Reinforcement Learning in Queue-Reactive Models: Application to Optimal Execution

Tomas Espana[1*] Yadh Hafsi[2†] Fabrizio Lillo[3‡] Edoardo Vittori[4§]

[1] ORFE, Princeton University, Princeton, NJ, USA
[2] CMAP, École Polytechnique, Palaiseau, France
[3] Scuola Normale Superiore, Pisa, Italy
[4] Intesa Sanpaolo, Milan, Italy

November 20, 2025

Abstract

We investigate the use of Reinforcement Learning for the optimal execution of meta-orders, where the objective is to execute incrementally large orders while minimizing implementation shortfall and market impact over an extended period of time. Departing from traditional parametric approaches to price dynamics and impact modeling, we adopt a model-free, data-driven framework. Since policy optimization requires counterfactual feedback that historical data cannot provide, we employ the Queue-Reactive Model to generate realistic and tractable limit order book simulations that encompass transient price impact, and nonlinear and dynamic order flow responses. Methodologically, we train a Double Deep Q-Network agent on a state space comprising time, inventory, price, and depth variables, and evaluate its performance against established benchmarks. Numerical simulation results show that the agent learns a policy that is both strategic and tactical, adapting effectively to order book conditions and outperforming standard approaches across multiple training configurations. These findings provide strong evidence that model-free Reinforcement Learning can yield adaptive and robust solutions to the optimal execution problem.

Keywords : Optimal Execution, Reinforcement Learning, Queue-Reactive Model, Limit Order Book, Market Microstructure, Price impact.

1 Introduction

Executing large orders efficiently is a fundamental challenge in electronic financial markets. Large transactions, typically initiated by institutional investors such as banks, asset managers, hedge funds, or proprietary trading firms, often consume visible liquidity across multiple price levels of the limit order book (LOB), generating significant price impact. This impact is manifested through both immediate price shifts and subsequent order flow reactions and motivates the need for execution strategies that optimally balance market impact against timing risk. Consequently, the design of optimal execution strategies has become a central topic in market microstructure and algorithmic trading research.

---

*tomas.espana@princeton.edu

†yadh.hafsi@polytechnique.edu, [ORCID]

‡fabrizio.lillo@sns.it, [ORCID]

§edoardo.vittori@intesasanpaolo.com

Several studies have deepened the theoretical and empirical understanding of market impact. Refs [8] and [7] documented the transient and concave nature of impact, highlighting its origin in the interplay between order flow, liquidity consumption, and market participant behavior. These empirical findings motivated the introduction of new constraints and mechanisms in the optimal execution problem. The first formal treatment of the problem is due to [6], who derive closed-form solutions under specific assumptions using dynamic programming. This framework was extended by [3], who introduced both permanent and temporary market impact components, as well as a risk-aversion parameter. Subsequent research has generalized the Almgren-Chriss model along several dimensions (see [4, 31, 23, 2, 40, 15, 25, 12, 11, 21, 22, 13, 17, 18]). Notably, [24] established that transient impact models must satisfy a no-dynamic-arbitrage condition, which tightly links the functional form of impact decay to the underlying price dynamics. This result rules out many ad-hoc impact kernels and provides structural constraints on any admissible transient impact model. [40] further developed this direction by introducing a continuous-time propagator model in which impact decays through limit-order-book resilience.

Although these developments significantly enriched our understanding of execution costs and impact mechanisms, they still rely on strong parametric and structural assumptions that might be in contrast with the true data generating process and might be difficult to calibrate empirically. Model parameters such as impact coefficients, volatility processes, and resilience rates remain highly context-dependent and vary across time and assets, limiting the robustness of these approaches. In fact, traditional methods for solving stochastic control problems face limitations. Closed-form solutions are rare and require restrictive assumptions on the model dynamics and objective function. Numerical approaches typically involve solving the Hamilton-Jacobi-Bellman (HJB) equation, quasi-variational inequalities, or backward stochastic differential equations (BSDEs). These formulations often rely on viscosity solution theory to establish existence, uniqueness, and regularity of the value function. Yet, the associated numerical schemes, such as Monte Carlo methods for BSDEs and finite-difference methods for PDEs, suffer from the curse of dimensionality. As a result, these techniques are generally limited to problems with low-dimensional state spaces.

Related Works. To address these limitations, reinforcement learning methods [46] have emerged as data-driven and model-free alternatives for sequential decision problems. Unlike classical approaches, reinforcement learning (RL) does not require strong parametric assumptions on market dynamics. In the context of optimal execution, RL provides a flexible framework to learn trading policies directly from simulated or historical market data. The first application of RL to optimal execution was introduced by [38], who trained a tabular Q-learning agent to minimize implementation shortfall using historical trade data. Building on this, [28] proposed a hybrid approach that combined the Almgren–Chriss (AC) model with Q-learning, allowing the agent to adjust its execution trajectory dynamically based on the observed market state. To overcome the limitations of tabular methods in high-dimensional settings, [39] employed Double Deep Q-Networks (DDQN) (see [35, 48]), integrating deep neural networks with Q-learning to generalize across continuous state spaces and mitigate value overestimation bias. More recently, [34] applied DDQN within the AC framework under time-varying liquidity conditions, demonstrating improved performance relative to benchmark strategies, while [33] employed Proximal Policy Optimization (PPO) (see [45]) to train agents that learn directly from limit order book data using sparse reward structures.

Existing approaches rely on specific market impact assumptions and on historical data, which do not capture how market conditions evolve in response to trading actions. It therefore remains unclear what reinforcement learning algorithms can learn in more realistic, microstructure-based settings that incorporate endogenous market impact. In this work, we address this question using a limit order book (LOB) simulator that models both

direct market impact, from liquidity consumption, and indirect impact, from participants' reactions to trades. Many market simulators have been proposed in the literature, including agent-based models [10, 1, 14, 27] and generative models [32, 19, 20, 36, 37]. Recent studies have explored such simulation-based frameworks [26, 16]. While these approaches can capture complex market behaviors, they can be more difficult to implement and calibrate, and, more importantly, they often lack analytical tractability.

Main Contributions. We employ the Queue Reactive Model (QRM) (see [29]). The QRM provides a realistic yet tractable description of short-term limit-order-book dynamics. Instead of assuming that price changes follow a simple diffusion, it treats the best bid and ask as queues of standing orders that evolve through limit-order arrivals, cancellations, and market-order executions. The rates at which these events occur depend on the current state of the book, particularly the size imbalance between the bid and ask queues, which means that the model naturally links order-flow pressure to short-term price movements. When either queue is depleted, the price moves by one tick, and new queue sizes are redrawn from empirical distributions calibrated to market data. This simple mechanism reproduces key microstructure features such as liquidity resilience, mean-reverting price behavior, and transient price impact: a trade that consumes liquidity temporarily shifts the state of the book, altering subsequent order flow and gradually decaying as liquidity replenishes. Because QRM captures both direct and indirect market impact within a statistically grounded framework, it serves as a practical simulator of endogenous market reactions.

