D Gould, Mason A Porter, Stacy Williams, Mark McDonald, Daniel J Fenn, and Sam D Howison. "Limit order books". In: *Quantitative Finance* 13.11 (2013), pp. 1709–1742.

[78] Richard Gran. "On the convergence of random search algorithms in continuous time with applications to adaptive control". In: *IEEE Transactions on Systems, Man, and Cybernetics* 1 (1973), pp. 62–66.

[79] Benjamin Gravell, Peyman Mohajerin Esfahani, and Tyler Summers. "Learning optimal controllers for linear systems with multiplicative noise via policy gradient". In: *IEEE Transactions on Automatic Control* 66.11 (2020), pp. 5283–5298.

[80] Richard C Grinold and Ronald N Kahn. *Active portfolio management*. New York: McGraw Hill, 2000.

[81] Olivier Guéant, Charles-Albert Lehalle, and Joaquin Fernandez-Tapia. "Dealing with the inventory risk: a solution to the market making problem". In: *Mathematics and financial economics* 7 (2013), pp. 477–507.

[82] Olivier Guéant, Charles-Albert Lehalle, and Joaquin Fernandez-Tapia. "Optimal portfolio liquidation with limit orders". In: *SIAM Journal on Financial Mathematics* 3.1 (2012), pp. 740–764.

[83] Fabien Guilbaud and Huyên Pham. "Optimal high-frequency trading in a pro rata microstructure with predictive information". In: *Mathematical Finance* 25.3 (2015), pp. 545–575.

[84] Fabien Guilbaud and Huyen Pham. "Optimal high-frequency trading with limit and market orders". In: *Quantitative Finance* 13.1 (2013), pp. 79–94.

[85] Xin Guo, Tze Leung Lai, Howard Shek, and Samuel Po-Shing Wong. *Quantitative trading: algorithms, analytics, data, models, optimization*. Chapman and Hall/CRC, 2017.

[86] Varun Gupta and Ana Radovanovic. "Online stochastic bin packing". In: *arXiv preprint arXiv:1211.2687* (2012).

[87] László Györfi, Michael Kohler, Adam Krzyzak, Harro Walk, et al. *A distribution-free theory of nonparametric regression*. Vol. 1. Springer, 2002.

[88] Ben Hambly, Renyuan Xu, and Huining Yang. "Policy gradient methods for the noisy linear quadratic regulator over a finite horizon". In: *SIAM Journal on Control and Optimization* 59.5 (2021), pp. 3359–3391.

[89] Ben Hambly, Renyuan Xu, and Huining Yang. "Recent advances in reinforcement learning in finance". In: *Mathematical Finance* 33.3 (2023), pp. 437–503.

[90] Ben M Hambly, Renyuan Xu, and Huining Yang. "Policy Gradient Methods for the Noisy Linear Quadratic Regulator over a Finite Horizon". In: *Available at SSRN* (2020).

[91] Andi Han, Bamdev Mishra, Pratik Kumar Jawanpuria, and Junbin Gao. "On Riemannian optimization over positive definite matrices with the Bures-Wasserstein geometry". In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 8940–8953.

[92] Jiequn Han and Weinan E. "Deep learning approximation for stochastic control problems". In: *arXiv preprint arXiv:1611.07422* (2016).

[93] Jiequn Han, Arnulf Jentzen, and Weinan E. "Solving high-dimensional partial differential equations using deep learning". In: *Proceedings of the National Academy of Sciences* 115.34 (2018), pp. 8505–8510.

[94] Botao Hao, Xiang Ji, Yaqi Duan, Hao Lu, Csaba Szepesvári, and Mengdi Wang. "Bootstrapping fitted q-evaluation for off-policy inference". In: *International Conference on Machine Learning*. PMLR. 2021, pp. 4074–4084.

[95] Moritz Hardt and Benjamin Recht. "Patterns, predictions, and actions: A story about machine learning". In: *arXiv preprint arXiv:2102.05242* (2021).

[96] Trevor Hastie, Robert Tibshirani, Jerome H Friedman, and Jerome H Friedman. *The elements of statistical learning: data mining, inference, and prediction*. Vol. 2. Springer, 2009.

[97] Nikolaus Hautsch and Ruihong Huang. "The market impact of a limit order". In: *Journal of Economic Dynamics and Control* 36.4 (2012), pp. 501–522.

[98] Erik Heitfield and Yang-Ho Park. *Inferring term rates from SOFR futures prices*. Tech. rep. 2019.

[99] Dieter Hendricks and Diane Wilcox. "A reinforcement learning extension to the Almgren-Chriss framework for optimal trade execution". In: *2014 IEEE Conference on computational intelligence for financial engineering & economics (CIFEr)*. IEEE. 2014, pp. 457–464.

[100] Jonathan Ho, Ajay Jain, and Pieter Abbeel. "Denoising diffusion probabilistic models". In: *Advances in neural information processing systems* 33 (2020), pp. 6840–6851.

[101] Ulrich Horst and Michael Paulsen. "A law of large numbers for limit order books". In: *Mathematics of Operations Research* 42.4 (2017), pp. 1280–1312.

[102] Ruihong Huang and Tomas Polak. "Lobster: Limit order book reconstruction system". In: *Available at SSRN 1977207* (2011).

[103] Weibing Huang, Charles-Albert Lehalle, and Mathieu Rosenbaum. "Simulating and analyzing order book data: The queue-reactive model". In: *Journal of the American Statistical Association* 110.509 (2015), pp. 107–122.

[104] Weibing Huang and Mathieu Rosenbaum. "Ergodicity and diffusivity of Markovian order book models: a general framework". In: *SIAM Journal on Financial Mathematics* 8.1 (2017), pp. 874–900.

[105] David Hudgins and Patrick M Crowley. "Stress-testing US macroeconomic policy: A computational approach using stochastic and robust designs in a wavelet-based optimal control framework". In: *Computational Economics* 53 (2019), pp. 1509–1546.

[106] Hanna Hultin, Henrik Hult, Alexandre Proutiere, Samuel Samama, and Ala Tarighati. "A generative model of a limit order book using recurrent neural networks". In: *Quantitative Finance* 23.6 (2023), pp. 931–958.

[107] Peter Conway Humphreys, Arthur Guez, Olivier Tieleman, Laurent Sifre, Theophane Weber, and Timothy P Lillicrap. "Large-Scale Retrieval for Reinforcement Learning". In: *Advances in Neural Information Processing Systems*. 2022.

[108] Ahmed Hussein, Mohamed Medhat Gaber, Eyad Elyan, and Chrisina Jayne. "Imitation learning: A survey of learning methods". In: *ACM Computing Surveys (CSUR)* 50.2 (2017), pp. 1–35.

[109] Konark Jain, Nick Firoozye, Jonathan Kochems, and Philip Treleaven. "Limit Order Book Simulations: A Review". In: *arXiv preprint arXiv:2402.17359* (2024).

[110] Eric Jang, Shixiang Gu, and Ben Poole. "Categorical Reparametrization with Gumble-Softmax". In: *International Conference on Learning Representations (ICLR 2017)*. 2017.

[111] Joseph Jerome, Leandro Sanchez-Betancourt, Rahul Savani, and Martin Herdegen. "Model-based gym environments for limit order book trading". In: *arXiv preprint arXiv:2209.07823* (2022).

[112] Yanwei Jia and Xun Yu Zhou. "Policy Evaluation and Temporal-Difference Learning in Continuous Time and Space: A Martingale Approach". In: *Available at SSRN 3905379* (2021).

[113] Yanwei Jia and Xun Yu Zhou. "q-Learning in continuous time". In: *arXiv preprint arXiv:2207.00713* (2022).

[114] Nan Jiang and Lihong Li. "Doubly robust off-policy value evaluation for reinforcement learning". In: *International Conference on Machine Learning*. PMLR. 2016, pp. 652–661.

[115] Zeyu Jin, Johann Michael Schmitt, and Zaiwen Wen. "On the analysis of model-free methods for the linear quadratic regulator". In: *arXiv preprint arXiv:2007.03861* (2020).

[116] Sham M Kakade. "A natural policy gradient". In: *Advances in neural information processing systems* 14 (2001).

[117] Nathan Kallus and Masatoshi Uehara. "Efficiently breaking the curse of horizon in off-policy evaluation with double reinforcement learning". In: *Operations Research* 70.6 (2022), pp. 3282–3302.

[118] David A Kendrick. "Stochastic control for economic models: past, present and the paths ahead". In: *Journal of Economic Dynamics and Control* 29.1-2 (2005), pp. 3–30.

[119] Bekzhan Kerimkulov, James-Michael Leahy, David Šiška, and Łukasz Szpruch. "Convergence of policy gradient for entropy regularized MDPs with neural network approximation in the mean-field regime". In: *arXiv preprint arXiv:2201.07296* (2022).

[120] Jack Kiefer and Jacob Wolfowitz. "Stochastic estimation of the maximum of a regression function". In: *The Annals of Mathematical Statistics* (1952), pp. 462–466.

[121] Sara Klein, Simon Weissmann, and Leif Döring. "Beyond Stationarity: Convergence Analysis of Stochastic Softmax Policy Gradient Methods". In: *International Conference on Learning Representations*. 2024.

[122] Vijay Konda and John Tsitsiklis. "Actor-critic algorithms". In: *Advances in neural information processing systems* 12 (1999).

[123] V Yu Korolev, AV Chertok, A Yu Korchagin, and Alexander I Zeifman. "Modeling high-frequency order flow imbalance by functional limit theorems for two-sided risk processes". In: *Applied Mathematics and Computation* 253 (2015), pp. 224–241.

[124] Robert Kowalski. *Firms and Exchanges to Swap Views on Markets' Future*. `https://web.archive.org/web/20070930183528/http://www.thestreet.com/stocks/brokerages/945721.html`. Published: 2000-05-24 Archived: 2007-09-30 Accessed: 2024-12-04.

[125] N Krylov. "Approximating value functions for controlled degenerate diffusion processes by using piece-wise constant policies". In: *Electronic Journal of Probability* 4 (1999), pp. 1–19.

[126] Harold J Kushner. "Numerical methods for stochastic control problems in continuous time". In: *SIAM Journal on Control and Optimization* 28.5 (1990), pp. 999–1048.

[127] Harold J Kushner. *Probability Methods for Approximations in Stochastic Control and for Elliptic Equations*. Vol. 129. Mathematics in Science and Engineering. Elsevier Science, 1977. ISBN: 9780080956381.

[128] Harold J Kushner and G George Yin. *Stochastic approximation and recursive algorithms and applications*. Vol. 35. Springer Science & Business Media, 2003.

[129] John W Labuszewski, John E Nyhoff, Paul E Peterson, et al. *The CME Group Risk Management Handbook: Products and Applications*. John Wiley & Sons, 2010.

[130] Upmanu Lall and Ashish Sharma. "A nearest neighbor bootstrap for resampling hydrologic time series". In: *Water Resources Research* 32.3 (1996), pp. 679–693.

[131] Hoang Le, Cameron Voloshin, and Yisong Yue. "Batch policy learning under constraints". In: *International Conference on Machine Learning*. PMLR. 2019, pp. 3703–3712.

[132] Laura Leal, Mathieu Laurière, and Charles-Albert Lehalle. "Learning a functional control for high-frequency finance". In: *arXiv preprint arXiv:2006.09611* (2020).

[133] Charles-Albert Lehalle and Eyal Neuman. "Incorporating signals into optimal trading". In: *Finance and Stochastics* 23 (2019), pp. 275–311.

[134] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. "End-to-end training of deep visuomotor policies". In: *The Journal of Machine Learning Research* 17.1 (2016), pp. 1334–1373.

[135] Junyi Li, Xintong Wang, Yaoyang Lin, Arunesh Sinha, and Michael Wellman. "Generating realistic stock market order streams". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 34. 01. 2020, pp. 727–734.

[136] Mengyuan Li, Arman Kazemi, Ann Franchesca Laguna, and X Sharon Hu. "Associative Memory Based Experience Replay for Deep Reinforcement Learning". In: *Proceedings of the 41st IEEE/ACM International Conference on Computer-Aided Design*. 2022, pp. 1–9.

[137] Wen Li, Ying Zhang, Yifang Sun, Wei Wang, Mingjie Li, Wenjie Zhang, and Xuemin Lin. "Approximate nearest neighbor search on high dimensional dataexperiments, analyses, and improvement". In: *IEEE Transactions on Knowledge and Data Engineering* 32.8 (2019), pp. 1475–1488.

[138] Ye-Sheen Lim and Denise Gorse. "Intra-Day Price Simulation with Generative Adversarial Modelling of the Order Flow". In: *2021 20th IEEE International Conference on Machine Learning and Applications (ICMLA)*. IEEE. 2021, pp. 397–402.

[139] Dong C Liu and Jorge Nocedal. "On the limited memory BFGS method for large scale optimization". In: *Mathematical programming* 45.1 (1989), pp. 503–528.

[140] Yao Liu, Omer Gottesman, Aniruddh Raghu, Matthieu Komorowski, Aldo A Faisal, Finale Doshi-Velez, and Emma Brunskill. "Representation balancing MDPs for off-policy policy evaluation". In: *Advances in Neural Information Processing Systems* 31 (2018).

[141] Xiaofei Lu and Frédéric Abergel. "Order-book modeling and market making strategies". In: *Market Microstructure and Liquidity* 4.01n02 (2018), p. 1950003.

[142] A Rupam Mahmood, Hado P Van Hasselt, and Richard S Sutton. "Weighted importance sampling for off-policy learning with linear function approximation". In: *Advances in Neural Information Processing Systems* 27 (2014).

[143] Ashique Rupam Mahmood, Huizhen Yu, and Richard S Sutton. "Multi-step off-policy learning without importance sampling ratios". In: *arXiv preprint arXiv:1702.03006* (2017).

[144] Horia Mania, Aurelia Guy, and Benjamin Recht. "Simple random search of static linear policies is competitive for reinforcement learning". In: *Proceedings of the 32nd International Conference on Neural Information Processing Systems*. 2018, pp. 1805–1814.

[145] Shie Mannor, Duncan Simester, Peng Sun, and John N Tsitsiklis. "Bias and variance approximation in value function estimates". In: *Management Science* 53.2 (2007), pp. 308–322.

[146] Elman Mansimov and Kyunghyun Cho. *Simple Nearest Neighbor Policy Method for Continuous Control Tasks*. 2018. URL: `https://openreview.net/forum?id=ByL48G-AW`.

[147] Antonio Martinez-Sykora, MC So, CSM Currie, Christopher Bayliss, and JA Bennell. "Optimizing pricing and packing of variable-sized cargo". In: *IMA Journal of Management Mathematics* 32.3 (2021), pp. 303–327.

[148] Jincheng Mei, Chenjun Xiao, Csaba Szepesvari, and Dale Schuurmans. "On the global convergence rates of softmax policy gradient methods". In: *International Conference on Machine Learning*. PMLR. 2020, pp. 6820–6829.

[149] Andrew William Moore. *Efficient memory-based learning for robot control*. Tech. rep. University of Cambridge, Computer Laboratory, 1990.

[150] Maxime Morariu-Patrichi and Mikko S Pakkanen. "State-dependent Hawkes processes and their application to limit order book modelling". In: *Quantitative Finance* 22.3 (2022), pp. 563–583.

[151] Esteban Moro, Javier Vicente, Luis G Moyano, Austin Gerig, J Doyne Farmer, Gabriella Vaglica, Fabrizio Lillo, and Rosario N Mantegna. "Market impact and trading profile of hidden orders in stock markets". In: *Physical Review E — Statistical, Nonlinear, and Soft Matter Physics* 80.6 (2009), p. 066102.

[152] Wenlong Mou, Peng Ding, Martin J Wainwright, and Peter L Bartlett. "Kernel-based off-policy estimation without overlap: Instance optimality beyond semiparametric efficiency". In: *arXiv preprint arXiv:2301.06240* (2023).

[153] Chaoxu Mu, Yong Zhang, Hongjie Jia, and Haibo He. "Energy-storage-based intelligent frequency control of microgrid with stochastic model uncertainties". In: *IEEE Transactions on Smart Grid* 11.2 (2019), pp. 1748–1758.

[154] Rémi Munos. "Policy gradient in continuous time". In: *Journal of Machine Learning Research* 7.May (2006), pp. 771–791.

[155] Rémi Munos and Paul Bourgine. "Reinforcement learning for continuous stochastic control problems". In: *NIPS*. 1997, pp. 1029–1035.

[156] Ofir Nachum, Yinlam Chow, Bo Dai, and Lihong Li. "Dualdice: Behavior-agnostic estimation of discounted stationary distribution corrections". In: *Advances in Neural Information Processing Systems* 32 (2019).

[157] Peer Nagy, Sascha Frey, Silvia Sapora, Kang Li, Anisoara Calinescu, Stefan Zohren, and Jakob Foerster. "Generative AI for End-to-End Limit Order Book Modelling: A Token-Level Autoregressive Generative Model of Message Flow Using a Deep State Space Network". In: *Proceedings of the Fourth ACM International Conference on AI in Finance*. 2023, pp. 91–99.

[158] Yurii Nesterov and Vladimir Spokoiny. "Random gradient-free minimization of convex functions". In: *Foundations of Computational Mathematics* 17.2 (2017), pp. 527–566.