The goal of this paper is to propose a methodology to build a theoretically grounded and microstructure-consistent training and testing environment for RL. Specifically, the QRM is analytically tractable and ergodic, reproducing key stylized facts of LOB dynamics such as resilience, transient impact, and order-flow imbalance. We use model parameters calibrated from real market data and train an RL agent in this environment to learn execution strategies that generalize across endogenous market states. In contrast to existing approaches, our framework captures both direct impact (via liquidity consumption) and indirect impact (via feedback effects on order flow), offering a more realistic and interpretable platform for learning execution policies.
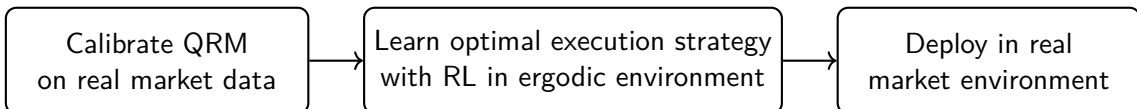

Figure 1: Proposed methodology: learning optimal execution strategies using RL.

This paper is organized as follows. Section 2 formalizes the optimal execution problem and establishes the notation used throughout. Section 3 introduces the Queue-Reactive Model (QRM), which provides the market environment. Section 4 embeds this problem within a Reinforcement Learning (RL) and Markov Decision Process (MDP) framework, describing how learning-based agents interact with the market. Finally, Section 5 presents and discusses the results of our experiments.

2   Problem Formulation

The optimal execution problem consists in executing a trade of X_0 shares over a fixed time horizon $[0, T]$. We consider here the problem of optimally executing a buy metaorder. When constructing such a strategy, it is essential to account for the immediate transaction impact of trades, their temporary price effects, and the potential long-term consequences arising from persistent market responses. A terminal penalty term may also be introduced to reflect the cost associated with failing to complete the purchase by the end of the horizon.

In a discrete-time setting with $N+1$ decision points, the purchasing strategy is formulated as a sequential decision process, in which the trader determines the quantity Δx_{τ_k} to buy at each time step $\tau_k = kT/N$, for $k \in \{0, \ldots, N\}$, with $\tau_0 = 0$ and $\tau_N = T$. The cumulative purchases form a trajectory $\{x_{\tau_0}, \ldots, x_{\tau_N}\}$, where x_{τ_k} denotes the cumulative number of shares acquired by time τ_k. By construction, $x_0 = 0$, and full completion requires $x_T = X_0$ at the terminal time. Let P_{τ_k} denote the average execution price for the purchase $\Delta x_{\tau_k} = x_{\tau_k} - x_{\tau_{k-1}}$, which depends on both the current and all preceding trades and incorporates all transaction costs. The objective of a risk-neutral trader is to minimize the expected total cost of purchase over the horizon T:

$$
\min_{x \in \mathcal{A}} \mathbb{E} \left[\sum_{k=0}^{N} P_{\tau_k} \Delta x_{\tau_k} \right],
$$

$$
\text{where} \quad \mathcal{A} = \left\{ (x_{\tau_0}, x_{\tau_1}, \ldots, x_{\tau_N}) \in \mathbb{R}_+^{N+1} : \sum_{k=0}^{N} \Delta x_{\tau_k} = X_0 \right\}.
$$

(1)

The cost functional in Equation (1) is directly related to the Implementation Shortfall (IS) measure introduced by [41], which quantifies the deviation between the realized cost of execution and the cost that would have been incurred had all shares been purchased instantaneously at the initial market price P_0. Formally, for a purchasing strategy, the IS is defined as

$$
\text{IS} = \sum_{k=0}^{N} P_{\tau_k} \Delta x_{\tau_k} - X_0 P_0,
$$

where the first term represents the realized cost of execution and the second term corresponds to the paper cost associated with immediate execution at P_0. Minimizing the expected total cost is therefore equivalent to minimizing the expected IS, since the benchmark term $X_0 P_0$ is constant and independent of the trading strategy. Economically, this formulation captures the fundamental trade-off between *market impact* and *timing risk* as in [3]: executing too quickly increases costs through adverse price impact, while executing too slowly exposes the trader to unfavorable price movements. Hence, minimizing the expected IS is equivalent to finding the purchasing trajectory that optimally balances liquidity consumption and exposure to price risk.

3 The Queue-Reactive Model

The QRM, introduced by [29], provides a stochastic representation of the limit order book specifically tailored to large-tick assets. This mechanism enables the model to reproduce the stylized facts in high-frequency markets, including the persistence of order-flow imbalance, asymmetric liquidity profiles, and mean-reverting mid-price dynamics.

3.1 Description of the Market Simulation

At each time t, the state of the LOB is represented by a $2K$-dimensional vector

$$
X(t) = (q_{-K}(t), \ldots, q_{-1}(t), q_1(t), \ldots, q_K(t)),
$$

where K denotes the number of visible price levels on each side of the book. The quantity $q_i(t)$ denotes the standing volume at level Q_i at time t priced at

$$
p_i = p_{\text{ref}} + \frac{i\delta}{2}, \quad \forall i \in \{-K, \ldots, -1, 1, \ldots, K\},
$$

with δ the tick size and p_{ref} an unobservable *reference price* centered within the LOB. By convention, Q_{-i} denotes a bid level and Q_i an ask level. Formally, $X(t)$ evolves as

a continuous-time Markov jump process taking values in $\mathbb{N}^{2K}$ with generator matrix $\mathcal{L}$ specified by

$$\mathcal{L}_{l,l+e_i} = f_i(l), \tag{2}$$

$$\mathcal{L}_{l,l-e_i} = g_i(l), \tag{3}$$

$$\mathcal{L}_{l,l} = -\sum_{p \neq l} \mathcal{L}_{l,p}, \tag{4}$$

$$\mathcal{L}_{l,p} = 0 \quad \text{otherwise}, \tag{5}$$

with $l = (l_{-K}, \ldots, l_{-1}, l_1, \ldots, l_K) \in \mathbb{N}^{2K}$ and e_i denoting the vector i of the canonical basis of $\mathbb{R}^{2K}$. Each queue $q_i(t)$ evolves as a one-dimensional birth–death process governed by the state-dependent intensities $\lambda_i^L(q_i(t))$ for limit order arrivals, $\lambda_i^M(q_i(t))$ for market orders consuming liquidity at level i, and $\lambda_i^C(q_i(t))$ for order cancellations. Note there is no bid-ask distinction as we suppose bid-ask symmetry. The transition rates at time t are

$$f_i(X(t)) = \lambda_i^L(q_i(t)), \quad \text{and} \quad g_i(X(t)) = \lambda_i^M(q_i(t)) + \lambda_i^C(q_i(t)),$$

for all $i \in [-K, \ldots, -1, 1, \ldots, K]$. Conditionally on the current state of the book, order arrivals at each level follow independent Poisson processes. The dependence of intensities on queue sizes induces both auto and cross correlations in the order flow, producing realistic microstructural dynamics.