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

[160] Victor Niederhoffer and Matthew Fontaine Maury Osborne. "Market making and reversal on the stock exchange". In: *Journal of the American Statistical Association* 61.316 (1966), pp. 897–916.

[161] Brian Ning, Franco Ho Ting Lin, and Sebastian Jaimungal. "Double deep q-learning for optimal execution". In: *Applied Mathematical Finance* 28.4 (2021), pp. 361–380.

[162] Jorge Nocedal and Stephen J Wright. "Line search methods". In: *Numerical optimization* (2006), pp. 30–65.

[163] Stephen M Omohundro. *Five balltree construction algorithms*. International Computer Science Institute Berkeley, 1989.

[164] Mark Paddrik, Roy Hayes, Andrew Todd, Steve Yang, Peter Beling, and William Scherer. "An agent based model of the E-Mini S&P 500 applied to Flash Crash analysis". In: *2012 IEEE Conference on Computational Intelligence for Financial Engineering & Economics (CIFEr)*. IEEE. 2012, pp. 1–8.

[165] Stephan Pareigis. "Multi-grid methods for reinforcement learning in controlled diffusion processes". In: *NIPS*. Citeseer. 1996, pp. 1033–1039.

[166] Jyothish Pari, Nur Muhammad Shafiullah, Sridhar Pandian Arunachalam, and Lerrel Pinto. "The surprising effectiveness of representation learning for visual imitation". In: *Robotics: Science and Systems 2022* (2022).

[167] Seohong Park, Jaekyeom Kim, and Gunhee Kim. "Time Discretization-Invariant Safe Action Repetition for Policy Gradient Methods". In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 267–279.

[168] Youngsuk Park, Ryan Rossi, Zheng Wen, Gang Wu, and Handong Zhao. "Structured Policy Iteration for Linear Quadratic Regulator". In: *International Conference on Machine Learning*. PMLR. 2020, pp. 7521–7531.

[169] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. "Automatic differentiation in pytorch". In: (2017).

[170] Judea Pearl. *Causality*. Cambridge university press, 2009.

[171] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[172] Boris Teodorovich Polyak. "Gradient methods for minimizing functionals". In: *Zhurnal vychislitel'noi matematiki i matematicheskoi fiziki* 3.4 (1963), pp. 643–653.

[173] Doina Precup. "Eligibility traces for off-policy policy evaluation". In: *Computer Science Department Faculty Publication Series* (2000), p. 80.

[174] Doina Precup, Richard S Sutton, and Sanjoy Dasgupta. "Off-policy temporal-difference learning with function approximation". In: *ICML*. 2001, pp. 417–424.

[175] Balaji Rajagopalan and Upmanu Lall. "A k-nearest-neighbor simulator for daily precipitation and other weather variables". In: *Water Resources Research* 35.10 (1999), pp. 3089–3101.

[176] Marcello Rambaldi, Emmanuel Bacry, and Fabrizio Lillo. "The role of volume in order book dynamics: a multivariate Hawkes process analysis". In: *Quantitative Finance* 17.7 (2017), pp. 999–1020.

[177] William J Raseman, Balaji Rajagopalan, Joseph R Kasprzyk, and William Kleiber. "Nearest neighbor time series bootstrap for generating influent water quality scenarios". In: *Stochastic Environmental Research and Risk Assessment* 34.1 (2020), pp. 23–31.

[178] Leonard Andreevich Rastrigin. "About convergence of random search method in extremal control of multi-parameter systems". In: *Avtomat. i Telemekh* 24.11 (1963), pp. 1467–1473.

[179] Benjamin Recht. "A tour of reinforcement learning: The view from continuous control". In: *Annual Review of Control, Robotics, and Autonomous Systems* 2 (2019), pp. 253–279.

[180] Christoph Reisinger, Wolfgang Stockinger, and Yufei Zhang. "A fast iterative PDE-based algorithm for feedback controls of nonsmooth mean-field control problems". In: *arXiv preprint arXiv:2108.06740* (2021).

[181] Herbert Robbins and Sutton Monro. "A stochastic approximation method". In: *The Annals of Mathematical Statistics* (1951), pp. 400–407.

[182] Johannes Ruf and Weiguan Wang. "Neural networks for option pricing and hedging: a literature review". In: *arXiv preprint arXiv:1911.05620* (2019).

[183] Emilio Said, Ahmed Bel Hadj Ayed, Alexandre Husson, and Frédéric Abergel. "Market impact: A systematic study of limit orders". In: *Market Microstructure and Liquidity* 3.03n04 (2017), p. 1850008.

[184] Tim Salimans, Jonathan Ho, Xi Chen, Szymon Sidor, and Ilya Sutskever. "Evolution strategies as a scalable alternative to reinforcement learning". In: *arXiv preprint arXiv:1703.03864* (2017).

[185] Matthew Schlegel, Wesley Chung, Daniel Graves, Jian Qian, and Martha White. "Importance resampling for off-policy prediction". In: *Advances in Neural Information Processing Systems* 32 (2019).

[186] Hanspeter Schmidli. *Stochastic Control in Insurance*. Springer Science & Business Media, 2007.

[187] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. "Trust region policy optimization". In: *International conference on machine learning*. PMLR. 2015, pp. 1889–1897.

[188] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. "Proximal policy optimization algorithms". In: *arXiv preprint arXiv:1707.06347* (2017).

[189] Hans-Paul Schwefel. "Evolutionsstrategien für die numerische Optimierung". In: *Numerische Optimierung von Computer-Modellen mittels der Evolutionsstrategie*. Springer, 1977, pp. 123–176.

[190] *SEC: Firms Will Pay More Than $240 Million in Penalties and Disgorgement*. `https://www.sec.gov/news/press/2004-42.htm`. Accessed: 2024-12-04.

[191] Thibault Séjourné, François-Xavier Vialard, and Gabriel Peyré. "The unbalanced Gromov Wasserstein distance: Conic formulation and relaxation". In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 8766–8779.

[192] Zijian Shi and John Cartlidge. "Neural Stochastic Agent-Based Limit Order Book Simulation: A Hybrid Methodology". In: *Proceedings of the 2023 International Conference on Autonomous Agents and Multiagent Systems*. 2023, pp. 2481–2483.

[193] Zijian Shi and John Cartlidge. "The Limit Order Book Recreation Model (LOBRM): An Extended Analysis". In: *Machine Learning and Knowledge Discovery in Databases. Applied Data Science Track: European Conference, ECML PKDD 2021, Bilbao, Spain, September 13–17, 2021, Proceedings, Part IV 21*. Springer. 2021, pp. 204–220.

[194] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. "Mastering the game of Go with deep neural networks and tree search". In: *nature* 529.7587 (2016), pp. 484–489.

[195] Justin Sirignano and Rama Cont. "Universal features of price formation in financial markets: perspectives from deep learning". In: *Quantitative Finance* 19.9 (2019), pp. 1449–1459.

[196] David Šiška and Łukasz Szpruch. "Gradient flows for regularized stochastic control problems". In: *arXiv preprint arXiv:2006.05956* (2020).

[197] James C Spall and John A Cristion. "Nonlinear adaptive control using neural networks: Estimation with a smoothed form of simultaneous perturbation gradient approximation". In: *Proceedings of 1994 American Control Conference-ACC'94*. Vol. 3. IEEE. 1994, pp. 2560–2564.

[198] James C Spall et al. "Multivariate stochastic approximation using a simultaneous perturbation gradient approximation". In: *IEEE transactions on automatic control* 37.3 (1992), pp. 332–341.

[199] Thomas Spooner, John Fearnley, Rahul Savani, and Andreas Koukorinis. "Market making via reinforcement learning". In: *AAMAS18*. International foundation for autonomous agents and multiagent systems. 2018, pp. 434–442.

[200] Jingrui Sun, Xun Li, and Jiongmin Yong. "Open-loop and closed-loop solvabilities for stochastic linear quadratic optimal control problems". In: *SIAM Journal on Control and Optimization* 54.5 (2016), pp. 2274–2308.

[201] Jingrui Sun and Jiongmin Yong. *Stochastic linear-quadratic optimal control theory: Open-loop and closed-loop solutions*. Springer Nature, 2020.

[202] Wenyu Sun and Ya-Xiang Yuan. *Optimization theory and methods: nonlinear programming*. Vol. 1. Springer Science & Business Media, 2006.

[203] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[204] Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. "Policy gradient methods for reinforcement learning with function approximation". In: *Advances in neural information processing systems* 12 (1999).

[205] Lukasz Szpruch, Tanut Treetanthiploet, and Yufei Zhang. "Optimal scheduling of entropy regulariser for continuous-time linear-quadratic reinforcement learning". In: *arXiv preprint arXiv:2208.04466* (2022).

[206] Corentin Tallec, Léonard Blier, and Yann Ollivier. "Making Deep Q-learning methods robust to time discretization". In: *arXiv preprint arXiv:1901.09732* (2019).

[207] Philip Thomas and Emma Brunskill. "Data-efficient off-policy policy evaluation for reinforcement learning". In: *International Conference on Machine Learning*. PMLR. 2016, pp. 2139–2148.

[208] Bence Tóth, Yves Lemperiere, Cyril Deremble, Joachim De Lataillade, Julien Kockelkoren, and J-P Bouchaud. "Anomalous price impact and the critical nature of liquidity in financial markets". In: *Physical Review X* 1.2 (2011), p. 021006.

[209] Erin Towler, Balaji Rajagopalan, Chad Seidel, and R Scott Summers. "Simulating ensembles of source water quality using a K-nearest neighbor resampling approach". In: *Environmental Science & Technology* 43.5 (2009), pp. 1407–1411.

[210] Masatoshi Uehara, Chengchun Shi, and Nathan Kallus. "A Review of Off-Policy Evaluation in Reinforcement Learning". In: *arXiv preprint arXiv:2212.06355* (2022).

[211] Masatoshi Uehara and Wen Sun. "Pessimistic model-based offline reinforcement learning under partial coverage". In: *arXiv preprint arXiv:2107.06226* (2021).

[212] Vladimir Vapnik. *The nature of statistical learning theory*. Springer Science & Business Media, 2013.

[213] Raja Velu, Maxence Hardy, and Daniel Nehren. *Algorithmic trading and quantitative strategies*. Chapman and Hall/CRC, 2020.

[214] Cameron Voloshin, Hoang Minh Le, Nan Jiang, and Yisong Yue. "Empirical Study of Off-Policy Policy Evaluation for Reinforcement Learning". In: *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*. 2021.

[215] Haoran Wang, Thaleia Zariphopoulou, and Xun Yu Zhou. "Reinforcement Learning in Continuous Time and Space: A Stochastic Control Approach." In: *J. Mach. Learn. Res.* 21 (2020), pp. 198–1.

[216] Haoran Wang and Xun Yu Zhou. "Continuous-time mean-variance portfolio selection: A reinforcement learning framework". In: *Mathematical Finance* 30.4 (2020), pp. 1273–1308.

[217] Weichen Wang, Jiequn Han, Zhuoran Yang, and Zhaoran Wang. "Global convergence of policy gradient for linear-quadratic mean-field control/game in continuous time". In: *International Conference on Machine Learning*. PMLR. 2021, pp. 10772–10782.

[218] Weiwei Wang, Yuqiang Li, and Xianyi Wu. "Off-policy evaluation for tabular reinforcement learning with synthetic trajectories". In: *Statistics and Computing* 34.1 (2024), pp. 1–17.

[219] Kevin Webster and Nicholas Westray. "Getting More for Less-Better A/B Testing via Causal Regularization". In: *B Testing via Causal Regularization (July 12, 2022)* (2022).

[220] Kevin T Webster. *Handbook of Price Impact Modeling*. CRC Press, 2023.

[221] Grady Williams, Paul Drews, Brian Goldfain, James M Rehg, and Evangelos A Theodorou. "Information-theoretic model predictive control: Theory and applications to autonomous driving". In: *IEEE Transactions on Robotics* 34.6 (2018), pp. 1603–1622.

[222] Zekun Ye, Weijie Deng, Shuigeng Zhou, Yi Xu, and Jihong Guan. "Optimal trade execution based on deep deterministic policy gradient". In: *Database Systems for Advanced Applications: 25th International Conference, DASFAA 2020, Jeju, South Korea, September 24–27, 2020, Proceedings, Part I 25*. Springer. 2020, pp. 638–654.

[223] Ming Yin and Yu-Xiang Wang. "Asymptotically efficient off-policy evaluation for tabular reinforcement learning". In: *International Conference on Artificial Intelligence and Statistics*. PMLR. 2020, pp. 3948–3958.

[224] Jiongmin Yong and Xun Yu Zhou. *Stochastic controls: Hamiltonian systems and HJB equations*. Vol. 43. Springer Science & Business Media, 1999.

[225] Kaiqing Zhang, Bin Hu, and Tamer Basar. "Policy Optimization for $\mathcal{H}_2$ Linear Control with $\mathcal{H}_\infty$ Robustness Guarantee: Implicit Regularization and Global Convergence". In: *SIAM Journal on Control and Optimization* 59.6 (2021), pp. 4081–4109.

[226] Kaiqing Zhang, Xiangyuan Zhang, Bin Hu, and Tamer Basar. "Derivative-free policy optimization for linear risk-sensitive and robust control design: Implicit regularization and sample complexity". In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 2949–2964.

[227] Zihao Zhang, Stefan Zohren, and Stephen Roberts. "Deep Reinforcement Learning for Trading". In: *The Journal of Financial Data Science* 2.2 (2020), pp. 25–40.

[228] Xun Yu Zhou and Duan Li. "Continuous-time mean-variance portfolio selection: A stochastic LQ framework". In: *Applied Mathematics and Optimization* 42.1 (2000), pp. 19–33.

Appendix A

Appendix: K-Nearest-Neighbor Resampling for Off-Policy Evaluation in Stochastic Control

A.1 Theoretical considerations

A.1.1 Environment

In this section, we discuss the relationship of our setting to general MDPs and the additional restrictions we impose on the dynamical system.

Comparison: MDP vs Controlled Iterated Function System (IFS) The standard MDP formulation for an episodic setting of fixed length can be related to our random dynamical systems formulation in the following sense: Consider an MDP $(\mathcal{X}, \mathcal{U}, \mathbb{P}, R)$ with a fixed episode length T where $\mathcal{X}$ is the state space and $\mathcal{U}$ the action space. $P(\cdot \mid x_t, a_t)$ denotes the transition kernel, $R(x_t, a_t, x_{t+1})$ the reward distribution, d_0 the distribution of the initial state and some Markovian policy $\pi(\cdot \mid x_t)$. The value of the policy is then given by $V(\pi) = \mathbb{E}_{x \sim d_0}\left[\sum_{t=0}^{T-1} r_t \mid x_0 = x\right]$ where $a_t \sim \pi_e(\cdot \mid x_t), x_{t+1} \sim P(\cdot \mid x_t, a_t), r_t \sim R(x_t, a_t, x_{t+1})$.

Further, recall that an MDP with a Markovian policy reduces to a time-homogenous Markov process on $S = \mathcal{X} \times \mathcal{U}$ where S is equipped with a σ-algebra $\mathcal{S}$ and the transition kernels are denoted by $(x_{t+1}, a_{t+1}) \sim \tilde{P}(\cdot \mid x_t, a_t)$. It is a well known fact that every time-homogenous Markov process can be generated via a random dynamical system with iid random functions that are measurable with respect to $(S, \mathcal{S})$ i.e. [6, 27]. The representation of a Markov process with random functions however is not unique. Furthermore, how restrictions on the distribution of the random functions (i.e. only continuous functions or sets with equicontinuous functions [1]) impact the scope of representable Markov processes in general is to our knowledge an open research question and is beyond the scope of our work. For the one dimensional case ([27] p.228), one can explicitly construct an iterated function system from the quantile function of the transition kernel. With the same method one can construct an iterated function system where the functions are continuous or from a set of equicontinuous functions from the quantile function of the transition kernel where the quantile function satisfies the same property. While it is unclear how the representative power of an iterative function system varies between distributions on $C(S, S)$ and equicontinuous subsets $\Gamma \subset C(S, S)$, for the case that S is compact we know that by Arzela-Ascoli's theorem

[1] Consider a set of functions Γ between two metric spaces (X, d_1) and (Y, d_2). Γ consists of uniformly equicontinuous functions if for every $\varepsilon > 0$, there exists a $\delta > 0$ such that $d_2(f(x_1), f(x_2)) < \varepsilon$ for all $f \in \Gamma$ and all $x_1, x_2 \in X$ where $d_1(x_1, x_2) < \delta$.

(e.g. see [4]) every compact set contained in $C(S,S)$ is uniformly equicontinuous. Finally, note that the setting we propose is more general compared to the settings considered in [62, 63, 76].

Assumptions on the dynamical system In general, we make three additional assumptions on the dynamical system:

Assumption 5. *1. $\Gamma \subset C(S,S)$ is a set of uniformly equicontinuous functions*

2. ϵ_t is supported on $[-M,M]$ for $M \in \mathbb{R}$ and it holds that $r : S \times \mathcal{X} \to [-M,M]$

3. $(\epsilon_t)_{t=0}^T$ and $(\alpha_t)_{t=1}^T$ are assumed to be independent and $\mathbb{E}[\epsilon_t] = 0$

The first assumption becomes necessary due to taking limits over compositions of transition functions. The impact of this assumption on the generality of the results is discussed in the previous paragraph and we note that sets of uniformly Lipschitz continuous functions (as in [62, 63, 76]) are in particular uniformly equicontinuous. The second assumption simplifies the proof considerably and the restriction on regression targets on a compact set is a common simplification in non-parametric statistics. It is possible that this assumption is not necessary as the nearest neighbor regression satisfies a strong concentration property (cf. [87] Lemma 6.3). Possibly this could be extended to our setting. The third assumption is necessary to disentangle the noise of the transition from averaging over the rewards. Note, however, with the generalization mentioned in Remark 1 of the main part one can introduce rewards depending on the transitions which are more general than the commonly assumed reward structures in a dynamical system formulation (i.e. [21]). Finally, the condition $\mathbb{E}[\epsilon_t] = 0$ can always be satisfied by shifting the reward function by the expectation of the noise.

A.1.2 Proof of Propositions 7

Proposition 7. *Given the setting described in Section 3.2 of the main part and under Assumption 2, it holds that D_n, E_n, F_n, and G_n from the proof of Theorem 2 vanish as $n \to \infty$.*

Proof. For

$$\begin{aligned}
E_n =& \mathbb{E}\left[\left(\sum_{i^\tau,t^\tau}^{n,T} \sum_{\substack{(j^{\tau-1},i_\tau),\\(s^{\tau-1},t_\tau)}}^{n,T} W_{n,i^\tau,t^\tau} W_{n,(j^{\tau-1},i_\tau),(s^{\tau-1},t_\tau)} (R_{i_\tau,t_\tau} - r(X_{i_\tau,t_\tau}, U_{i_\tau,t_\tau}))^2\right)\right] \\
&+ \mathbb{E}\left[\left(\sum_{i^\tau,t^\tau}^{n,T} \sum_{\substack{j^\tau,s^\tau \\ j_\tau \neq i_\tau \vee s_\tau \neq t_\tau}}^{N,T} W_{n,i^\tau,t^\tau} W_{n,j^\tau,s^\tau} (R_{i_\tau,t_\tau} - r(X_{i_\tau,t_\tau}, U_{i_\tau,t_\tau})) \right.\right. \\
&\qquad \left.\left. \cdot (R_{j_\tau,s_\tau} - r(X_{j_\tau,s_\tau}, U_{j_\tau,s_\tau}))\right)\right] \\
:=& E'_n + E''_n
\end{aligned}$$

note first that by Assumption 2.1,

$$|E'_n| \leq 4M^2 \mathbb{E}\left[\sum_{i^\tau,t^\tau}^{n,T} \sum_{\substack{(j^{\tau-1},i_\tau),\\(s^{\tau-1},t_\tau)}}^{n,T} |W_{n,i^\tau,t^\tau} W_{n,(j^{\tau-1},i_\tau),(s^{\tau-1},t_\tau)}|\right] \to 0$$

and

$$E''_n = 0$$

since for $i_\tau \neq j_\tau$ or $s_\tau \neq t_\tau$

$$\begin{aligned}
&\mathbb{E}\Big[W_{n,i^\tau,t^\tau} W_{n,j^\tau,s^\tau}(R_{i_\tau,t_\tau} - r(X_{i_\tau,t_\tau}, U_{i_\tau,t_\tau}))(R_{j_\tau,s_\tau} - r(X_{j_\tau,s_\tau}, U_{j_\tau,s_\tau}))\Big] \\
\leq &\mathbb{E}\Big[\mathbb{E}\Big[W_{n,i^\tau,t^\tau} W_{n,j^\tau,s^\tau} \epsilon_{i_\tau,t_\tau} \epsilon_{j_\tau,s_\tau} \Big| D\Big]\Big] \\
\leq &\mathbb{E}\Big[W_{n,i^\tau,t^\tau} W_{n,j^\tau,s^\tau} \mathbb{E}[\epsilon_{i_\tau,t_\tau}] \mathbb{E}[\epsilon_{j_\tau,t_\tau}]\Big] \\
= &0
\end{aligned}$$

by the definition of ϵ_{i_τ,t_τ} and ϵ_{j_τ,s_τ}.

Next, F_n can be represented by a telescoping sum, such that

$$\begin{aligned}
F_n \leq\ & (\tau+2)\mathbb{E}\Bigg[\Bigg(\sum_{i^\tau,t^\tau}^{n,T} W_{n,i^\tau,t^\tau}\big(r(X_{i_\tau,t_\tau}, U_{i_\tau,t_\tau}) \\
& - r(\alpha_{i_{\tau-1},t_{\tau-1}}|_u(X_{i_{\tau-1},t_{\tau-1}}, U_{i_{\tau-1},t_{\tau-1}}))\big)\Bigg)^2\Bigg] \\
& + (\tau+2)\mathbb{E}\Bigg[\sum_{k=1}^{\tau-1} \\
& \Bigg(\sum_{i^\tau,t^\tau}^{n,T} W_{n,i^\tau,t^\tau}\big(r(\alpha_{i_{\tau-1},t_{\tau-1}}|_u \circ \cdots \circ \alpha_{i_{\tau-k},t_{\tau-k}}|_u(X_{i_{\tau-k},t_{\tau-k}}, U_{i_{\tau-k},t_{\tau-k}})) \\
& - r(\alpha_{i_{\tau-1},t_{\tau-1}}|_u \circ \cdots \circ \alpha_{i_{\tau-k-1},t_{\tau-k-1}}|_u(X_{i_{\tau-k-1},t_{\tau-k-1}}, U_{i_{\tau-k-1},t_{\tau-k-1}}))\big)\Bigg)^2\Bigg] \\
& + (\tau+2)\mathbb{E}\Bigg[\Bigg(\sum_{i^\tau,t^\tau}^{n,T} W_{n,i^\tau,t^\tau}\big(r(\alpha_{i_{\tau-1},t_{\tau-1}}|_u \circ \cdots \circ \alpha_{i_0,t_0}|_u(X_{i_0,t_0}, U_{i_0,t_0})) \\
& - r(\alpha_{i_{\tau-1},t_{\tau-1}}|_u \circ \cdots \circ \alpha_{i_0,t_0}|_u(\tilde{X}_1, u_0(\tilde{X}_1)))\big)\Bigg)^2\Bigg] \\
:= &\ (\tau+2)\left(F_n^\tau + \sum_{k=1}^{\tau} F_n^{\tau-k} + F_n^{-1}\right).
\end{aligned}$$

By Cauchy–Schwarz inequality, and Assumption 2.2,

$$
\begin{aligned}
F_n^\tau \leq \mathbb{E}\Bigg[\Bigg(&\sum_{i^\tau,t^\tau}^{n,T}\sqrt{|W_{n,i^\tau,t^\tau}|}\sqrt{|W_{n,i^\tau,t^\tau}|}\\
&\left|r(X_{i_\tau,t_\tau},U_{i_\tau,t_\tau})-r(\alpha_{i_{\tau-1},t_{\tau-1}}|_u(X_{i_{\tau-1},t_{\tau-1}},U_{i_{\tau-1},t_{\tau-1}}))\right|\Bigg)^2\Bigg]\\
\leq \mathbb{E}\Bigg[&\left(\sum_{i^\tau,t^\tau}^{n,T}|W_{n,i^\tau,t^\tau}|\right)\\
&\cdot\left(\sum_{i^\tau,t^\tau}^{n,T}|W_{n,i^\tau,t^\tau}|\left(r(X_{i_\tau,t_\tau},U_{i_\tau,t_\tau})-r(X_{i_{\tau-1},t_{\tau-1}+1},u_\tau(X_{i_{\tau-1},t_{\tau-1}+1}))\right)^2\right)\Bigg]\\
\leq L\mathbb{E}&\left[\sum_{i^\tau,t^\tau}^{n,T}|W_{n,i^\tau,t^\tau}|\left(r(X_{i_\tau,t_\tau},U_{i_\tau,t_\tau})-r(X_{i_{\tau-1},t_{\tau-1}+1},u_\tau(X_{i_{\tau-1},t_{\tau-1}+1}))\right)^2\right]\\
=L&\tilde{F}_n^\tau
\end{aligned}
$$

and, for an arbitrary $\delta>0$,

$$
\begin{aligned}
\tilde{F}_n^\tau \leq&\mathbb{E}\Bigg[\sum_{i^\tau,t^\tau}^{n,T}|W_{n,i^\tau,t^\tau}|\left(r(X_{i_\tau,t_\tau},U_{i_\tau,t_\tau})-r(X_{i_{\tau-1},t_{\tau-1}+1},u_\tau(X_{i_{\tau-1},t_{\tau-1}+1}))\right)^2\\
&\cdot I_{d\left((X_{i_\tau,t_\tau},U_{i_\tau,t_\tau}),(X_{i_{\tau-1},t_{\tau-1}+1},u_\tau(X_{i_{\tau-1},t_{\tau-1}+1}))\right)\leq\delta}\Bigg]\\
+\mathbb{E}&\Bigg[\sum_{i^\tau,t^\tau}^{n,T}|W_{n,i^\tau,t^\tau}|\left(r(X_{i_\tau,t_\tau},U_{i_\tau,t_\tau})-r(X_{i_{\tau-1},t_{\tau-1}+1},u_\tau(X_{i_{\tau-1},t_{\tau-1}+1}))\right)^2\\
&\cdot I_{d\left((X_{i_\tau,t_\tau},U_{i_\tau,t_\tau}),(X_{i_{\tau-1},t_{\tau-1}+1},u_\tau(X_{i_{\tau-1},t_{\tau-1}+1}))\right)>\delta}\Bigg]\\
\leq&L\left(\sup_{x,y\in S:d(x,y)\leq\delta}|r(x)-r(y)|\right)^2\\
&+4M^2\mathbb{E}\left[\sum_{i^\tau,t^\tau}^{n,T}|W_{n,i^\tau,t^\tau}|I_{d\left((X_{i_\tau,t_\tau},U_{i_\tau,t_\tau}),(X_{i_{\tau-1},t_{\tau-1}+1},u_\tau(X_{i_{\tau-1},t_{\tau-1}+1}))\right)>\delta}\right]
\end{aligned}
$$

Hence, by Assumption 2.3,

$$
\limsup_{n\to\infty}F_n^\tau\leq L^2\left(\sup_{x,y\in S:d(x,y)\leq\delta}|r(x)-r(y)|\right)^2
$$

and since r uniformly continuous

$$
F_n^\tau\to 0
$$

as $\delta\to 0$. The argument for $F_n^{\tau-k}$ with $k=1,...,\tau+1$ is analogous additionally using the uniform equicontinuity of the random functions.

Consider

$$\begin{aligned}
G_n \leq & 3\mathbb{E}\Bigg[\Bigg(\sum_{i^\tau,t^\tau}^{n,T}\Big(W_{n,i^\tau,t^\tau} \\
& - W_{n,i^\tau,t^\tau}(D_{i^\tau,t^\tau})\Big)r(\alpha_{i_\tau,t_\tau}|_u \circ \cdots \circ \alpha_{i_0,t_0}|_u(\tilde{X}_1, u_0(\tilde{X}_1)))\Bigg)^2\Bigg] \\
& + 3\mathbb{E}\left[\left(\sum_{i^\tau,t^\tau}^{n,T} W_{n,i^\tau,t^\tau}(D_{i^\tau,t^\tau})\Delta_{i^\tau,t^\tau} r(\tilde{X}_1)\right)^2\right] \\
& + 3\mathbb{E}\left[\left(\left(\sum_{i^\tau,t^\tau}^{n,T} W_{n,i^\tau,t^\tau}(D_{i^\tau,t^\tau}) - 1\right)\mathbb{E}[R_\tau|_u|\tilde{X}_1]\right)^2\right] \\
=: & 3H_n + 3I_n + 3J_n
\end{aligned}$$

where

$$\Delta_{i^\tau,t^\tau} r(\tilde{X}_1) \coloneqq \Big(r(\alpha_{i_{\tau-1},t_{\tau-1}}|_u \circ \cdots \circ \alpha_{i_0,t_0}|_u(\tilde{X}_1, u_0(\tilde{X}_1))) - \mathbb{E}[R_\tau|_u|\tilde{X}_1]\Big).$$

Next, for

$$H_n \leq 4M^2\mathbb{E}\left[\left(\sum_{i^\tau,t^\tau}^{n,T} |W_{n,i^\tau,t^\tau} - W_{n,i^\tau,t^\tau}(D_{i^\tau,t^\tau})|\right)^2\right]$$

and, thus, by Assumption 2.5, $H_n \to 0$.

Further,

$$\begin{aligned}
I_n = & \mathbb{E}\Bigg[\sum_{i^\tau,t^\tau}^{n,T} \sum_{\substack{j^\tau,s^\tau \\ |(i^\tau,j^\tau)|<2\tau+2}}^{n,T} W_{n,i^\tau,t^\tau}(D_{i^\tau,t^\tau})W_{n,j^\tau,s^\tau}(D_{j^\tau,s^\tau}) \\
& \Delta_{i^\tau,t^\tau} r(\tilde{X}_1)\Delta_{j^\tau,s^\tau} r(\tilde{X}_1)\Bigg] \\
& + \mathbb{E}\Bigg[\sum_{i^\tau,t^\tau}^{n,T} \sum_{\substack{j^\tau,s^\tau \\ |(i^\tau,j^\tau)|=2\tau+2}}^{n,T} W_{n,i^\tau,t^\tau}(D_{i^\tau,t^\tau})W_{n,j^\tau,s^\tau}(D_{j^\tau,s^\tau}) \\
& \Delta_{i^\tau,t^\tau} r(\tilde{X}_1)\Delta_{j^\tau,s^\tau} r(\tilde{X}_1)\Bigg] \\
\coloneqq & I'_n + I''_n
\end{aligned}$$

where by Assumption 2.1

$$I'_n \leq 4M^2\mathbb{E}\left[\sum_{i^\tau,t^\tau}^{n,T} \sum_{\substack{j^\tau,s^\tau \\ |(i^\tau,j^\tau)|<2\tau+2}}^{n,T} |W_{n,i^\tau,t^\tau}(D_{i^\tau,t^\tau})W_{n,j^\tau,s^\tau}(D_{j^\tau,s^\tau})|\right] \to 0.$$

Note that

$$
\begin{aligned}
|I_n''| \leq 4M^2\mathbb{E}\Bigg[&\sum_{i^\tau,t^\tau}^{n,T}\sum_{\substack{j^\tau,s^\tau \\ |(i^\tau,j^\tau)|=2\tau+2}}^{n,T} |W_{n,i^\tau,t^\tau}(D_{i^\tau,t^\tau})| \\
&\cdot |W_{n,j^\tau,s^\tau}(D_{j^\tau,s^\tau}) - W_{n,j^\tau,s^\tau}(D_{(i^\tau,j^\tau),(t^\tau,s^\tau)})|\Bigg] \\
&+ \left|\mathbb{E}\left[\sum_{i^\tau,t^\tau}^{n,T}\sum_{\substack{j^\tau,s^\tau \\ |(i^\tau,j^\tau)|=2\tau+2}}^{n,T} W_{n,i^\tau,t^\tau}(D_{i^\tau,t^\tau})W_{n,j^\tau,s^\tau}(D_{(i^\tau,j^\tau),(t^\tau,s^\tau)})\right]\right| \\
&\coloneqq \hat{I}_n' + |\hat{I}_n''|
\end{aligned}
$$

where by Assumption 2.5 $\hat{I}_n' \to 0$. For $\hat{I}_n''$, define $\Theta_{(j^\tau,s^\tau)} \coloneqq (\tilde{X}_1, (\alpha_{j_k,s_k})_{k=0}^{\tau-1})$. Note that by definition $W_{n,i^\tau,t^\tau}(D_{i^\tau,t^\tau})W_{n,j^\tau,s^\tau}(D_{(i^\tau,j^\tau),(t^\tau,s^\tau)})$ and $\Delta_{i^\tau,t^\tau}r(\tilde{X}_1)$ are independent conditioned on $\Theta_{(j^\tau,s^\tau)}$ and that $\Delta_{i^\tau,t^\tau}r(\tilde{X}_1)$ is independent of $(\alpha_{j_k,s_k})_{k=0}^{\tau-1}$. This observation with the measurability of $\Delta_{j^\tau,s^\tau}r(\tilde{X}_1)$ with respect to the σ-algebra generated by $\Theta_{(j^\tau,s^\tau)}$ and the fact that $\mathbb{E}[\Delta_{i^\tau,t^\tau}r(\tilde{X}_1) \mid \tilde{X}_1] = 0$, yields for any summand where $|(i^\tau, j^\tau)| = \tau + 2$ that

$$
\begin{aligned}
&\mathbb{E}\Big[W_{n,i^\tau,t^\tau}(D_{i^\tau,t^\tau})W_{n,j^\tau,s^\tau}(D_{(i^\tau,j^\tau),(t^\tau,s^\tau)})\Delta_{i^\tau,t^\tau}r(\tilde{X}_1)\Delta_{j^\tau,s^\tau}r(\tilde{X}_1)\Big] \\
=&\mathbb{E}\Big[\mathbb{E}\Big[W_{n,i^\tau,t^\tau}(D_{i^\tau,t^\tau})W_{n,j^\tau,s^\tau}(D_{(i^\tau,j^\tau),(t^\tau,s^\tau)})\Delta_{i^\tau,t^\tau}r(\tilde{X}_1) \mid \Theta_{(j^\tau,s^\tau)}\Big]\Delta_{j^\tau,s^\tau}r(\tilde{X}_1)\Big] \\
=&\mathbb{E}\Big[\mathbb{E}\Big[W_{n,i^\tau,t^\tau}(D_{i^\tau,t^\tau})W_{n,j^\tau,s^\tau}(D_{(i^\tau,j^\tau),(t^\tau,s^\tau)}) \mid \Theta_{(j^\tau,s^\tau)}\Big] \\
&\quad\cdot \mathbb{E}\Big[\Delta_{i^\tau,t^\tau}r(\tilde{X}_1) \mid \tilde{X}_1\Big]\Delta_{j^\tau,s^\tau}r(\tilde{X}_1)\Big] \\
=&0
\end{aligned}
$$

By Assumption 2.2, 2.4, 2.5 and dominated convergence,

$$
\lim_{n\to\infty} J_n = \lim_{n\to\infty} \mathbb{E}\left[\left(\left(\sum_{i^\tau,t^\tau}^{n,T} W_{n,i^\tau,t^\tau}(D_{i^\tau,t^\tau}) - 1\right)\mathbb{E}[R_\tau|_u|\tilde{X}_1]\right)^2\right] = 0.
$$

and for

$$
\lim_{n\to\infty} D_n = \lim_{n\to\infty} \frac{\sigma^2_{\mathbb{E}[R_\tau|_u|\tilde{X}_j]}}{l_n} = 0
$$

where $\sigma^2_{\mathbb{E}[R_\tau|_u|\tilde{X}_j]}$ is the variance of $\mathbb{E}[R_\tau|_u|\tilde{X}_j]$. □

A.2 Proof of Proposition 2

This section will show that Assumptions 2.1 and 2.2 are satisfied by the K_n-NN path regression estimator.