Remark 3.1. *We retain Model 1 of [29], where the intensities λ_i^L, λ_i^M, and λ_i^C depend solely on the size of the corresponding queue $q_i(t)$. Empirical calibration in [29] shows that this specification captures most of the variation in order-flow intensities, with only minor gains in likelihood from adding neighboring queues or imbalance. Later studies [30] confirm that richer dependencies mainly improve qualitative realism, while the overall quantitative fit remains comparable.*

3.2 Invariant Distribution and Ergodicity

Under mild assumptions, [29] prove that the $2K$-dimensional Markov jump process X is ergodic. This means that there exists a unique invariant probability measure π and that the process converges to it from any start.

Theorem 3.1 (Ergodicity). *Assume that*

(i) there exist $C_{\text{bound}} \in \mathbb{N}$ and $\delta_0 > 0$ such that, for all $i \in [-K, \ldots, -1, 1, \ldots, K]$ and any $p = (p_{-K}, \ldots, p_{-1}, p_1, \ldots, p_K) \in \mathbb{N}^{2K}$ with $p_i > C_{\text{bound}}$, we have

$$f_i(p) - g_i(p) \leq -\delta_0;$$

(ii) there exists $H > 0$ such that $\sum_i f_i(p) \leq H$ for every state $p \in \mathbb{N}^{2K}$.

Then X is non-explosive, irreducible, positive recurrent, and therefore ergodic with a unique invariant probability measure π.

Proof. See Theorem 2.1 in [29]. □

The invariant distribution π_i of the limit Q_i is given by the intensities of the process introduced previously. Define the arrival/departure ratio ρ_i by

$$\rho_i(n) = \frac{\lambda_i^L(n)}{\lambda_i^C(n+1) + \lambda_i^M(n+1)}. \tag{6}$$

The invariant distribution satisfies

$$\pi_i(0) = \left(1 + \sum_{n=1}^{\infty} \prod_{j=1}^{n} \rho_i(j-1)\right)^{-1}, \quad \text{and} \quad \pi_i(n) = \pi_i(0) \prod_{j=1}^{n} \rho_i(j-1). \tag{7}$$

3.3 Price Dynamics

The process X described above characterizes the LOB dynamics for a fixed reference price p_{ref}. To endogenize price movements, two additional parameters are introduced, θ and θ^{reinit}. Whenever the mid-price p_{mid} changes, the reference price p_{ref} moves by one tick in the same direction with probability θ, provided that the corresponding best queue is depleted ($q_{\pm 1}=0$). Following such a reference-price change, the LOB state is either redrawn from its invariant distribution π, centered around the new p_{ref}, with probability θ^{reinit}, or the standing volumes are shifted accordingly with probability $1-\theta^{\mathrm{reinit}}$. Figure 2 illustrates these mechanisms when a market order depletes the best ask volume.
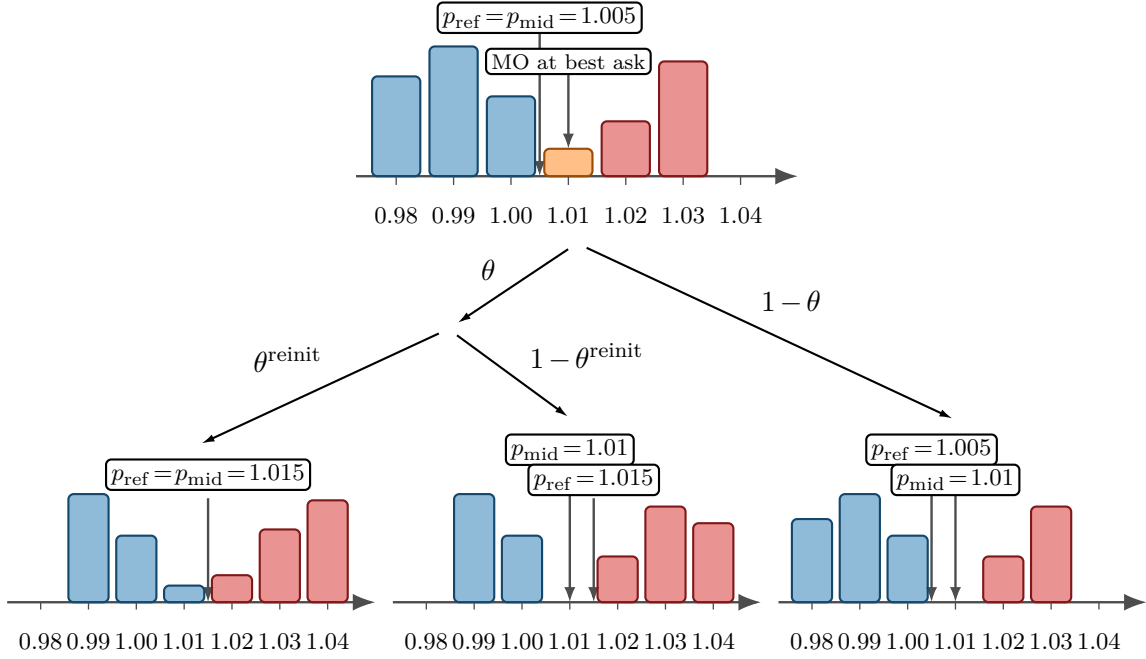

Figure 2: QRM response to consuming the best ask q_1 at time t with tick size $\delta = 0.01$. The root node shows a typical LOB state just before the trade at time t^-, with volumes drawn from the invariant distribution, while the leaf nodes depict the possible post-trade configurations generated by the QRM dynamics at time t.

In [29], the authors of interpret parameter θ^{reinit} as quantifying the proportion of price changes associated with exogenous information shocks. Following such shocks, market participants typically rebalance their order flows around the new reference price almost instantaneously. The empirical calibration of [29] using France Télécom (Euronext Paris) data from January 2010 to March 2012, yields $\theta=0.7$ and $\theta^{\mathrm{reinit}}=0.85$, implying that most price adjustments are accompanied by rapid order-book reconfigurations. Although this value may seem high, since volatility models generally attribute around 80% of price variance to endogenous, self-referential mechanisms (see for example [9]), it can alternatively be understood as the probability that the order book refills following a price movement driven either by market makers or by algorithmic liquidity provision. Henceforth, unless otherwise stated, we adopt the same calibration as [29] and use the same order-flow intensities.

3.4 Market impact

In this Section, we explore the joint role of $(\theta, \theta^{\mathrm{reinit}})$ in shaping liquidity dynamics. To this end, we analyze how the QRM responds when an external trader consumes the entire

best ask volume q_1, as shown in Figure 2. This controlled perturbation shows that different parameter combinations give rise to distinct market-impact regimes, where some configurations lead to prices mean-reverting after the trade, corresponding to a transient impact, while others result in prices continuing to rise on average, corresponding to a permanent impact.