This proposition in slightly varied forms is central to proving the consistency of the K_n-NN path regression estimator. To simplify readability we make several definitions: First, let $\rho \in \mathbb{N}$. Consider the multi-index for the data j^ρ, s^ρ and the data set D_{j^ρ,s^ρ} (as defined above). For any $l = 0, \dots, \tau$ denote the product,

$$V_{n,i^{l:\tau},t^{l:\tau}}(D_{j^\rho,s^\rho}) = \prod_{k=l}^{\tau} V_{n,[(i_{k-1},t_k k-1),(i_k,t_k)]} V_{n,[(i_{k-1},t_{k-1}),(i_k,t_k)]}(D_{j^\rho,s^\rho}).$$

For $l = 0, \dots \rho$, and the multi-index i^τ, t^τ denote

$$\begin{aligned}&\tilde{V}_{n,[(i_{k-1},t_{k-1}),(i_k,t_k)]}(D_{j^{l+1:\rho},s^{l+1:\rho}}, D_{j^{l:\rho},s^{l:\rho}}) \\ &\quad = \frac{1}{K_n} I_{(X_{i_k,t_k},U_{i_k,t_k}) \text{ is a } K_n-\text{NN in } D_{j^{l:\rho},s^{l:\rho}}}(X_{i_{k-1},t_{k-1}+1}, u_{t+1}(X_{i_{k-1},t_{k-1}+1}))\end{aligned}$$

where $(X_{i_{k-1},t_{k-1}}, U_{i_{k-1},t_{k-1}})$ are taken from the data set $D_{j^{l+1:\rho},s^{l+1:\rho}}$.

The variations of Proposition 1 needed to prove Proposition 2 will be contained in the following lemma:

Lemma 5. *Let Assumption 1.1 be satisfied. The following results hold for all $\tau \le T$.*

1. *For any multi-index for the data j^ρ, s^ρ where $\rho \in \mathbb{N}$, it holds that*

$$\mathbb{E}[V_{n,i^\tau,t^\tau}(D_{j^\rho,s^\rho})] \le (n-\tau-1)^{-|i^\tau|} K_n^{-(\tau+1-|i^\tau|)}.$$

2. *For $r = 1, \dots, \tau$ and $i_0 = i_r$, it holds that*

$$\begin{aligned}&\mathbb{E}[|V_{n,i_0,t_0}(\tilde{X}_1, u, D_{i^{r:\tau},t^{r:\tau}}) - V_{n,i_0,t_0}(\tilde{X}_1, u, D_{i^{r+1:\tau},t^{r+1:\tau}})| V_{n,i^{0:\tau},t^{0:\tau}}(D_{i^{r:\tau},t^{r:\tau}})] \\ &\le 2(n-\tau-1)^{-|i^\tau|} K_n^{-(\tau+1-|i^\tau|)}.\end{aligned}$$

3. *For $r = 0, \dots, \tau$; $l = 0, \dots, \tau-1$ and $i_{l+1} = i_r$,*

$$\begin{aligned}&\mathbb{E}[V_{n,i^l,t^l}(D_{i^{r+1:\tau},t^{r+1:\tau}}) \\ &\quad \cdot |\tilde{V}_{n,[(i_l,t_l),(i_{l+1},t_{l+1})]}(D_{j^{r+1:\tau},s^{r+1:\tau}}, D_{j^{r:\tau},s^{r:\tau}}) - V_{n,[(i_l,t_l),(i_{l+1},t_{l+1})]}(D_{j^{r:\tau},s^{r:\tau}})| \\ &\quad \cdot V_{n,i^{l+1:\tau},t^{l+1:\tau}}(D_{i^{r:\tau},t^{r:\tau}})] \\ &\le 2(n-\tau-1)^{-|i^\tau|} K_n^{-(\tau+1-|i^\tau|)}.\end{aligned}$$

4. *For $r = 0, \dots, \tau$; $l = 0, \dots, \tau-1$ and $i_l = i_r$,*

$$\begin{aligned}&\mathbb{E}[V_{n,i^l,t^l}(D_{i^{r+1:\tau},t^{r+1:\tau}}) \\ &\quad \cdot |V_{n,[(i_l,t_l),(i_{l+1},t_{l+1})]}(D_{j^{r+1:\tau},s^{r+1:\tau}}) \\ &\qquad - \tilde{V}_{n,[(i_l,t_l),(i_{l+1},t_{l+1})]}(D_{j^{r+1:\tau},s^{r+1:\tau}}, D_{j^{r:\tau},s^{r:\tau}})| \\ &\quad \cdot V_{n,i^{l+1:\tau},t^{l+1:\tau}}(D_{i^{r:\tau},t^{r:\tau}})] \\ &\le 2(n-\tau-1)^{-|i^\tau|} K_n^{-(\tau+1-|i^\tau|)}.\end{aligned}$$

5. *For $r = 0, \dots, \tau$ and $l = 0, \dots, \tau-1$, where either $l = r-1$ and $i_l \neq i_r$ or $i_l \neq i_r$ and $i_{l+1} \neq i_r$, it holds that*

$$\begin{aligned}&\mathbb{E}[V_{n,i^l,t^l}(D_{i^{r+1:\tau},t^{r+1:\tau}}) \\ &\quad \cdot |V_{n,[(i_l,t_l),(i_{l+1},t_{l+1})]}(D_{i^{r+1:\tau},t^{r+1:\tau}}) - V_{n,[(i_l,t_l),(i_{l+1},t_{l+1})]}(D_{i^{r:\tau},t^{r:\tau}})| \\ &\quad \cdot V_{n,i^{l+1:\tau},t^{l+1:\tau}}(D_{i^{r:\tau},t^{r:\tau}})] \\ &\le \begin{cases} 2(n-\tau-1)^{-|i^\tau|} K_n^{-(\tau+1-|i^\tau|)} & \text{if } |i^l| = |i^{l+1}| \\ 4(T+1)(n-\tau-1)^{-|i^\tau|} K_n^{-(\tau+2-|i^\tau|)} & \text{if } |i^l|+1 = |i^{l+1}| \end{cases}\end{aligned}$$

and if $r = 0$ *or* $i_0 \neq i_r$*, then*

$$\begin{aligned}&\mathbb{E}[|V_{n,i_0,t_0}(\tilde{X}_1, u, D_{i^{r:\tau},t^{r:\tau}}) - V_{n,i_0,t_0}(\tilde{X}_1, u, D_{i^{r+1:\tau},t^{r+1:\tau}})| \\ &\qquad \cdot V_{n,i^{0:\tau},t^{0:\tau}}(D_{i^{r:\tau},t^{r:\tau}})] \\ \leq &2(T+1)(n-\tau-1)^{-|i^\tau|}K_n^{-(\tau+2-|i^\tau|)}\end{aligned}$$

6. *For* $r = 0, \ldots, \tau$ *and* $l = 0, \ldots, \tau - 1$*, it holds that*

$$\begin{aligned}&\mathbb{E}[V_{n,i^l,t^l}(D_{i^{r+1:\tau},t^{r+1:\tau}}) \\ &\qquad \cdot |V_{n,[(i_l,t_l),(i_{l+1},t_{l+1})]}(D_{j^{r+1:\tau},s^{r+1:\tau}}) - V_{n,[(i_l,t_l),(i_{l+1},t_{l+1})]}(D_{j^{r:\tau},s^{r:\tau}})| \\ &\qquad \cdot V_{n,i^{l+1:\tau},t^{l+1:\tau}}(D_{i^{r:\tau},t^{r:\tau}})] \\ \leq &\begin{cases} 4(n-\tau-1)^{-|i^\tau|}K_n^{-(\tau+1-|i^\tau|)} & \text{if } |i^\tau| < \tau+1 \\ 16(T+1)(n-\tau-1)^{-|i^\tau|}K_n^{-(\tau+2-|i^\tau|)} & \text{if } |i^\tau| = \tau+1 \end{cases}\end{aligned}$$

and

$$\begin{aligned}&\mathbb{E}[|V_{n,i_0,t_0}(\tilde{X}_1, u, D_{i^{r:\tau},t^{r:\tau}}) - V_{n,i_0,t_0}(\tilde{X}_1, u, D_{i^{r+1:\tau},t^{r+1:\tau}})| \\ &\qquad \cdot V_{n,i^{0:\tau},t^{0:\tau}}(D_{i^{r:\tau},t^{r:\tau}})] \\ \leq &\begin{cases} 4(n-\tau-1)^{-|i^\tau|}K_n^{-(\tau+1-|i^\tau|)} & \text{if } |i^\tau| < \tau+1 \\ 2T(n-\tau-1)^{-|i^\tau|}K_n^{-(\tau+2-|i^\tau|)} & \text{if } |i^\tau| = \tau+1 \end{cases}\end{aligned}$$

7. *Consider two multi-indices* i^τ, t^τ *and* j^τ, s^τ *and two data sets with a finite number of replacements* $\tilde{D}_1$ *and* $\tilde{D}_2$ *compared to* D*, then*

$$\mathbb{E}[V_{n,i^\tau,t^\tau}(\tilde{D}_1)V_{n,j^\tau,s^\tau}(\tilde{D}_2)] \leq (n-2\tau-2)^{-|(i^\tau,j^\tau)|}K_n^{-(2\tau+2-|(i^\tau,j^\tau)|)}.$$

8. *Consider two multi-indices* i^τ, t^τ *and* j^τ, s^τ*. For* $r = 0, \ldots, \tau$ *and* $l = 0, \ldots, \tau - 1$*, it holds that*

$$\begin{aligned}&\mathbb{E}[V_{n,i^l,t^l}(D_{i^{r+1:\tau},t^{r+1:\tau}}) \\ &\qquad \cdot |V_{n,[(i_l,t_l),(i_{l+1},t_{l+1})]}(D_{i^{r+1:\tau},t^{r+1:\tau}}) - V_{n,[(i_l,t_l),(i_{l+1},t_{l+1})]}(D_{i^{r:\tau},t^{r:\tau}})| \\ &\qquad \cdot V_{n,i^{l+1:\tau},t^{l+1:\tau}}(D_{i^{r:\tau},t^{r:\tau}})V_{n,j^l,s^l}(D_{j^{r+1:\tau},s^{r+1:\tau}}) \\ &\qquad \cdot |V_{n,[(j_l,s_l),(j_{l+1},s_{l+1})]}(D_{j^{r+1:\tau},s^{r+1:\tau}}) - V_{n,[(j_l,s_l),(j_{l+1},s_{l+1})]}(D_{j^{r:\tau},s^{r:\tau}})| \\ &\qquad \cdot V_{n,j^{l+1:\tau},s^{l+1:\tau}}(D_{j^{r:\tau},s^{r:\tau}})] \\ \leq &\begin{cases} 8(n-2\tau-2)^{-|(i^\tau,j^\tau)|}K_n^{-(\tau+1-|(i^\tau,j^\tau)|)} & \text{if } |(i^\tau,j^\tau)| < 2\tau+2 \\ 4T^2(n-2\tau-2)^{-|i^\tau|}K_n^{-(\tau+3-|i^\tau|)} & \text{if } |(i^\tau,j^\tau)| = 2\tau+2 \end{cases}\end{aligned}$$

and

$$\begin{aligned}&\mathbb{E}[|V_{n,i_0,t_0}(\tilde{X}_1, u, D_{i^{r:\tau},t^{r:\tau}}) - V_{n,i_0,t_0}(\tilde{X}_1, u, D_{i^{r+1:\tau},t^{r+1:\tau}})| \\ &\qquad \cdot V_{n,i^{0:\tau},t^{0:\tau}}(D_{i^{r:\tau},t^{r:\tau}})|V_{n,j_0,s_0}(\tilde{X}_1, u, D_{j^{r:\tau},s^{r:\tau}}) - V_{n,j_0,s_0}(\tilde{X}_1, u, D_{j^{r+1:\tau},s^{r+1:\tau}})| \\ &\qquad \cdot V_{n,j^{0:\tau},s^{0:\tau}}(D_{j^{r:\tau},s^{r:\tau}})] \\ \leq &\begin{cases} 8(n-2\tau-2)^{-|(i^\tau,j^\tau)|}K_n^{-(\tau+1-|(i^\tau,j^\tau)|)} & \text{if } |(i^\tau,j^\tau)| < 2\tau+2 \\ 4(T+1)^2(n-2\tau-2)^{-|i^\tau|}K_n^{-(\tau+3-|i^\tau|)} & \text{if } |(i^\tau,j^\tau)| = 2\tau+2 \end{cases}\end{aligned}$$

9. *Consider two multi-indices i^τ, t^τ and j^τ, s^τ where $|(i^\tau, j^\tau)| = 2\tau + 2$. For $r = 0, \dots, \tau$ and $l = 0, \dots, \tau - 1$, it holds that*

$$\begin{aligned}
&\mathbb{E}[V_{n,i^\tau,t^\tau}(D_{i^\tau,t^\tau}) V_{n,j^l,s^l}(D_{(i^{r+1:\tau},j^\tau),(t^{r+1:\tau},s^\tau)}) \\
&\quad \cdot |V_{n,[(j_l,s_l),(j_{l+1},s_{l+1})]}(D_{(i^{r+1:\tau},j^\tau),(t^{r+1:\tau},s^\tau)}) \\
&\qquad - V_{n,[(j_l,s_l),(j_{l+1},s_{l+1})]}(D_{(i^{r:\tau},j^\tau),(t^{r:\tau},s^\tau)})| \\
&\quad \cdot V_{n,j^{l+1:\tau},s^{l+1:\tau}}(D_{(i^{r:\tau},j^\tau),(t^{r:\tau},s^\tau)})] \\
&\leq 2T(n - 2\tau - 2)^{-(2\tau+2)} K_n^{-1}
\end{aligned}$$

and

$$\begin{aligned}
&\mathbb{E}[V_{n,i^\tau,t^\tau}(D_{i^\tau,t^\tau}) \\
&\quad \cdot |V_{n,j_0,s_0}(\tilde{X}_1, u, D_{(i^{r+1:\tau},j^\tau),(t^{r+1:\tau},s^\tau)}) - V_{n,j_0,s_0}(\tilde{X}_1, u, D_{(i^{r:\tau},j^\tau),(t^{r:\tau},s^\tau)})| \\
&\quad \cdot V_{n,j^{0:\tau},s^{0:\tau}}(D_{(i^{r:\tau},j^\tau),(t^{r:\tau},s^\tau)})] \\
&\leq 2T(n - 2\tau - 2)^{-(2\tau+2)} K_n^{-1}
\end{aligned}$$

Proof. 1. The proof is completely analogous to Proposition 1.

2. Using the triangle inequality

$$\begin{aligned}
&\mathbb{E}[|V_{n,i_0,t_0}(\tilde{X}_1, u, D_{i^{r:\tau},t^{r:\tau}}) - V_{n,i_0,t_0}(\tilde{X}_1, u, D_{i^{r+1:\tau},t^{r+1:\tau}})| V_{n,i^{0:\tau},t^{0:\tau}}(D_{i^{r:\tau},t^{r:\tau}})] \\
&\leq \mathbb{E}[V_{n,i_0,t_0}(\tilde{X}_1, u, D_{i^{r:\tau},t^{r:\tau}}) V_{n,i^{0:\tau},t^{0:\tau}}(D_{i^{r:\tau},t^{r:\tau}})] \\
&\qquad + \mathbb{E}[V_{n,i_0,t_0}(\tilde{X}_1, u, D_{i^{r+1:\tau},t^{r+1:\tau}}) V_{n,i^{0:\tau},t^{0:\tau}}(D_{i^{r:\tau},t^{r:\tau}})]
\end{aligned}$$

The two terms can be treated as in Proposition 1.

3. The arguments are analogous to item 2.

4. The arguments are analogous to item 2.

5. We will only discuss the first upper-bound. The second is almost completely analogous.

Consider the case where $|i^l| = |i^{l+1}|$. For this use the triangle inequality as in the proof to Lemma 5.2 and then use the argument as in the induction step in the proof of Proposition 1.

If $|i^l| + 1 = |i^{l+1}|$, note that almost surely

$$\begin{aligned}
&|V_{n,[(i_l,t_l),(i_{l+1},t_{l+1})]}(D_{i^{r+1:\tau},t^{r+1:\tau}}) - V_{n,[(i_l,t_l),(i_{l+1},t_{l+1})]}(D_{i^{r:\tau},t^{r:\tau}})| \\
&= K_n^{-1}(I_{A^{(i_{l+1},t_{l+1})}_{(i_l,t_l)}} + I_{B^{(i_{l+1},t_{l+1})}_{(i_l,t_l)}})
\end{aligned}$$

where

$$\begin{aligned}
A^{(i_{l+1},t_{l+1})}_{(i_l,t_l)} &:= \Big\{ V_{n,[(i_l,t_l),(i_{l+1},t_{l+1})]}(D_{i^{r+1:\tau},t^{r+1:\tau}}) = 1 \\
&\qquad \text{and } V_{n,[(i_l,t_l),(i_{l+1},t_{l+1})]}(D_{i^{r:\tau},t^{r:\tau}}) = 0 \Big\} \\
B^{(i_{l+1},t_{l+1})}_{(i_l,t_l)} &:= \Big\{ V_{n,[(i_l,t_l),(i_{l+1},t_{l+1})]}(D_{i^{r+1:\tau},t^{r+1:\tau}}) = 0 \\
&\qquad \text{and } V_{n,[(i_l,t_l),(i_{l+1},t_{l+1})]}(D_{i^{r:\tau},t^{r:\tau}}) = 1 \Big\}.
\end{aligned}$$

For a data set D and two integers $a, b \in \mathbb{N}$ with $a < b$, define the set

$$C^{(i_{l+1},t_{l+1})}_{(i_l,t_l)}(D, a, b) \coloneqq \Big\{(X_{i_{l+1},t_{l+1}}, U_{i_{l+1},t_{l+1}}) \text{ is not a } (a-1)\text{-NN but a } b\text{-NN in } D\Big\}.$$

Note that $(X_{i_l,t_l+1}, U_{i_l,t_l+1})$ and $(X_{i_{l+1},t_{l+1}}, U_{i_{l+1},t_{l+1}})$ are the same random variables in $D_{i^{r:\tau},t^{r:\tau}}$ and in $D_{i^{r+1:\tau},t^{r+1:\tau}}$. Since at most T elements are exchanged from $D_{i^{r+1:\tau},t^{r+1:\tau}}$ compared to $D_{i^{r:\tau},t^{r:\tau}}$, the nearest neighbor ranking will at most change by $2T$ ranks. Thus, it holds that:

$$A^{(i_{l+1},t_{l+1})}_{(i_l,t_l)} \subset C^{(i_{l+1},t_{l+1})}_{(i_l,t_l)}(D_{i^{r+1:\tau},t^{r+1:\tau}}, K_n - 2T, K_n)$$
$$B^{(i_{l+1},t_{l+1})}_{(i_l,t_l)} \subset C^{(i_{l+1},t_{l+1})}_{(i_l,t_l)}(D_{i^{r+1:\tau},t^{r+1:\tau}}, K_n + 1, K_n + 2T + 1)$$

and

$$\begin{aligned}
&\mathbb{E}\Big[V_{n,i^l,t^l}(D_{i^{r+1:\tau},t^{r+1:\tau}}) | V_{n,[(i_l,t_l),(i_{l+1},t_{l+1})]}(D_{i^{r+1:\tau},t^{r+1:\tau}}) \\
&\qquad - V_{n,[(i_l,t_l),(i_{l+1},t_{l+1})]}(D_{i^{r:\tau},t^{r:\tau}})|\Big] \\
&\leq K_n^{-1}\mathbb{E}[V_{n,i^l,t^l}(D_{i^{r+1:\tau},t^{r+1:\tau}}) I_{C^{(i_{l+1},t_{l+1})}_{(i_l,t_l)}(D_{i^{r+1:\tau},t^{r+1:\tau}}, K_n-2T, K_n)}] \\
&\quad + K_n^{-1}\mathbb{E}[V_{n,i^l,t^l}(D_{i^{r+1:\tau},t^{r+1:\tau}}) I_{C^{(i_{l+1},t_{l+1})}_{(i_l,t_l)}(D_{i^{r+1:\tau},t^{r+1:\tau}}, K_n+1, K_n+2T+1)}]
\end{aligned}$$

We will prove the appropriate upper-bound for the first term. The upper-bound for the second term can be proved analogously.

Assume that

$$\begin{aligned}
&K_n^{-1}\mathbb{E}[V_{n,i^l,t^l}(D_{i^{r+1:\tau},t^{r+1:\tau}}) I_{C^{(i_{l+1},t_{l+1})}_{(i_l,t_l)}(D_{i^{r+1:\tau},t^{r+1:\tau}}, K_n-2T, K_n)}] \\
&\quad > 2T(n-l)^{-|i^{l+1}|} K_n^{|i^{l+1}|-l-3}
\end{aligned}$$

By the iid assumption across episodes it has to hold that

$$\begin{aligned}
&K_n^{-1}\mathbb{E}[V_{n,i^l,t^l}(D_{i^{r+1:\tau},t^{r+1:\tau}}) I_{C^{(j,t_{l+1})}_{(i_l,t_l)}(D_{i^{r+1:\tau},t^{r+1:\tau}}, K_n-2T, K_n)}] \\
&\quad > 2T(n-l)^{-|i^{l+1}|} K_n^{|i^{l+1}|-l-3}
\end{aligned}$$

for all j where $|(i^l, j)| = |i^l| + 1$. With Lemma 5.1, we then have that

$$\begin{aligned}
&\mathbb{E}[I_{C^{(j,t_{l+1})}_{(i_l,t_l)}(D_{i^{r+1:\tau},t^{r+1:\tau}}, K_n-2T, K_n)} | V_{n,i^l,t^l}(D_{i^{r+1:\tau},t^{r+1:\tau}}) = K_n^{-(l+1)}] \\
&> 2T(n-l)^{-1}.
\end{aligned}$$

Note that by definition it has to hold

$$\frac{1}{2T}\sum_{i=1}^{n}\sum_{t=0}^{T-1}\mathbb{E}[I_{C^{(i,t)}_{(i_l,t_l)}(D_{i^{r+1:\tau},t^{r+1:\tau}}, K_n-2T, K_n)} | V_{n,i^l,t^l}(D_{i^{r+1:\tau},t^{r+1:\tau}})] = 1.$$

Splitting the sum in a similar fashion as in the proof of Proposition 1 then yields

$$\begin{aligned}1 &\geq \frac{1}{2T}\sum_{\substack{i=1\\ i\neq i^l}}^{n}\sum_{t=0}^{T-1}\mathbb{E}[I_{C^{(i,t_{l+1})}_{(i_l,t_l)}(D_{i^{r+1:\tau},t^{r+1:\tau}},K_n-2T,K_n)}|V_{n,i^l,t^l}(D_{i^{r+1:\tau},t^{r+1:\tau}})]\\ &> \frac{2T(n-l)}{2T(n-l)} = 1\end{aligned}$$

This is a contradiction and, hence,

$$\begin{aligned}&K_n^{-1}\mathbb{E}[V_{n,i^l,t^l}(D_{i^{r+1:\tau},t^{r+1:\tau}})I_{C^{(i_{l+1},t_{l+1})}_{(i_l,t_l)}(D_{i^{r+1:\tau},t^{r+1:\tau}},K_n-2T,K_n)}]\\ &\quad\leq 2T(n-l)^{-|i^{l+1}|}K_n^{|i^{l+1}|-l-3}\end{aligned}$$

Using the same arguments as in the induction steps in the proof of Proposition 1 then yields the full result.

6. This result is a direct consequence from the previous four results.
7. Use the asymptotic behaviour known from Lemma 5.1 $\mathbb{E}[V_{n,i^\tau,t^\tau}(\tilde{D}_1)]$. Then use the same arguments as in the induction step in Proposition 1 for $\mathbb{E}[V_{n,i^\tau,t^\tau}(\tilde{D}_1)V_{n,j_0,s_0}(\tilde{X}_1,u,(\tilde{D}_2))]$ and for each $\mathbb{E}[V_{n,i^\tau,t^\tau}(\tilde{D}_1)V_{n,j^k,s^k}(\tilde{X}_1,u,(\tilde{D}_2))]$ where $k<\tau$.
8. This result is a direct combination of the arguments used in Lemma 5.6 and 5.7.
9. This result is again a direct combination of the arguments used in Lemma 5.6 and 5.7.

□

Remark 15. *Note that all the results in this section and all the following results can be easily extended to a version where the weights prevent reusing the same sample twice in a K_n-NN path by simply observing that the weights are zero in such cases. This corresponds to the implemented algorithm in some of the experiments (see Appendix Section A.3).*

Proof of Proposition 2. 1. By Lemma 5.7 it holds that

$$\begin{aligned}&\mathbb{E}\left[\sum_{i^\tau,t^\tau}^{n,T}\sum_{\substack{j^\tau,s^\tau\\ |(i^\tau,j^\tau)|<2\tau+2}}^{n,T}V_{n,i^\tau,t^\tau}(\tilde{D}_1)V_{n,j^\tau,s^\tau}(\tilde{D}_2)\right]\\ &\leq\sum_{i^\tau,t^\tau}^{n,T}\sum_{\substack{j^\tau,s^\tau\\ |(i^\tau,j^\tau)|<2\tau+2}}^{n,T}(n-2\tau-2)^{-|(i^\tau,j^\tau)|}K_n^{-(2\tau+2-|(i^\tau,j^\tau)|)}\\ &=T^{2\tau}\sum_{i^\tau}^{n}\sum_{\substack{j^\tau\\ |(i^\tau,j^\tau)|<2\tau+2}}^{n}(n-2\tau-2)^{-|(i^\tau,j^\tau)|}K_n^{-(2\tau+2-|(i^\tau,j^\tau)|)}\end{aligned}$$

where $\sum_{i^k}^n \coloneqq \sum_{i_0=1}^n\cdots\sum_{i_k=1}^n$. Fix an arbitrary $a\in\mathbb{N}$ with $a<2\tau+2$, then it holds that

$$T^{2\tau}\sum_{i^\tau}^{n}\sum_{\substack{j^\tau\\ |(i^\tau,j^\tau)|=a}}^{n}(n-2\tau-2)^{-a}K_n^{-(2\tau+2-a)} = \frac{n^a}{(n-2\tau-2)^aK_n^{2\tau+2-a}}\to 0$$

since $K_n\to\infty$. Since a was arbitrary the statement follows.

2. Note that

$$\begin{aligned}
&\mathbb{E}\left[\left(\sum_{i^\tau,t^\tau}^{n,T}|V_{n,i^\tau,t^\tau}(D_{i^\tau,t^\tau})-V_{n,i^\tau,t^\tau}(D)|\right)^2\right]\\
\leq&\mathbb{E}\left[\left(\sum_{i^\tau,t^\tau}^{n,T}\sum_{r=0}^{\tau}|V_{n,i^\tau,t^\tau}(D_{i^{r:\tau},t^{r:\tau}})-V_{n,i^\tau,t^\tau}(D_{i^{r+1:\tau},t^{r+1:\tau}})|\right)^2\right]\\
\leq&(\tau+1)\mathbb{E}\left[\sum_{r=0}^{\tau}\left(\sum_{i^\tau,t^\tau}^{n,T}|V_{n,i^\tau,t^\tau}(D_{i^{r:\tau},t^{r:\tau}})-V_{n,i^\tau,t^\tau}(D_{i^{r+1:\tau},t^{r+1:\tau}})\right)^2\right]
\end{aligned}$$

where we define $D_{i^{\tau+1:\tau},t^{\tau+1:\tau}}\coloneqq D$. Furthermore, for any $r=0,\dots,\tau$, we have

$$\begin{aligned}
&\mathbb{E}\left[\left(\sum_{i^\tau,t^\tau}^{n,T}|V_{n,i^\tau,t^\tau}(D_{i^{r:\tau},t^{r:\tau}})-V_{n,i^\tau,t^\tau}(D_{i^{r+1:\tau},t^{r+1:\tau}})\right)^2\right]\\
\leq&\mathbb{E}\Bigg[\Bigg(\sum_{i^\tau,t^\tau}^{n,T}|V_{n,i_0,t_0}(\tilde{X}_1,u,D_{i^{r:\tau},t^{r:\tau}})-V_{n,i_0,t_0}(\tilde{X}_1,u,D_{i^{r+1:\tau},t^{r+1:\tau}})|\\
&\quad\cdot V_{n,i^{0:\tau},t^{0:\tau}}(D_{i^{r:\tau},t^{r:\tau}})\\
&\quad+\sum_{l=0}^{\tau-1}V_{n,i^l,t^l}(D_{i^{r+1:\tau},t^{r+1:\tau}})\\
&\qquad\cdot|V_{n,[(i_l,t_l),(i_{l+1},t_{l+1})]}(D_{j^{r+1:\tau},s^{r+1:\tau}})-V_{n,[(i_l,t_l),(i_{l+1},t_{l+1})]}(D_{j^{r:\tau},s^{r:\tau}})|\\
&\qquad\cdot V_{n,i^{l+1:\tau},t^{l+1:\tau}}(D_{i^{r:\tau},t^{r:\tau}})\Bigg)^2\Bigg]\\
\leq&(\tau+1)\mathbb{E}\Big[\Big(\sum_{i^\tau,t^\tau}^{n,T}|V_{n,i_0,t_0}(\tilde{X}_1,u,D_{i^{r:\tau},t^{r:\tau}})-V_{n,i_0,t_0}(\tilde{X}_1,u,D_{i^{r+1:\tau},t^{r+1:\tau}})|\\
&\qquad\cdot V_{n,i^{0:\tau},t^{0:\tau}}(D_{i^{r:\tau},t^{r:\tau}})\Big)^2\Big]\\
&+(\tau+1)\sum_{l=0}^{\tau-1}\mathbb{E}\Big[\Big(\sum_{i^\tau,t^\tau}^{n,T}V_{n,i^l,t^l}(D_{i^{r+1:\tau},t^{r+1:\tau}})\\
&\qquad\cdot|V_{n,[(i_l,t_l),(i_{l+1},t_{l+1})]}(D_{j^{r+1:\tau},s^{r+1:\tau}})-V_{n,[(i_l,t_l),(i_{l+1},t_{l+1})]}(D_{j^{r:\tau},s^{r:\tau}})|\\
&\qquad\cdot V_{n,i^{l+1:\tau},t^{l+1:\tau}}(D_{i^{r:\tau},t^{r:\tau}})\Big)^2\Big]
\end{aligned}$$

Fix an $l = 0, \dots, \tau - 1$,

$$
\begin{aligned}
&\mathbb{E}\Big[\Big(\sum_{i^\tau, t^\tau}^{n,T} V_{n,i^l,t^l}(D_{i^{r+1:\tau},t^{r+1:\tau}}) \\
&\quad \cdot |V_{n,[(i_l,t_l),(i_{l+1},t_{l+1})]}(D_{j^{r+1:\tau},s^{r+1:\tau}}) - V_{n,[(i_l,t_l),(i_{l+1},t_{l+1})]}(D_{j^{r:\tau},s^{r:\tau}})| \\
&\quad \cdot V_{n,i^{l+1:\tau},t^{l+1:\tau}}(D_{i^{r:\tau},t^{r:\tau}})\Big)^2\Big] \\
=&\mathbb{E}\Big[\sum_{i^\tau, t^\tau}^{n,T} \sum_{j^\tau, s^\tau}^{n,T} V_{n,i^l,t^l}(D_{i^{r+1:\tau},t^{r+1:\tau}}) \\
&\quad \cdot |V_{n,[(i_l,t_l),(i_{l+1},t_{l+1})]}(D_{j^{r+1:\tau},s^{r+1:\tau}}) - V_{n,[(i_l,t_l),(i_{l+1},t_{l+1})]}(D_{j^{r:\tau},s^{r:\tau}})| \\
&\quad \cdot V_{n,i^{l+1:\tau},t^{l+1:\tau}}(D_{i^{r:\tau},t^{r:\tau}}) V_{n,j^l,s^l}(D_{j^{r+1:\tau},s^{r+1:\tau}}) \\
&\quad \cdot |V_{n,[(j_l,s_l),(j_{l+1},s_{l+1})]}(D_{j^{r+1:\tau},s^{r+1:\tau}}) - V_{n,[(j_l,s_l),(j_{l+1},s_{l+1})]}(D_{j^{r:\tau},s^{r:\tau}})| \\
&\quad \cdot V_{n,j^{l+1:\tau},s^{l+1:\tau}}(D_{j^{r:\tau},s^{r:\tau}})\Big]
\end{aligned}
$$