Let t_k^- denote the time just before a buy MO at the best ask that consumes all the best ask volume and p_k the associated mid-price after. The index k runs over all the events, and not only the MOs. Furthermore, we assume $p_{\text{ref}}(t_k^-) = p_{\text{mid}}(t_k^-)$, with the surrounding volumes $q_{\pm i}$ sampled from the invariant distribution, consistent with the ergodicity of the process X under mild conditions. For simplicity, all queues $q_{\pm i}$ are considered non-empty, as this represents the typical configuration of the invariant distribution π. When the LOB is redrawn from π, we assume without loss of generality that all queues are non-empty, as the contribution of empty queues cancels out on average by bid–ask symmetry. The expected mid-price jump after consuming the best ask is

$$\mathbb{E}[\Delta p_k] := \mathbb{E}[p_k - p_{k-1}] = (1 + \theta\,\theta^{\text{reinit}})\frac{\delta}{2}, \tag{8}$$

where δ is the tick size. As expected, $\mathbb{E}[\Delta p_k]$ increases with both θ and θ^{reinit}, consistent with the behavior observed in Figure 3.
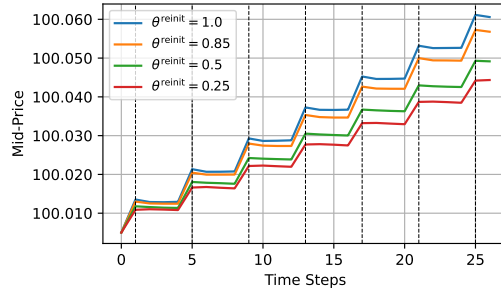

Figure 3: Average mid-price across 20,000 simulations in which a trader systematically buys the entire best ask at fixed time intervals (vertical dashed lines). We set $\theta = 0.7$.

Consider now the next event $k+1$ that updates the LOB. With probability $\theta\theta^{\text{reinit}}$, the book is redrawn from its invariant distribution and the mid-price remains unchanged on average. With probability $\theta(1-\theta^{\text{reinit}})$, a bid refill occurs, increasing the price by $\delta/2$; conversely, with probability $(1-\theta)$, an ask refill occurs, decreasing the price by $\delta/2$ and inducing mean reversion. These last two events provide a first-order approximation, as in the large-tick regime the next events are typically bid or ask refills. Averaging over these outcomes yields

$$\mathbb{E}[\Delta p_{k+1}] = \left[\theta(2 - \theta^{\text{reinit}}) - 1\right]\frac{\delta}{2}, \tag{9}$$

which holds on very short time scales only. The sign of $\theta(2 - \theta^{\text{reinit}}) - 1$ delineates the post-trade regime: a positive value implies that the quote adjustment after the trade is on average in the same direction of the trade, while a negative value indicates a mean reversion of the midprice after the trade.

In order to numerically test this expression, we initialize the LOB by sampling it from the invariant distribution and then we "send" a buy market order that completely depletes the best ask. Here and in the following we set the tick size at $\delta = 0.01$. Figure 4a illustrates the contour plot of the estimated $\mathbb{E}[\Delta p_{k+1}]$ for different values of $\theta, \theta^{\text{reinit}} \in [0.5, 1.0]$. The dashed black line in the figure denotes the theoretical frontier $\mathbb{E}[\Delta p_{k+1}] = 0$ described by Equation (9), aligning closely with the empirical phase boundary (white region).
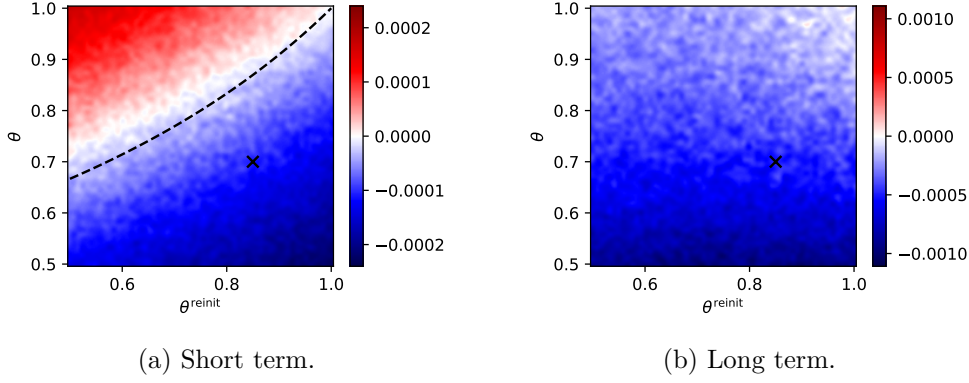
(a) Short term.　　　　　　　　　　(b) Long term.

Figure 4: Heatmaps of expected immediate price change $\mathbb{E}[\Delta p_{k+1}]$ (left) and cumulative impact $\mathbb{E}[p_{k+75} - p_k]$ averaged over 20,000 simulations (right) starting from a stationary LOB state with an exogenous trader consuming the best ask. The black cross marks $(\theta, \theta^{\text{reinit}}) = (0.7, 0.85)$.

We now numerically study the long term behavior of the price after an ask depleting market order. The difficulty of course is to establish, for the different parameter configurations, when the asymptotic value of the price has been reached. We first consider a fixed time interval and estimate $\mathbb{E}[p_{k+75} - p_k]$ as a function of $(\theta, \theta^{\text{reinit}})$ (see Figure 4b). We observe that prices consistently exhibit long-term mean reversion across all parameter configurations. The only exception is the white region in the top-right corner of Figure 4b, which occurs because for large values of both θ and θ^{reinit} the volumes are almost always redrawn from the invariant distribution around the updated reference price. As θ and θ^{reinit} decrease, prices exhibit a stronger mean reversion (blue regions), with the long-term intensity of this effect governed primarily by θ. As shown in Figure 2, there are two scenarios after buying the best ask: either the LOB is redrawn from its invariant distribution, or there is a bid-ask refill. When the LOB is redrawn from the invariant distribution, the average mid-price remains constant due to bid–ask symmetry. Thus, this scenario does not contribute to the observed mid-price changes, which are entirely accounted for by the bid-ask refill scenarios (see Appendix B.2).
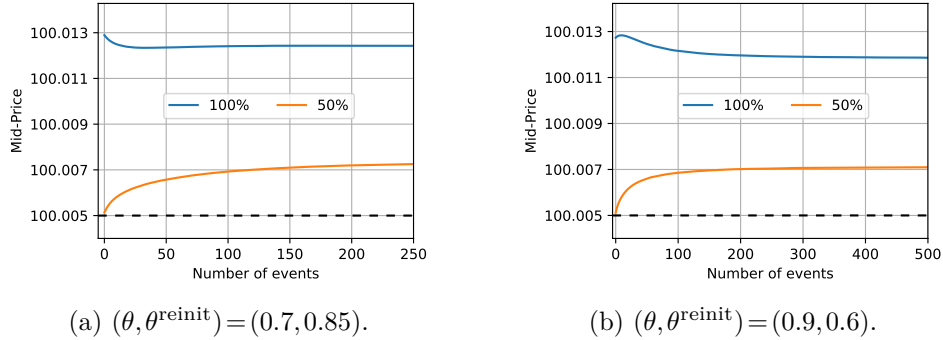

(a) $(\theta, \theta^{\text{reinit}}) = (0.7, 0.85)$.　　　　　(b) $(\theta, \theta^{\text{reinit}}) = (0.9, 0.6)$.