For any $a \in \mathbb{N}$ with $a < 2\tau + 2$, Lemma5.8 implies

$$
\begin{aligned}
&\mathbb{E}\Big[\sum_{i^\tau, t^\tau}^{n,T} \sum_{\substack{j^\tau, s^\tau \\ |j^\tau, i^\tau| = a}}^{n,T} V_{n,i^l,t^l}(D_{i^{r+1:\tau},t^{r+1:\tau}}) \\
&\quad \cdot |V_{n,[(i_l,t_l),(i_{l+1},t_{l+1})]}(D_{j^{r+1:\tau},s^{r+1:\tau}}) - V_{n,[(i_l,t_l),(i_{l+1},t_{l+1})]}(D_{j^{r:\tau},s^{r:\tau}})| \\
&\quad \cdot V_{n,i^{l+1:\tau},t^{l+1:\tau}}(D_{i^{r:\tau},t^{r:\tau}}) V_{n,j^l,s^l}(D_{j^{r+1:\tau},s^{r+1:\tau}}) \\
&\quad \cdot |V_{n,[(j_l,s_l),(j_{l+1},s_{l+1})]}(D_{j^{r+1:\tau},s^{r+1:\tau}}) - V_{n,[(j_l,s_l),(j_{l+1},s_{l+1})]}(D_{j^{r:\tau},s^{r:\tau}})| \\
&\quad \cdot V_{n,j^{l+1:\tau},s^{l+1:\tau}}(D_{j^{r:\tau},s^{r:\tau}})\Big] \\
&\leq L \frac{n^a}{(n - 2\tau - 2)^a K_n^{(\tau+1-a)}} \to 0
\end{aligned}
$$

and

$$
\begin{aligned}
&\mathbb{E}\Big[\sum_{i^\tau, t^\tau}^{n,T} \sum_{\substack{j^\tau, s^\tau \\ |j^\tau, i^\tau| = 2\tau+2}}^{n,T} V_{n,i^l,t^l}(D_{i^{r+1:\tau},t^{r+1:\tau}}) \\
&\quad \cdot |V_{n,[(i_l,t_l),(i_{l+1},t_{l+1})]}(D_{j^{r+1:\tau},s^{r+1:\tau}}) - V_{n,[(i_l,t_l),(i_{l+1},t_{l+1})]}(D_{j^{r:\tau},s^{r:\tau}})| \\
&\quad \cdot V_{n,i^{l+1:\tau},t^{l+1:\tau}}(D_{i^{r:\tau},t^{r:\tau}}) V_{n,j^l,s^l}(D_{j^{r+1:\tau},s^{r+1:\tau}}) \\
&\quad \cdot |V_{n,[(j_l,s_l),(j_{l+1},s_{l+1})]}(D_{j^{r+1:\tau},s^{r+1:\tau}}) - V_{n,[(j_l,s_l),(j_{l+1},s_{l+1})]}(D_{j^{r:\tau},s^{r:\tau}})| \\
&\quad \cdot V_{n,j^{l+1:\tau},s^{l+1:\tau}}(D_{j^{r:\tau},s^{r:\tau}})\Big] \\
&\leq \tilde{L} \frac{n^{2\tau+2}}{(n - 2\tau - 2)^{2\tau+2} K_n^2} \to 0
\end{aligned}
$$

where $L, \tilde{L} > 0$ are some constants. Note that the analogous arguments can be applied to

$$\mathbb{E}\Big[\Big(\sum_{i^\tau,t^\tau}^{n,T} |V_{n,i_0,t_0}(\tilde{X}_1, u, D_{i^{r:\tau},t^{r:\tau}}) - V_{n,i_0,t_0}(\tilde{X}_1, u, D_{i^{r+1:\tau},t^{r+1:\tau}})| \cdot V_{n,i^{0:\tau},t^{0:\tau}}(D_{i^{r:\tau},t^{r:\tau}})\Big)^2\Big].$$

3. Note that

$$\mathbb{E}\Bigg[\sum_{i^\tau,t^\tau}^{n,T} \sum_{\substack{j^\tau,s^\tau \\ |(i^\tau,j^\tau)|=2\tau+2}}^{n,T} V_{n,i^\tau,t^\tau}(D_{i^\tau,t^\tau})|V_{n,j^\tau,s^\tau}(D_{j^\tau,s^\tau}) - V_{n,j^\tau,s^\tau}(D_{(i^\tau,j^\tau),(t^\tau,s^\tau)})|\Bigg]$$
$$\leq \sum_{r=0}^{\tau} \mathbb{E}\Bigg[\sum_{i^\tau,t^\tau}^{n,T} \sum_{\substack{j^\tau,s^\tau \\ |(i^\tau,j^\tau)|=2\tau+2}}^{n,T} V_{n,i^\tau,t^\tau}(D_{i^\tau,t^\tau}) \cdot \Big|V_{n,j^\tau,s^\tau}(D_{(i^{r:\tau},j^\tau),(t^{r:\tau},s^\tau)}) - V_{n,j^\tau,s^\tau}(D_{(i^{r+1:\tau},j^\tau),(t^{r+1:\tau},s^\tau)})\Big|\Bigg]$$

where $D_{(i^{\tau+1:\tau},j^\tau),(t^{\tau+1:\tau},s^\tau)} \coloneqq D_{j^\tau,s^\tau}$. For any $r = 0, \ldots, \tau$ it holds that

$$\mathbb{E}\Bigg[\sum_{i^\tau,t^\tau}^{n,T} \sum_{\substack{j^\tau,s^\tau \\ |(i^\tau,j^\tau)|=2\tau+2}}^{n,T} V_{n,i^\tau,t^\tau}(D_{i^\tau,t^\tau}) \cdot \Big|V_{n,j^\tau,s^\tau}(D_{(i^{r:\tau},j^\tau),(t^{r:\tau},s^\tau)}) - V_{n,j^\tau,s^\tau}(D_{(i^{r+1:\tau},j^\tau),(t^{r+1:\tau},s^\tau)})\Big|\Bigg]$$
$$\leq \sum_{i^\tau,t^\tau}^{n,T} \sum_{\substack{j^\tau,s^\tau \\ |(i^\tau,j^\tau)|=2\tau+2}}^{n,T} \mathbb{E}[V_{n,i^\tau,t^\tau}(D_{i^\tau,t^\tau}) \cdot |V_{n,j_0,s_0}(\tilde{X}_1, u, D_{(i^{r+1:\tau},j^\tau),(t^{r+1:\tau},s^\tau)}) - V_{n,j_0,s_0}(\tilde{X}_1, u, D_{(i^{r:\tau},j^\tau),(t^{r:\tau},s^\tau)})| \cdot V_{n,j^{0:\tau},s^{0:\tau}}(D_{(i^{r:\tau},j^\tau),(t^{r:\tau},s^\tau)})]$$
$$+ \sum_{l=0}^{\tau-1} \sum_{i^\tau,t^\tau}^{n,T} \sum_{\substack{j^\tau,s^\tau \\ |(i^\tau,j^\tau)|=2\tau+2}}^{n,T} \mathbb{E}[V_{n,i^\tau,t^\tau}(D_{i^\tau,t^\tau})V_{n,j^l,s^l}(D_{(i^{r+1:\tau},j^\tau),(t^{r+1:\tau},s^\tau)}) \cdot |V_{n,[(j_l,s_l),(j_{l+1},s_{l+1})]}(D_{(i^{r+1:\tau},j^\tau),(t^{r+1:\tau},s^\tau)}) - V_{n,[(j_l,s_l),(j_{l+1},s_{l+1})]}(D_{(i^{r:\tau},j^\tau),(t^{r:\tau},s^\tau)})| \cdot V_{n,j^{l+1:\tau},s^{l+1:\tau}}(D_{(i^{r:\tau},j^\tau),(t^{r:\tau},s^\tau)})]$$

For any $l = 0, \dots, \tau - 1$, Lemma 5.9 implies

$$\sum_{i^\tau, t^\tau}^{n,T} \sum_{\substack{j^\tau, s^\tau \\ |(i^\tau, j^\tau)|=2\tau+2}}^{n,T} \mathbb{E}[V_{n,i^\tau,t^\tau}(D_{i^\tau,t^\tau}) V_{n,j^l,s^l}(D_{(i^{r+1:\tau},j^\tau),(t^{r+1:\tau},s^\tau)})$$
$$\cdot |V_{n,[(j_l,s_l),(j_{l+1},s_{l+1})]}(D_{(i^{r+1:\tau},j^\tau),(t^{r+1:\tau},s^\tau)})$$
$$- V_{n,[(j_l,s_l),(j_{l+1},s_{l+1})]}(D_{(i^{r:\tau},j^\tau),(t^{r:\tau},s^\tau)})|$$
$$\cdot V_{n,j^{l+1:\tau},s^{l+1:\tau}}(D_{(i^{r:\tau},j^\tau),(t^{r:\tau},s^\tau)})] \leq L \frac{n^{2\tau+2}}{(n-2\tau-2)^{2\tau+2} K_n} \to 0$$

where $L > 0$ is a constant. The arguments for the term

$$\sum_{i^\tau, t^\tau}^{n,T} \sum_{\substack{j^\tau, s^\tau \\ |(i^\tau, j^\tau)|=2\tau+2}}^{n,T} \mathbb{E}[V_{n,i^\tau,t^\tau}(D_{i^\tau,t^\tau})$$
$$\cdot |V_{n,j_0,s_0}(\tilde{X}_1, u, D_{(i^{r+1:\tau},j^\tau),(t^{r+1:\tau},s^\tau)}) - V_{n,j_0,s_0}(\tilde{X}_1, u, D_{(i^{r:\tau},j^\tau),(t^{r:\tau},s^\tau)})|$$
$$\cdot V_{n,j^{0:\tau},s^{0:\tau}}(D_{(i^{r:\tau},j^\tau),(t^{r:\tau},s^\tau)})]$$

is analogous.

□

A.3 Implementation

In this section, we present an extended version of the KNNR Algorithm from the main part by including a tree-based nearest neighbor search. We then state the MFMC Algorithm from [62, 63] and compare the complexity between MFMC and KNNR. Related to this we discuss why tree-based methods and parallelization in trajectory generation is not practical for the MFMC.

A.3.1 Extended KNNR algorithm

The main extension of the KNNR algorithm in comparison to the main part is the usage of search trees as a speed-up. Tree-based nearest neighbour search incurs an additional computational cost at the start for fitting a search tree on the given data set. This additional computational cost is more than off-set by a reduced cost for nearest neighbor queries. We will give more details on the complexity of tree-based nearest neighbour search below. For now, we will introduce two generic functions: First, TreeBuild$(\mathcal{D}, K)$ takes the entire data set $\mathcal{D}$ and the number of nearest neighbors K as arguments and returns a search tree for K-nearest neighbors. The second function TreeEval$((x, v), \mathcal{T})$ uses as arguments a state-action pair $(x, v) \in S$ and a search tree for K-nearest neighbors and outputs the K-nearest neighbors to (x, v). For actual implementation of the search trees popular choices are K-D Tree [20] and Ball Tree [163]. In our experiments, we use the heuristic provided in [171] to choose the nearest neighbor method.

Remark 16. *1. In our experiments instead of sampling independently from ν_0, we randomly and uniformly select initial states from the data sets.*

2. For the LOB and LQR experiment, we slightly alter the algorithm to not allow reusing the same transition twice within one nearest neighbor path. This does not change the complexity rates discussed below or the theoretical consistency and reduces the bias in these experiments.

Algorithm KNNR K-nearest neighbor resampling for OPE (extended)

Input: Data set $\mathcal{D}$; target policy u; nearest neighbors parameter K; metric d on S; number of resampled trajectories l

$R \leftarrow 0$
$\mathcal{T} \leftarrow$ TreeBuild$(\mathcal{D}, K)$
for j from 1 to l **do**
  $r \leftarrow 0$
  Sample an initial state $X_{j,0}^{\mathrm{NN}}$ from ν_0
  for s from 0 to T **do**
    Randomly choose $\mathcal{K}_s$ where $\mathcal{K}_s \sim \mathcal{V}(\{1, \dots, K\})$ and $\mathcal{K}^s \leftarrow (\mathcal{K}_0, \dots, \mathcal{K}_s)$
    Apply TreeEval$((X_{j,s}^{\mathrm{NN}}, u_s(X_{j,s}^{\mathrm{NN}})), \mathcal{T})$ to get K-nearest neighbor under d
    Choose the $\mathcal{K}_s$-nearest neighbor and denote them by $(X_{i_s,t_s}^{\mathcal{K}^s}, U_{i_s,t_s}^{\mathcal{K}^s})$ with the corresponding reward $R_{i_s,t_s}^{\mathcal{K}^s}$
    Set $r \leftarrow r + R_{i_s,t_s}^{\mathcal{K}^s}$ and $X_{j,s+1}^{\mathrm{NN}} \leftarrow X_{i_s,t_s+1}^{\mathcal{K}^s}$ if $s < T$
  end for
  $R \leftarrow r + R$
end for
Output: $\frac{R}{l}$

3. *In the experiments we use the standard Euclidean norm for nearest neighbor search in KNNR and MFMC for the LOB and LQR environment. In the BP environment we transform the state-action pair slightly before applying the Euclidean norm on the transformed state-action pair (see Appendix Section A.4).*

4. *A further speed up could be possibly achieved with using approximate nearest neighbors methods, e.g. [137]. However, it is unclear whether the theoretical guarantees can be extended to this setting. We will leave this consideration for future investigations.*

A.3.2 MFMC algorithm

Algorithm MFMC implements the model-free Monte Carlo algorithm proposed in [62, 63]. Note in particular that the data set is different at each iteration on which the nearest neighbor is searched. Moreover, the change in the data depends on previous iterations. Thus, one cannot parallelize the outer for-loop in Algorithm MFMC. This is in stark contrast to Algorithm KNNR where the calculations in the outer for-loop can be conducted without interdependencies, making parallelization viable for trajectory generation using the KNNR algorithm.

A.3.3 Complexity analysis

In this section, we discuss the complexity of the MFMC algorithm and the KNNR algorithm. For this purpose, recall that the computational complexity for a brute force K nearest neighbour search with n samples is $\mathcal{O}(1)$ at initialization and $\mathcal{O}(Kn)$ at evaluation.[2] For tree-based methods the computational costs are $\mathcal{O}(Kn\log(n))$ at initialization and $\mathcal{O}(K\log(n))$ at evaluation (cf. Documentation of [171]).

We will make the following assumptions on the parameter choices: For both KNNR and MFMC we assume that the number of resampled paths growths with the same order as the data set i.e. $l_n = \mathcal{O}(n)$. If any slower rate is chosen for l_n, the variance bounds in [62] become suboptimal. In the experiments we use the rate $l_n = \lfloor 0.1n \rfloor$ for both the MFMC and the KNNR. For K_n in KNNR, we assume $K_n = \mathcal{O}(n^{0.25})$ motivated by the empirical success in the experiments with continuous state-action spaces. As for the LQR and the LOB experiment we used the rate $K_n = \lfloor n^{0.25} \rfloor$. For the BP experiment, we got the best results with $K_n = 5I_{n<10^4} + 7I_{n=10^4}$. We

[2] Note that we omit dimension dependence for all complexities discussed here.

Algorithm MFMC Model-free Monte Carlo [62, 63]

Input: Data set $\mathcal{D}$; target policy u; metric d on S; number of resampled trajectories l

$R \leftarrow 0$
$\widetilde{\mathcal{D}} \leftarrow \mathcal{D}$
for j from 1 to l **do**
  $r \leftarrow 0$
  Sample an initial state $X_{j,0}^{\mathrm{NN}}$ from ν_0
  for s from 0 to T **do**
    Find the 1-nearest neighbor of $(X_{j,s}^{\mathrm{NN}}, u_s(X_{j,s}^{\mathrm{NN}}))$ in $\widetilde{\mathcal{D}}$ and denote them by $(X_{i_s,t_s}, U_{i_s,t_s})$ with the corresponding reward R_{i_s,t_s}
    Set $r \leftarrow r + R_{i_s,t_s}$, $\widetilde{\mathcal{D}} \leftarrow \widetilde{\mathcal{D}} \setminus \{(X_{i_s,t_s}, U_{i_s,t_s}, R_{i_s,t_s})\}$, and $X_{j,s+1}^{\mathrm{NN}} \leftarrow X_{i_s,t_s+1}$ if $s < T$
  end for
  $R \leftarrow r + R$
end for

Output: $\frac{R}{l}$

believe that the improved performance with a differing rate is potentially explained by the discretization of the state-action space in the BP experiment.

For the KNNR, we need to initialize the tree model at a cost of $\mathcal{O}(n^{1.25}\log(n))$ and search the tree for K_n nearest neighbors $l_n \cdot T$ times yielding the rate $\mathcal{O}(n^{1.25}\log(n))$ for evaluation. Thus, overall KNNR has the complexity $\mathcal{O}(n^{1.25}\log(n))$. The MFMC on the other hand looks for the nearest neighbour $l_n \cdot T$ times using a brute force nearest neighbour search yielding an overall rate of $\mathcal{O}(n^2)$. Note that a tree based method cannot improve this rate, since the data set is changing at each iteration. One would either need to fit $l_n \cdot T$ search trees or build one tree searching for $l_n \cdot T$ nearest neighbors. In either case, the computational cost would be of the order $\mathcal{O}(n^2\log(n))$ and, thus, worse than brute force search.

A.4 Experiments

In this section, we give more details on the experiments we conducted and reported on in Section 3.4 of the main part. In particular, we describe the three environments and comment on the implementation of the baselines.

A.4.1 Linear quadratic regulator

The first environment is the LQR problem proposed in [179] under the name of double integrator. We mainly follow the parametrization from [179] except that we choose a higher variance for the system noise. The environment is given as follows:

For $t = 1, \ldots, T$ with $T = 10$, the dynamic model is given by

$$X_{t+1} = AX_t + B\hat{u}_t + e_t \tag{A.1}$$

with the initial condition

$$X_0 = \begin{bmatrix} -1 + 0.5Z_1 \\ Z_2 \end{bmatrix} \tag{A.2}$$

where $X_t \in \mathbb{R}$ denotes the state, $\hat{u}_t \in \mathbb{R}$ the action, Z_1, Z_2 are independent standard normal random variables and

$$A = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}, \quad B = \begin{bmatrix} 0 \\ 1 \end{bmatrix}. \tag{A.3}$$

Furthermore, for all t, e_t is normal random vector with mean zero and covariance $10^{-2}I_2$ where I_n is the 2-dimensional identity matrix. The system noise is independent over time and from the initial noise. The rewards are given by $r_t = X_t^T Q X_t + R\hat{u}_t^2$ where

$$Q = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \text{ and } R = 0.5. \tag{A.4}$$

The behavior policy is given by

$$U_t = -K_t X_t \tag{A.5}$$

with $K_t = [K_{t,1}, K_{t,2}]^T$ where $K_{t,2}$ and $K_{t,2}$ are uniformly distributed on $[0, 1]$ and $[1, 2]$ respectively and independently for each t. As a target policy we use the policy given in [179]:

$$u_t = -\widehat{K} X_t \tag{A.6}$$

where

$$\widehat{K} = \left(R + B^T M B\right)^{-1} B^T M A \tag{A.7}$$

and M is the solution to the discrete algebraic Riccati equation

$$M = Q + A^T M A - \left(A^T M B\right)\left(R + B^T M B\right)^{-1}\left(B^T M A\right). \tag{A.8}$$

For reporting the performance on this environment in Figure 3.2 in the main part, we run our KNNR algorithm and the baselines a hundred times each for each data set size. We then compare the estimated performances of the target policy with the actual performance that we obtain by simulating 10^6 episodes sampled under the target policy.

A.4.2 Optimal execution in limit order book markets

The next environment arises in the optimal execution of financial assets in limit order book markets (e.g. [39] Chapter 8.2). We give a detailed description of the continuous-time version of this model:

- $\mathfrak{N}$ is the initial inventory of assets,
- T the terminal time,
- $S = (S_t)_{0 \le t \le T}$ a stochastic process, modelling the evolution of the mid-price, given by $S_t = S_0 + \sigma W_t, \sigma > 0$, where $W = (W_t)_{0 \le t \le T}$ is a standard Brownian motion,
- $\delta = (\delta)_{0 \le t \le T}$ the difference between the mid-price and the offered price, which is the control variable,
- $M = (M_t)_{0 \le t \le T}$ a Poisson process with intensity λ where M_t denotes the number of arriving orders,
- $N^\delta = \left(N_t^\delta\right)_{0 \le t \le T}$ the (controlled) counting process corresponding to the number of assets from the original inventory which have been sold,
- $P(\delta) = e^{-\kappa\delta}$, with $\kappa > 0$, the probability that an asset is being sold conditioned on an order arriving,
- $X^\delta = \left(X_t^\delta\right)_{0 \le t \le T}$ the reward process denoting the revenue of trade $dX_t^\delta = (S_t + \delta_t)\, dN_t^\delta$,
- $Q_t^\delta = \mathfrak{N} - N_t^\delta$ the remaining assets in the inventory.

Our value of policy δ is given by

$$V_\delta = \mathbb{E}\left[X_\tau^\delta + Q_\tau^\delta\left(S_\tau - \alpha Q_\tau^\delta\right) \mid X_{0^-}^\delta = 0, S_0, Q_{0^-}^\delta = \mathfrak{N}\right] \tag{A.9}$$

where $\alpha \geq 0$ determines the penalty for having inventory at time T (via a linear impact function) and $\tau = T \wedge \min\left\{t : Q_t^\delta = 0\right\}$ is a stopping time. As a behavior policy, we consider the random policy $\widehat{\delta}_t \sim \mathcal{V}[-0.01, 0.04]$ independently sampled for each time step and as target policy we consider the (non-stationary) optimal policy given by

$$\delta_t(q) = \frac{1}{\kappa}\left[1 + \log\frac{\sum_{n=0}^{q}\frac{\tilde{\lambda}^n}{n!}\exp(-\kappa\alpha(q-n)^2)(T-t)^n}{\sum_{n=0}^{q-1}\frac{\tilde{\lambda}^n}{n!}\exp(-\kappa\alpha(q-1-n)^2)(T-t)^n}\right] \tag{A.10}$$

where $\tilde{\lambda} = \lambda\exp(-1)$.

For the implementation, we discretize this model using a standard Euler-Maryuama approach and use the following parameters

$$\begin{aligned} &T = 20, \quad \lambda = 50/60, \quad \kappa = 100, \quad \alpha = 0.1 \quad \mathfrak{N} = 3, \\ &S_0 = 30, \quad \Delta t = 1 \text{ and } \quad \sigma = 0.1. \end{aligned}$$

where Δt denotes the discretization step-size.

Remark 17. *Note that the LOB environment can get to a terminal state before the time limit is reached and that if the inventory is not equal to zero at the last point in time an additional reward is generated dependent deterministically on the remaining inventory. We adapt the baselines and the KNNR algorithm accordingly. The modification to account for this in the KNNR algorithm are minimal and following the consideration in Remark 6 of the main part the estimation of the additional final reward is consistent.*

For reporting the performance on this environment in Figure 3.2 in the main part, we run our KNNR algorithm and the baselines 30 times each for each data set size. We then compare the estimated performances of the target policy with the actual performance that we obtain by simulating 10^6 episodes sampled under the target policy.

A.4.3 Online stochastic bin packing

Next we consider a slightly altered version of the stochastic bin packing problem (BP) as introduced in [13, 86]. For this model at each time period $t \in \{1, \ldots, T\}$ one item arrives where the item is of some size $j \in \{1, \ldots, J\}$ where $J, T \in \mathbb{N}$. At arrival the item needs to be packed in a bin of bin size $B \in \mathbb{N}$ where $J < B$. For that purpose an item j can be packed in any existing bin that has free space of at least $B - j$ or in a newly opened bin. The number of bins that can be newly opened is not restricted. Let $b \in \{1, \ldots, B\}$ denote all the possible levels of fill of a bin, then the state of this problem is of the form $X_t = ((N_{b,t})_{b=1}^{B-1}, j_t)$ where N_b denotes the number of open bins filled to the level b at time t and j_t the item arriving at time t. The action a is encoded as an integer where 0 signifies opening a new bin and any integer above zero marks a bin at level b at which an item is placed. The only modification compared to [13] is that we adapt the mechanism on how the item size is sampled at each period in time. We assume that j_t is sampled from a Poisson distribution truncated between 1 and J. The intensity parameter λ of this distribution depends on $((N_{b,t})_{b=1}^{B-1})$. We calculate the average free space available over all opened bins and set λ equal to this. If the average free space available over all opened bins is larger than the maximal item size or no bin is opened, we take λ as the maximal item size. [3] The reward R_t in this system is calculated as the negative increment on the total unused capacity, meaning that if an item

[3] In practice this mechanism leads to larger items arriving if more bins with a lot space remaining are open. This behaviour would be consistent with the implementation of a dynamic pricing mechanism that increases prices as less capacity is available.

of size j is placed in an existing bin the reward is given by the item size j. If a new bin needs to be opened the reward is $-(B-j)$. The quantity of interest is the expected sum of rewards over the episode:

$$\mathbb{E}\Big[\sum_{t=1}^{T} R_t\Big] \tag{A.11}$$

As a behavior policy, we choose a random policy which at each time step randomly samples uniformly from all feasible actions. As a target policy we use the sum of squares heuristic introduced and explained in [13]. In our implementation, we use the following parameters

$$T = 30, \quad J = 9 \text{ and } \quad B = 10.$$

For reporting the performance on this environment in Figure 3.2 in the main part, we run our KNNR algorithm and the baselines 30 times each for each data set size. We then compare the estimated performances of the target policy with the actual performance that we obtain by simulating $0.5 \cdot 10^5$ episodes sampled under the target policy.

Remark 18. *Strictly speaking we do not use the Euclidean norm directly in the state-action space for the KNNR and MFMC algorithm, but we project the state-action space onto the bin counters and apply the Euclidean norm only on the bin counters for nearest neighbor search. To make this more explicit, introduce the following function $\pi : \mathbb{N}^{B+2} \to \mathbb{N}^B$ where $\pi(X_t, a_t)$ maps the current state and action to the tuple of open bins $(N_{t^+,b})_{b=1}^{B-1}$ after action a_t has been taken. This means, if $a_t > 0$, for $b = a_t$, $N_{t^+,b} = N_{t,b} - 1$ and, if $a_t + j_t < B$, $N_{t^+,a_t+j_t} = N_{t,a_t+j_t} + 1$. If $a_t = 0$, $N_{t^+,j_t} = N_{t,j_t} + 1$. In other words, we exploit that the instant change of the state due to the action is known and that only this modified state impacts the distribution of the next state. This is often the case in applications and makes our algorithm applicable to situations where the actions are not directly observable but the state is.*

A.4.4 Baselines

Since the baselines used for our experiments are well-known, we will refrain from discussing the general approaches and only point out details in the implementation. The MFMC was discussed at length in a previous section. For the other baselines we generally refer to the Appendix of [214]. For the PEIS and PDIS we note that these refer to the weighted versions of importance sampling. For PEIS, PDIS and WDR in the LOB and LQR setting, we randomize the target policy for calculating the importance weights (since otherwise they would not be well-defined). We assume a normal distribution with the actual target policy as the mean and a constant standard deviation (LQR:0.01; LOB:0.005). The complexity of PEIS, PDIS and WDR is $O(n)$. For the FQE baselines we adapt them to the episodic setting i.e. Appendix of [94]. We consider two different regressions for the FQE. FQE Lin uses a linear regression with a linear-quadratic feature transformation of the state-action pair. FQE NN uses a nearest neighbor regression directly on the state-action pair. For the LOB and LQR setting we choose the number of nearest neighbors in the regression as $n^{0.25}$ and for the BP environment as $n^{0.5}$ where n denotes the number of episodes in the data set. These choices are based on empirical performance. The complexity of FQE depends on the chosen regression model. For FQE Lin, it is $O(n)$ and for FQE NN, either $O(n^{1.25}\log(n))$ or $O(n^{1.5}\log(n))$ depending on the chosen number of nearest neighbors. For the WDR we use FQE Lin as the model component. Finally, the naive average of the rewards under the behavior policy (NA) is self-explanatory and has the complexity $O(n)$.

Appendix B

Appendix: Limit Order Book Simulation and Trade Evaluation with K-Nearest-Neighbor Resampling

B.1 Benchmarks

This section contains implementation details and results for the comparison of the benchmarks with the real market. For the results, we use the same data and follow the same data split. The resulting plots are generated in the same way as in the main part.

B.1.1 Naive sampling

The first benchmark is a naive sampling of LOB transitions from the data. The sampling is uniform over the data set and unconditional. To report the returns, randomly selected transitions are chained together. This benchmark solely relies on the unconditional law of large numbers and does not capture any conditional behavior of the LOB dynamics.

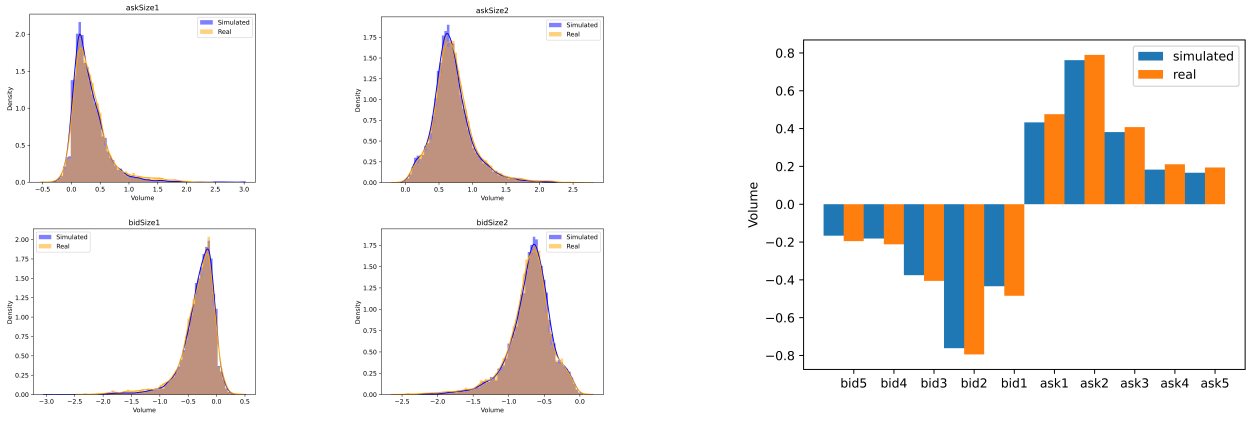


Figure B.1: Volume comparison for naive sampling: Marginal volumes per order book level after one transition (left); Average LOB shape after one transition (right)

Figure B.1 corresponds to Figure 4.6 in the main part. In Figure B.1, we observe that naive sampling reproduces the marginal volume distributions relatively well although with more discrepancies compared to Algorithm KNNR-LOB (cf. Table 4.1). The average LOB volumes have a similar shape but are slightly lower compared to the transitions from the test data. The discrepancies reveal a slight distributional shift between the training data that was used for naive sampling and the test data used for the real transitions. The LLN should give the same results if the sampling distributions are the same (up to the sampling error). This sampling error becomes negligible especially for the average LOB shapes with 10^4 samples. While naive sampling is not capable to account for this shift, Algorithm KNNR-LOB with its conditional resampling is able to adapt.

Figure B.2 corresponds to Figure 4.7 in the main part. The first plot shows that the unconditional LLN is relatively efficient in reproducing the static correlation structure between different order book levels. However, the naive method is unable to reproduce the correlation structure of changes in volumes, as can be seen in the second plot of Figure B.2. As the naive sampling approach can be seen as independent of the initial state, a simple calculation on the correlation coefficient reveals that the correlation of the differences simply corresponds to the correlation of the static volumes for the naive sampling method. This reveals an important distinction between naive sampling and Algorithm KNNR-LOB, which is capable of reproducing the correlation structure of changes in volume.

Figure B.3 corresponds to Figure 4.8 in the main part. In Figure B.3, we compare the mid and the weighted mid-return of the naive sampling method with real return series. We observe that for both plots the average returns coincide at zero. Note that all quantiles of the simulated returns expand faster compared to the quantiles of the real return time series. The slower expansion of the real return distribution indicates that a price movement in the same direction is less likely to occur right after another. In contrast to Algorithm KNNR-LOB, the naive sampling method does not capture this conditional price movement behavior.

Figure B.4 corresponds to Figure 4.9 in the main part. The first plot in Figure B.4 reveals that there is no correlation of the mid-price return for the naive sampling method. The second plot for weighted mid-price returns shows a small positive correlation that lasts for a short duration. After that, the correlation becomes statistically insignificant. Considering the covariance structure of the returns, this is most likely explained by the correlation of the initial weighted price with the real return time series. We note in particular that the weighted price contains more information on the order book shape compared to the unweighted mid-price. In comparison to Algorithm KNNR-LOB, we observe that the correlation for naive sampling vanishes faster and is smaller in scale. This further points to the fact that Algorithm KNNR-LOB is capable of extracting the conditional influence of the initial state beyond the information contained in the initial weighted price.

B.1.2 CGAN

The second benchmark we implement is the generative adversarial neural network for conditional LOB simulation proposed in [48]. Wherever possible, we follow the implementation as stated in the original paper and refer to that paper for implementation details. In cases where the original paper does not specify concrete parameter values, we conducted a heuristic hyperparameter search[1]. We present the model giving the best performance for the KS test statistics as reported in Table 4.1. Furthermore, the results presented below confirm that the CGAN method is applicable to our data set and that the chosen model is a good LOB simulator.

For this method, we only simulate the three highest levels of the LOB instead of the five highest. As the CGAN method encodes price movements within a given window of price levels, the two extra order book levels are needed as a buffer in the generation of the training data set. Furthermore, this implementation is consistent with [48] who also consider the three highest levels of the order book.

Figure B.5 corresponds to Figure 4.6 in the main part. The left-hand plots show that the CGAN method is well capable of reproducing the marginal distributions of volumes for our data set. The right-hand plot demonstrates that also the average order book shape produced by the CGAN method fits the real transitions

[1] By hyperparameter search, we choose the noise seed with 10 dimensions and a batch size of 256 samples.