Figure 5: Evolution of $\mathbb{E}[p_k]$ after buying the best ask at $k = 0$, averaged over 1,000,000 simulations. The mid-price before the trade is 100.005 (horizontal dashed line).

To investigate in more detail the price reversion after a trade, we now fix the value of $(\theta, \theta^{\text{reinit}})$ and study the average price dynamics via numerical simulations. As before, we draw the initial state of the LOB from the invariant distribution and then we consider two scenarios: in the first, as above, we send a buy market order of size equal to the best ask, while in the second the size is half of the best ask volume[1]. We make this choice because

---

[1]Since QRM orders have integer size, we set the market order size to the floor of half the ask volume.

the RL algorithm used below will have these options (plus the choice of not trading) in the action space. Figure 5 shows the average price trajectories for two configurations of $(\theta, \theta^{\text{reinit}})$. We observe that in both cases, when the market order depletes the best ask, the price displays a clear mean reversion, which can be very slow as in the right top panel. On the contrary, when the size of the market order is half the volume at the best ask (clearly without mechanically moving the price), the price trends in the same direction of the trade. This difference indicates that the impact model associated with the QRM is inherently nonlinear in the trade size, differently from reduced form standard models used in optimal execution, such as Almgren & Chriss or the Transient Impact Model[2] [8]. This implies that the optimal trading for the QRM should tactically place market orders of a size which depends, among other things, on the state of the LOB and this makes the problem inherently complex justifying the adoption of deep RL to solve it. Moreover, we expect that the price trajectory after a trade depends also on the absolute volume at the ask and not only on the fraction of it taken by the trade[3]. We will add this variable in the state space of the RL algorithm and we will show below that it brings a significant contribution to its performance.

Remark 3.2. *From an implementation perspective, the simulator is designed such that when the trader consumes the full best ask, triggering a mid-price change, the QRM reacts identically to a mid-price change generated endogenously within the model, since it does not differentiate between endogenous and exogenous price movements.*

4 Optimal Execution Setting

4.1 Markov Decision Process Embedding

The execution problem is formulated as a Markov Decision Process (MDP). The MDP is defined as a tuple $\langle \mathcal{S}, \mathcal{A}, \mathbb{P}, \mathcal{R}, \gamma, \mu \rangle$ (see [44]), where $\mathcal{S}$ is the state space, $\mathcal{A}$ the action space, $\mathbb{P}(\cdot|s,a)$ is a Markovian transition model that assigns to each state-action pair (s,a) the probability of reaching the next state s', $\mathcal{R}(s,a)$ is a bounded reward function, $\gamma \in [0,1[$ is the discount factor, and μ is the distribution of the initial state.

State space Each state s_{τ_k} at time τ_k for $k \in \{0, \dots n\}$ is defined as

$$s_{\tau_k} = \big(\tau_k, \text{inventory}_{\tau_k}, \text{best ask price}_{\tau_k}, \text{best bid volume}_{\tau_k}, \text{best ask volume}_{\tau_k} \big).$$

This constitutes a minimal state representation. The agent is informed of the remaining time, its current inventory, and relevant local market conditions in terms of price and liquidity. In particular, the inclusion of both best bid and best ask volumes allows the agent to infer volume imbalance, a well-established short-term predictor of price movements in market microstructure (see [43]). A state is considered terminal either when the time horizon T is reached or when the agent has fully executed its inventory prior to T.

Action space Since the QRM is designed to model large-tick stocks, the action space is constructed so that the agent may consume at most the best ask volume, as trading beyond the first depth level is prohibitively costly. Empirical evidence supports that traders rarely consume more than the first level (see [42]). The action space therefore consists of

---

[2]Interestingly, extensions of the Transient Impact Model with more propagators, such as the one in [47], are able to reproduce the behavior in Figure 5. However the closed form solution for the optimal execution problem is not known in this case.

[3]We cannot test it directly with this type of simulations because it depends on the correlation between consecutive queue sizes, which is zero when sampling from the invariant distribution. In fact, when the best ask has a small volume, it is likely that the second best ask volume is also small, inducing the possibility of a trending price in the direction of the depleting market order.

percentages, where taking an action of, say, 50% at time τ_k corresponds to buying half of the best available ask at time τ_k^-. This formulation enables the RL agent to adjust its trading volume proportionally to prevailing market conditions: executing 50% of the available best when liquidity is high is more profitable than when liquidity is low, while the market impact in both cases remains comparable.

In the QRM, the state variables q_i at each depth i are expressed in units normalized by the *Average Event Size* (AES$_i$), which represents the mean event size across limit, market, and cancel orders observed at level Q_i. Accordingly, the executable quantities Δx_{τ_k} are integer-valued, $x_{\tau_k} \in \mathbb{N}$. In what follows, we focus on a simplified action space,

$$\mathcal{A} = \{0\%, 50\%, 100\%\}, \tag{10}$$

which allows the agent either to remain inactive, consume half the entire best ask or the entire best ask, thereby accelerating convergence during training.

Reward function In this work, our goal is to minimize the expected implementation shortfall under the risk-neutral probability measure. The instantaneous reward at time τ_k is defined as

$$r_{\tau_k} = \Delta x_{\tau_k}(P_0 - P_{\tau_k}) - \alpha \mathbb{1}_{\{\tau_k=T\}}(X_0 - x_T), \tag{11}$$

where P_0 denotes the initial midprice, Δx_{τ_k} the number of shares executed, $\alpha > 0$ a positive constant and P_{τ_k} the execution price, which in our setting is always the best ask. A terminal penalty is applied if the agent fails to fully execute the inventory by the end of the horizon. This penalty is equal to the number of shares remaining to be executed, scaled by a final penalty parameter α, thereby encouraging complete execution of the X_0 shares before time T. For higher risk aversion, the execution horizon T may be shortened to limit price-risk exposure.

4.2   Reinforcement Learning

In this paper, we employ Deep Reinforcement Learning to approximate the optimal execution schedule within the QRM model. Specifically, we adopt the Double Deep Q-Network (DDQN) algorithm [48], a model-free, online, off-policy reinforcement learning approach, which provides a model-agnostic solution to the optimal execution problem.

In this work, we focus on the Double Deep Q-Network (DDQN) algorithm as it provided the best learning results. Algorithm 1 presents the training procedure for a Double Deep Q-Network (DDQN) agent. Two neural networks are initialized: the main network Q_{main} for action selection, and the target network Q_{tgt} for value estimation. At each time step, the agent observes the current state and selects an action via an ϵ-greedy policy that balances exploration and exploitation. The resulting transition and reward are stored in a replay buffer. Once the buffer reaches a sufficient size, mini-batches are sampled to update Q_{main} by minimizing the Bellman loss using target values computed from Q_{tgt}. Every m steps, the target network is synchronized with the main network and the exploration rate ϵ is decayed. This iterative process allows the agent to approximate the optimal action–value function while controlling value overestimation. After training, Q_{main} serves as the agent's policy for decision-making in the environment.