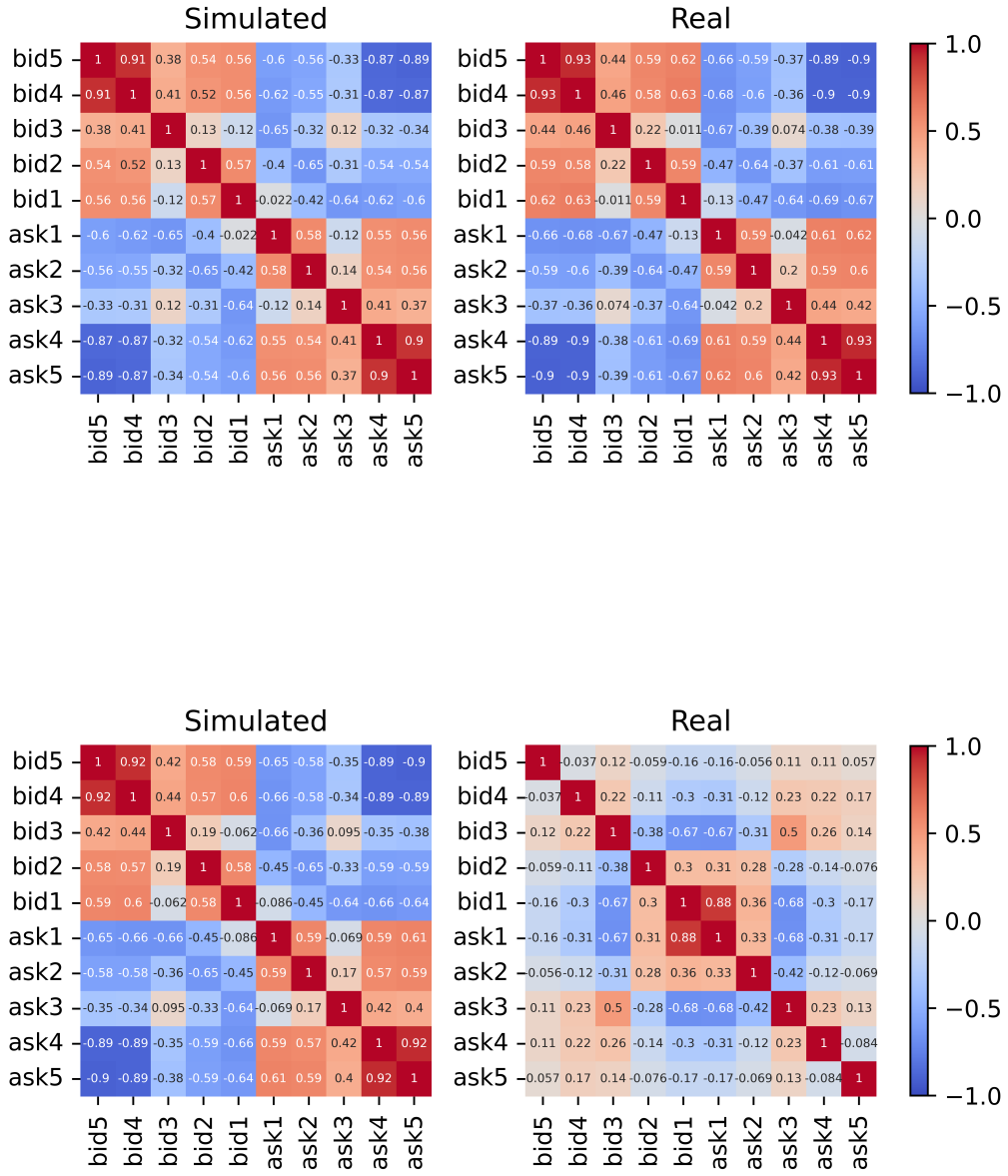


Figure B.2: Correlation for naive sampling: Absolute Volumes (first row) Volume differences (second row)

well. Note that the missing averages for the forth and fifth highest order book level are due to the encoding of price changes as explained above.

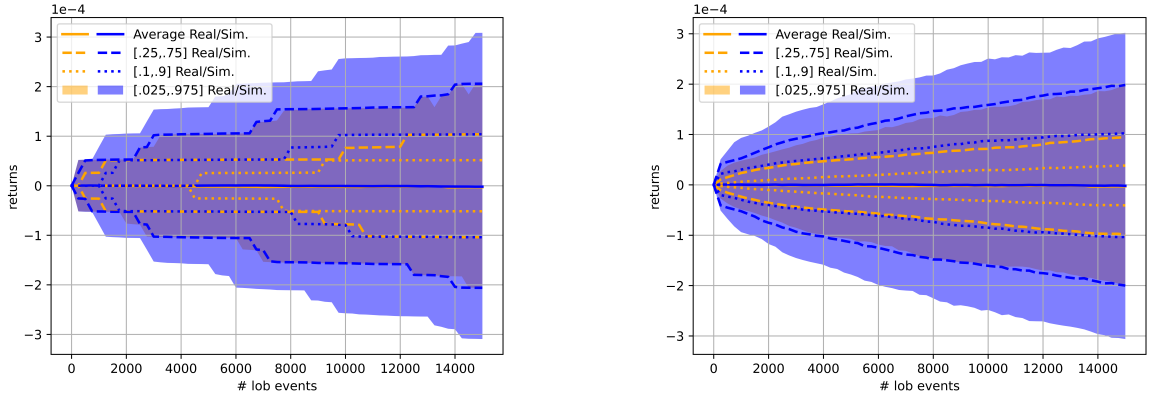


Figure B.3: Unconditional Return for naive sampling: Mid-price (left); Weighted mid-price (right)

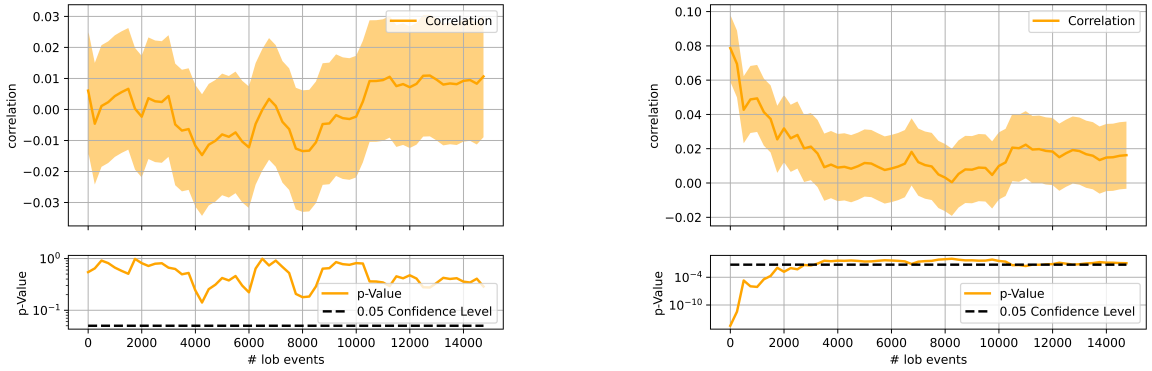


Figure B.4: Return Correlation for naive sampling: Mid-Price (left); Weighted mid-price (right)

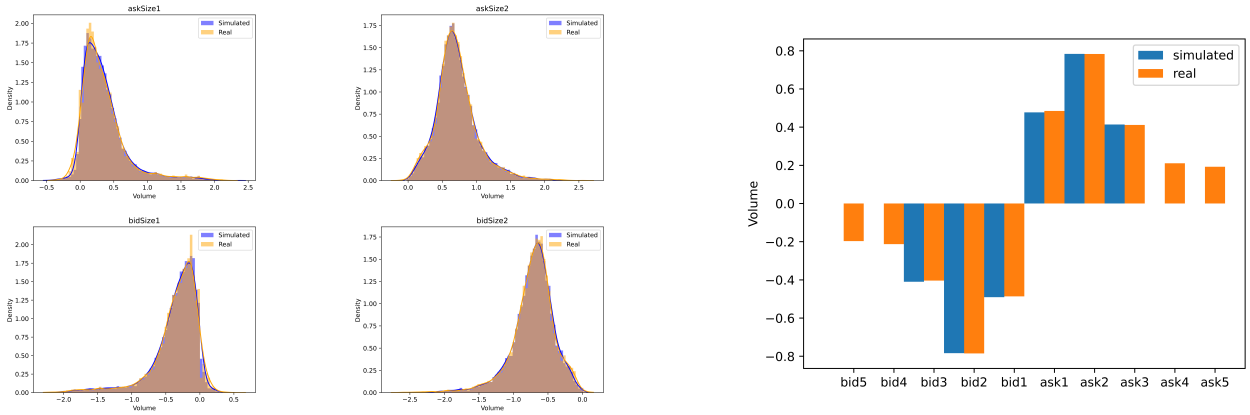


Figure B.5: Volume Comparison for CGAN: Marginal volumes per order book level after one transition (left) Average LOB shape after one transition

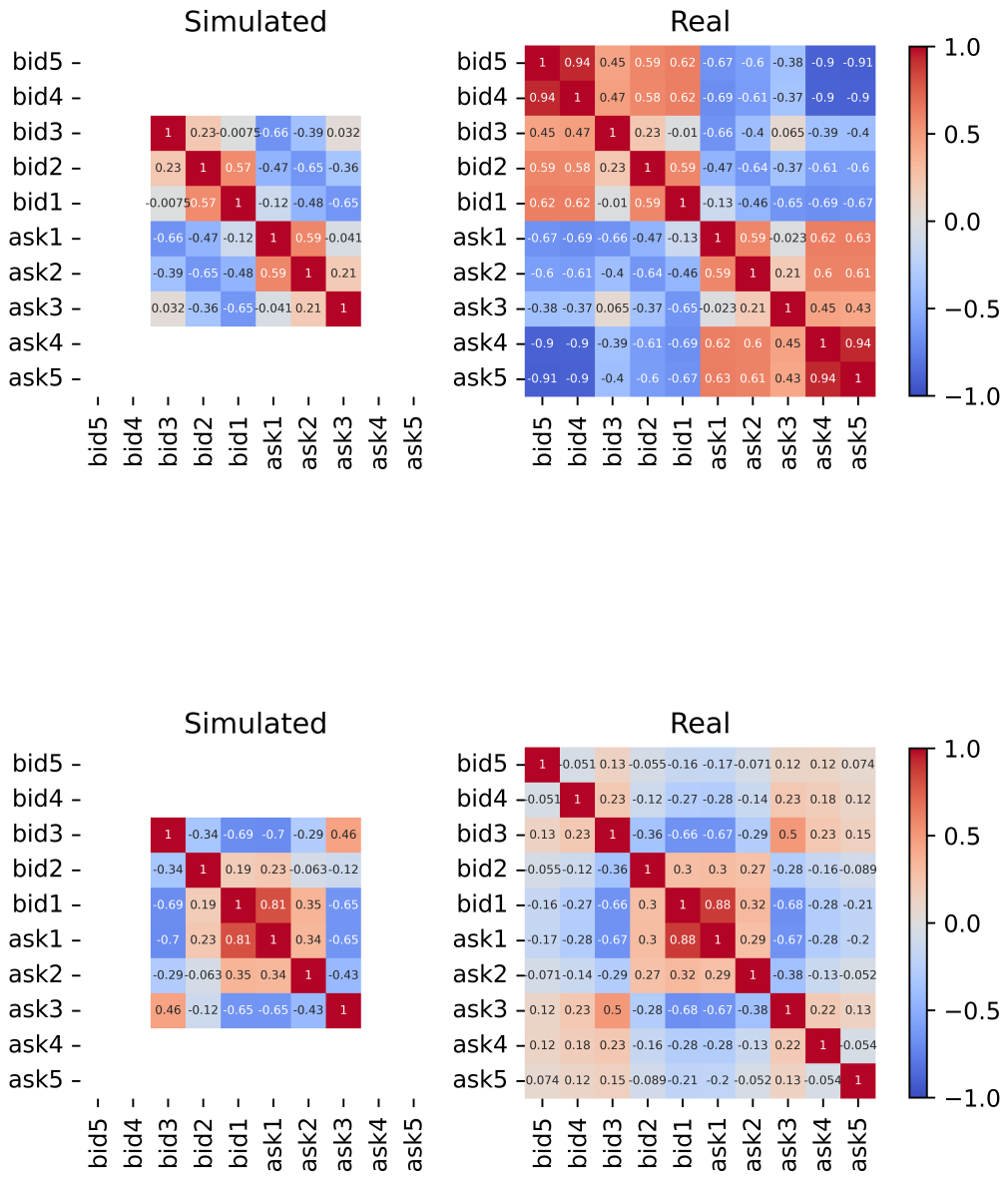


Figure B.6: Correlation for CGAN: Absolute Volumes (first row) Volume differences (second row)

Figure B.6 corresponds to Figure 4.7 in the main part. In the first plot in Figure B.6, we observe that the CGAN method is effective in capturing the correlation structure for volume levels and only shows minor

differences. The second plot in Figure B.6 reveals an overall good fit in the correlation of volume changes pointing to the effectiveness in conditional sampling of the CGAN method. For a few volume levels, we see some larger differences in correlations, e.g. ask2-bid2.

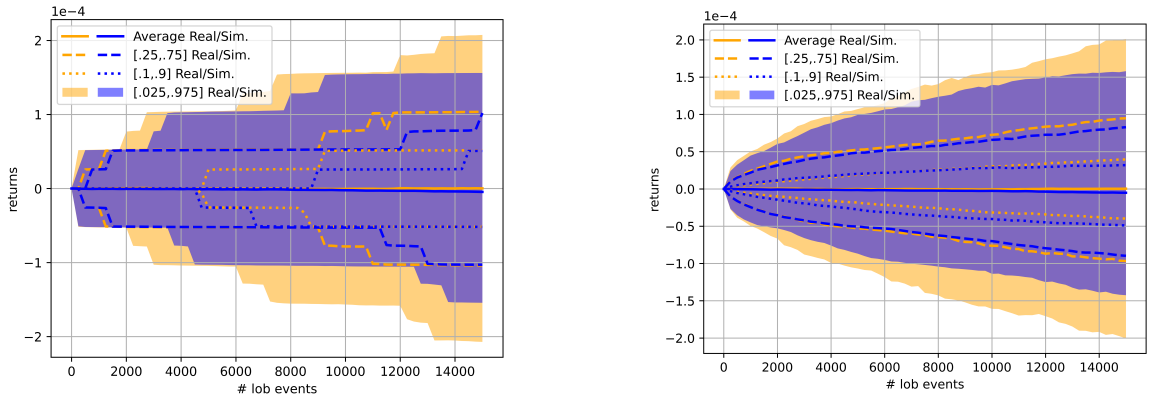


Figure B.7: Unconditional Return for CGAN: Mid-price (left); Weighted mid-price (right)

Figure B.7 corresponds to Figure 4.8 in the main part. For both the weighted and the unweighted return series the central quantiles of the simulation fit well with the central quantiles of the real returns time series. As in the results for Algorithm KNNR-LOB, the tails of the real returns extend further than the tails of the simulation.

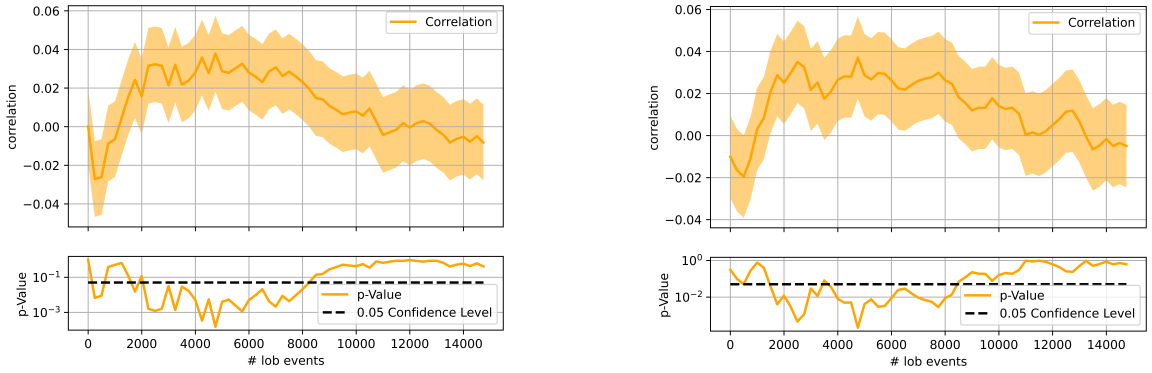


Figure B.8: Return Correlation for CGAN: Mid-Price (left); Weighted mid-price (right)

Figure B.8 corresponds to Figure 4.9 in the main part. Figure B.8 reveals a correlation between the real and the simulated return time series for both weighted and unweighted mid-price. Surprisingly, the correlation is initially small for a short time and then becomes positive before vanishing again.

To conclude, we compare some other aspects of the CGAN approach with Algorithm KNNR-LOB. First, we note that training CGAN requires us to solve a complicated optimization problem that is potentially unstable (cf. [48]). In contrast, Algorithm KNNR-LOB does not require any optimization. Once trained, the parameters of the CGAN encode all learned market dynamics. First, this has the advantage that the system noise can be easily controlled in evaluation, which simplifies counterfactual comparisons. Second, the evaluation complexity of a CGAN data generator is constant in data set size. Computational costs for evaluation phase for algorithm KNNR-LOB on the other hand grow on average logarithmically in the data set size ([69]). The disadvantage of

encoding the market dynamics in a neural network is that the simulator is effectively a black box and unrealistic order book transitions cannot be ruled out. Algorithm KNNR-LOB is based on a simple rule based assignment and each sampled transition stems from a historical observation. Finally, the CGAN method as presented in [48] did not address evaluating trading strategies with limit orders, which we included in Algorithm KNNR-LOB.