We consider a finite-horizon setting with exponentially discounted future rewards by the factor γ. More specifically, the per-step reward entering the Bellman recursion is

$$r_k := r(s_{\tau_k}, a_{\tau_k}) = a_{\tau_k}(P_0 - P_{\tau_k}) - \alpha \mathbf{1}_{\{\tau_k=T\}}(X_0 - x_T).$$

A trajectory is a sequence of states, actions, and rewards up to a stopping time τ, i.e.,

$$(s_0, a_0, r_1, s_1, a_1, r_2, ..., s_{\tau-1}, a_{\tau-1}, r_\tau).$$

Algorithm 1 DDQN Algorithm in the QRM environment

Require: Initialize Q_{main} (random weights), $Q_{\text{tgt}} \leftarrow Q_{\text{main}}$;

1: Replay memory (size L), $\epsilon = 1$, batch size b, episodes M, $c < 1$;
2: Set QRM parameters and market state S_0, inventory q_0.
3: **for** $i = 1$ to M **do**
4:     Initialize the QRM simulation
5:     **for** $t = 1$ to N **do**
6:         $s_t \leftarrow QRM(t)$
7:         $a_t \leftarrow \begin{cases} \text{random action} & \text{w.p. } \epsilon \\ \arg\max\limits_{a} Q_{\text{main}}(s_t, a | \theta_{\text{main}}) & \text{w.p. } 1 - \epsilon \end{cases}$
8:         Execute a_t in QRM $\Rightarrow$ new state s_t, observe reward r_t
9:         Store (s_t, r_t, a_t, s_{t+1}) in memory
10:        **if** $|\text{Memory}| \geq b$ **then**
11:            Sample $(s_t^j, r_t^j, a_t^j, s_{t+1}^j)_{j=1}^b$
12:            $a^{*,j} = \arg\max\limits_{a} Q_{\text{main}}(s_{t+1}^j, a | \theta_{\text{main}})$
13:            $y^j = r_t^j + \gamma Q_{\text{tgt}}(s_{t+1}^j, a^{*,j} | \theta_{\text{tgt}})$
14:            Update θ_{main} minimizing $\mathcal{L} = \frac{1}{b} \sum_j (y^j - Q_{\text{main}}(s_t^j, a_t^j | \theta_{\text{main}}))^2$
15:        **end if**
16:        **if** $t \bmod m = 0$ **then**
17:            $\epsilon \leftarrow \epsilon - c, \quad \theta_{\text{tgt}} \leftarrow \theta_{\text{main}}$
18:        **end if**
19:     **end for**
20: **end for**

Given a policy π we can define the *State-Action Value Function*

$$Q_\pi(s, a) = \mathbb{E}_\pi[\sum_{i=1}^{\tau} \gamma^{i-1} r_i | s_0 = s, a_0 = a], \tag{12}$$

which represents the expected return from state s if we take action a and then we follow the policy π and can be recursively defined by the following Bellman equation [5],

$$Q_\pi(s, a) = r(s, a) + \gamma \mathbb{E}_{\substack{s' \sim \mathcal{P}(\cdot|s,a) \\ a' \sim \pi(\cdot|s')}} [Q_\pi(s', a')]. \tag{13}$$

Solving the MDP means finding the *optimal* policy π^* which is the policy that maximizes the objective

$$J_\pi := \mathbb{E}_{\substack{\pi \\ s_0 \sim \mu}} \Big[\sum_{i=1}^{\tau} \gamma^{i-1} r_i \Big].$$

5 Experiments and results

In this section we present the results of our numerical investigations. The aim of the experiments is to study whether the DDQN agent is able to find robust optimal execution strategies without any form of knowledge of the underlying impact model. We focus here on the best-performing configuration, while additional experiments exploring alternative state and action spaces are reported in Appendix C for comparison.

5.1 Benchmark Strategies

To evaluate the DQN policy, we introduce a set of benchmark execution strategies designed to minimize Implementation Shortfall (IS) and provide a basis for performance comparison.

Baseline Model   We consider the Time-Weighted Average Price (TWAP) benchmark, in which the trader executes X_0 number of shares uniformly over a fixed horizon $[0, T]$ divided into N discrete intervals. The trading rate is constant, given by $\Delta x^* = \left(\frac{X_0}{N}, \ldots, \frac{X_0}{N} \right)$, or equivalently, the number of executed shares after time step τ_k is equal to $x^*_{\tau_k} = k\frac{X_0}{N}$, for $k = 0, \ldots, N$. For a risk-neutral trader, this policy coincides with the Almgren–Chriss (A&C) solution [3], whose objective is to minimize the expected Implementation Shortfall (IS). Under the standard A&C assumptions of linear permanent and temporary market impact, and asset price dynamics following a Brownian motion with constant volatility, the TWAP is the optimal strategy.

The Percentage of Posted Volume Benchmark   We introduce a family of new benchmarks we call Percentage Of Posted Volume (POPV). More precisely, we define POPV$_i$ as the strategy that purchases a fixed fraction (50% or 100%) of the available volume at the best ask every i time steps, while remaining inactive otherwise. The intuition behind this strategy is to exploit pauses in execution during which prices tend to mean revert (see Figure 3).

5.2   Training Configuration

Algorithm parametrization   In our implementation, we employ fully connected feed-forward neural networks comprising 5 layers, each with 30 hidden units and leaky-ReLU activation functions. We use the ADAM optimizer for optimization. The RL agent is trained on approximately $500,000$ episodes. The epsilon-greedy exploration policy starts at 1.0 and decays linearly to 0.01 over the first 3% of training. We set the final penalty $\alpha = 1.0$. The parameters used to calibrate the algorithm are reported in Table 1. The parameters not shown in the table change depending on the experiments and are reported below accordingly.

DDQN Parameters		Model Parameters	
NN layers	5	Time horizon (T)	600 s
Hidden nodes	30	Time intervals (N)	25
ADAM lr	1e-4	Shares to execute (X_0)	25
Batch size (b)	1,024	Final Penalty (α)	1.0
Replay memory (L)	1e6	θ	0.7
Target update (m)	1e3	θ^{reinit}	0.85
Training episodes (M)	5e5		
Test episodes (B)	2e4		
Discount factor (γ)	0.995		

Note. The target update m is in number of environment steps (not episodes).

Table 1: Fixed parameters used in the DDQN algorithm.

Feature Normalization   As the learning model relies on neural networks, all input features are normalized. Time and inventory are linearly rescaled to the interval $[-1, 1]$, while prices and volumes are standardized using z-score normalization.

5.3   Simulation under Market-Calibrated Dynamics

Environment Setup.   In this section, we adopt the calibration proposed in [29], with parameters $\theta = 0.7$ and $\theta^{\text{reinit}} = 0.85$, and use identical order-flow intensities. These parameters were calibrated on France Télécom (Euronext Paris) using data from January 2010 to March 2012, where the average bid–ask spread was approximately 1.43 ticks. The agent wants to

buy 25 shares and may remain inactive or purchase half or the entire best ask volume at each decision point, i.e., $\mathcal{A} = \{0\%, 50\%, 100\%\}$. We simulate 600 seconds of the QRM with a trader time step of 25 seconds, meaning the agent can act at $\tau_0 = 0$ and subsequently every 25 seconds. Each of these 25 second interval is called *Trader Step*. This combination of parameters is designed to balance execution urgency with tactical flexibility. The number of shares to execute is large enough to require multiple market interventions, yet not so large relative to the time horizon that the optimal policy collapses into systematic buying at every step. This setup allows us to assess whether the RL agent can learn to wait for favorable market conditions and adapt its trading behavior accordingly. To compare the description in events given above with the one in seconds used here, it is useful to remark that there are on average 7 events per second .

We benchmark the RL agent against the strategies introduced in Section 5.1: TWAP, POPV1, POPV2, POPV3 and POPV4. For a fair comparison, POPV1 and POPV2 take action 50% while POPV3 and POPV4 take action 100%. Executing 50% of the posted volume results in a higher Implementation Shortfall (IS) due to reduced market impact compared to the more aggressive 100% setting. Moreover, it is necessary to ensure that the entire inventory is executed within the specified time horizon. This constraint justifies using only POPV1 and POPV2 for the 50% action, as the slower execution of POPV3 and POPV4 would prevent full completion. Conversely, in the 100% action, POPV1 and POPV2 become overly aggressive and yield worse IS, motivating the focus on POPV3 and POPV4. To assess statistical significance, we performed a one-sided Welch's t-test to determine whether the best-performing strategy has a significantly higher average IS than the second-best. Statistical significance is indicated by asterisks: (*) for $p < 0.05$, (**) for $p < 0.01$, and (***) for $p < 0.001$. To ensure comparability across methods, when the agent fails to fully execute its inventory, a final trade is executed at an additional time step and its cost is included in the IS.

Learning Dynamics and Q-Value Analysis. We consider the 5-dimensional state space that has been introduced in Section 4.1. The learning curve in Figure 6 shows steady reward improvement with convergence around 50k episodes, while the TD loss decreases smoothly and stabilizes at low values[4]. This indicates that the DDQN agent learned a stable and well-optimized policy throughout the training.

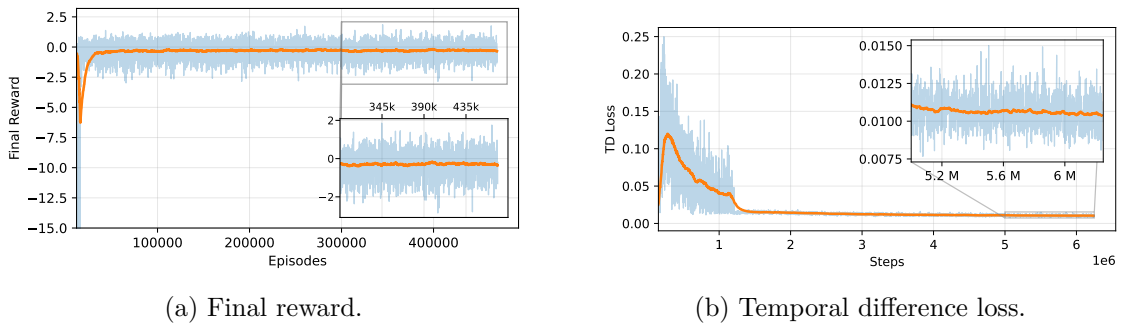(a) Final reward. (b) Temporal difference loss.

Figure 6: Learning across 6,240,000 environment steps ($\approx 500,000$ episodes).

To assess the quality of the learning, we analyze the Q-values for all actions in Figures

---

[4]Note that the reward initially decreases. Under our chosen parameter configuration (in particular, the episode length and the magnitude of the admissible actions), a nearly random policy in the early ϵ-greedy phase trades aggressively enough to execute most of the inventory before the time horizon, so the initial reward is relatively high. As ϵ decreases and the policy becomes more structured, the agent temporarily learns to trade more cautiously, leaving a non-negligible inventory at maturity and thus suffering a larger terminal penalty, which explains the drop in reward. After this transient phase, the agent adjusts its strategy, and the reward increases and eventually converges.

7a, 7b, and 7c. More precisely, we plot the Q-values of the trained RL agent as a function of inventory and time, when the ask price is equal to the arrival price and the bid and ask volumes are equal to their average values.
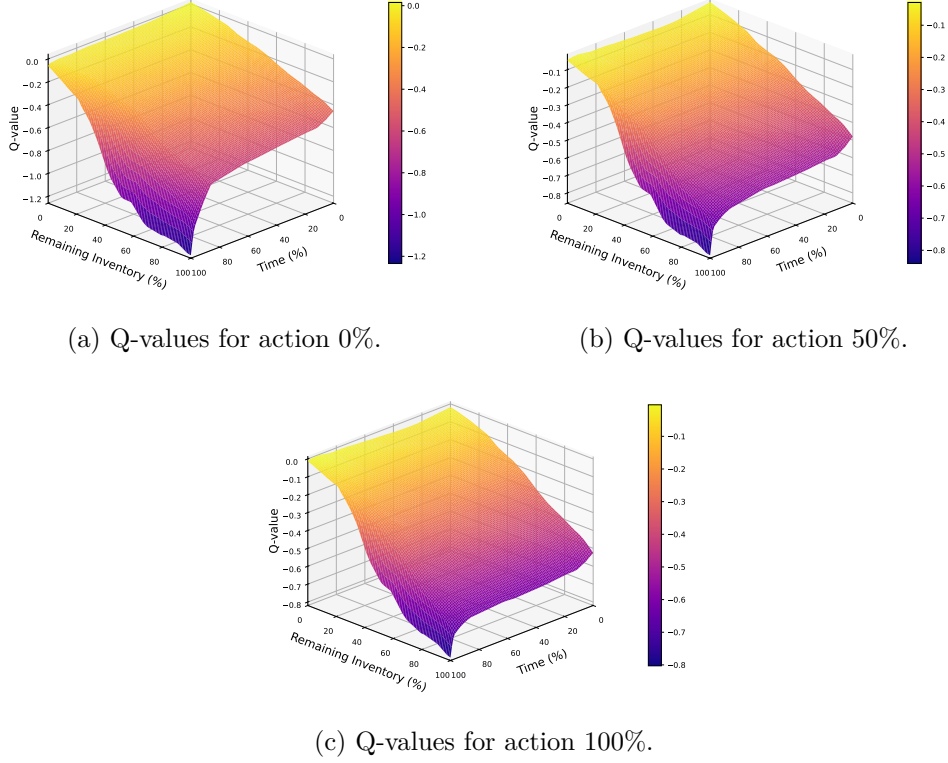

(a) Q-values for action 0%.

(b) Q-values for action 50%.

(c) Q-values for action 100%.

Figure 7: Q-value surfaces at $P = P_0$ with mean bid and ask volumes.

All plots reveal that, for a fixed inventory, Q-values decline over time as the remaining horizon shortens, reflecting the agent's increasing risk of incomplete execution, which is penalized in the reward function. The lowest Q-values are observed at the terminal time T, representing a lower bound on the execution cost. Furthermore, execution costs tend to rise with larger inventories, consistent with the greater market impact and urgency associated with executing larger positions. Moreover, execution costs tend to rise with larger inventories.

Performance Evaluation Against Baselines. Table 2 reports the performance of the DDQN agent compared to benchmark strategies. Since the reward function is defined as the negative of the IS, minimizing IS is equivalent to maximizing the reward.

	POPV1	POPV2	POPV3	POPV4	TWAP	DDQN
Mean	-0.343	-0.342	-0.400	-0.399	-0.365	$-\mathbf{0.259}$***
Std	**0.378**	0.472	0.388	0.437	0.652	0.631

Table 2: Reward results on 20,000 test episodes.

It is clear that the RL agent achieves the best overall performance, with a significantly higher average reward than all benchmarks. As shown in Figure 8, it matches TWAP's best-case performance while maintaining limited worst-case losses.

Figure 8: Reward distribution for the different tested strategies.

We checked how often the different strategies are unable to complete the purchase within the time window. We find that the DDQN agent fails to fully complete the purchase in only 0.045% of episodes, with an average of 1.44 shares remaining. In comparison, POPV2 fails in 0.035% of episodes, leaving an average of 2.14 shares and POPV4 fails in 0.265% of episodes, leaving an average of 2.51 shares. Thus, we can safely conclude that the considered strategies almost always complete the trading program.

Trading Patterns and Tactical Adaptation. Optimal execution problems are typically addressed using a two-layer framework consisting of strategy and tactic. The strategy component determines the overall trading schedule, namely the number of shares to execute within each time interval (see Figure 9). The tactic component, on the other hand, governs how these scheduled orders are executed within each interval (see Figures 11). The following figures report results averaged over 20,000 test episodes.

Figure 9: Average execution trajectory of the different tested strategies as a function of time (measured in trader step).

Figure 9 displays the average execution trajectory of the RL agent compared with those of TWAP and POPV. The DDQN trajectory remains approximately linear for most of the trading horizon, becoming slightly concave only near completion. This pattern contrasts with the more uniform or front-loaded behavior of the rule-based benchmarks and highlights the dynamic adjustment of execution pace observed in the DDQN runs.

The figure, however, should be properly interpreted as it might convey the wrong impression that the DDQN strategy is static (as the TWAP or, more generally, the AC solution). First of all we show that the time needed to complete the execution with the DDQN strategy is highly variable. Figure 10 shows the distribution of the metaorder execution time, i.e. the number of trader steps required to complete the execution under the DDQN policy. Short episodes correspond to periods of abundant liquidity at the best ask, whereas longer episodes occur when the agent waits for more favorable trading opportunities. This broad distribution contrasts with the sharply peaked episode length profiles of benchmark

15

strategies (see Figure 15), underscoring the adaptive nature of the learned policy. Finally, the decline in density a few steps before the time horizon indicates that the RL agent has correctly internalized the terminal penalty, completing execution by the time horizon.
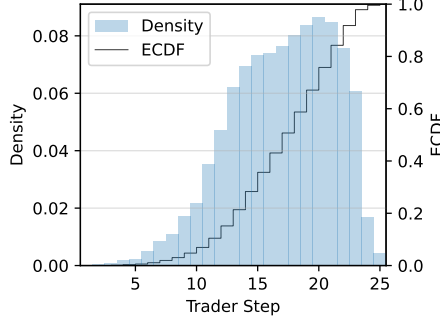

Figure 10: Distribution of the metaorder execution time in number of trader steps.

Second, even restricting to episodes with the same metaorder execution length, we observe a large variability in the strategy. To show this, we compute for each metaorder execution time the average and variance of the gaps between consecutive executions, as shown in Figure 11 (gaps are measured in time steps: a gap of zero indicates two executions occurred at consecutive time steps).

Figure 11: Boxplots of average gaps (left) and gap variance (right) between consecutive executions per episode length for the DDQN strategy.

The broad interquartile ranges and long whiskers indicate that the RL agent does not trade at uniform intervals: even for episodes of comparable duration, the spacing between trades varies substantially. This variability becomes more pronounced for longer episodes, suggesting that the agent adjusts its trading frequency dynamically over time rather than following a fixed temporal schedule. Such heterogeneity across episodes of equal length implies that execution timing depends strongly on the prevailing market conditions. The resulting behavior is therefore tactical in nature, as the agent adapts its actions in response to short-term liquidity and price dynamics rather than adhering to a predetermined strategic rhythm.

To conclude, the DDQN trajectory exhibits a distinctly nonlinear, state-dependent profile: execution is accelerated under favorable conditions and decelerated as the horizon shortens. This pattern reflects a learned balance between immediacy and market impact, driven by the temporal structure of Q-value updates and declining continuation value near maturity. The resulting behavior demonstrates that the DDQN policy learns both strategic and tactical dimensions of execution, dynamically adapting to market states beyond the capability of fixed benchmark strategies (see [38]).

Feature Importance. To identify the main drivers of the RL agent's decisions, we perform an input-gradient analysis and the results are shown in Table 3 (see also Appendix A).

Feature	Gradient (Action 0%)	Gradient (Action 50%)	Gradient (Action 100%)
Inventory	0.40	0.37	0.41
Ask Price	0.34	0.34	0.35
Time	0.10	0.07	0.06
Ask Volume	0.07	0.07	0.07
Bid Volume	0.06	0.05	0.04

Table 3: Input-gradient for actions 0%, 50% and 100%.

The results indicate that the ask price and inventory are the primary determinants of action selection, suggesting that the agent mainly reacts to immediate market conditions. In contrast, the influence of time is small and decreases with action magnitude. The larger the executed volume, the less significant the time feature, reinforcing that the learned policy is tactical and adaptive rather than schedule-driven. These findings are consistent with a complementary SHAP value analysis[5].

5.4 Robustness to Different Market Conditions

To test robustness, the RL policy trained at $(\theta, \theta^{\text{reinit}}) = (0.7, 0.85)$ is evaluated across QRM simulations with $\theta, \theta^{\text{reinit}} \in [0.5, 1.0]$. Figure 12 shows that it consistently exceeds the best benchmark, achieving up to 27% higher performance. This demonstrates that the learned policy generalizes well to different market regimes and can be relied upon to maintain strong performance under varying conditions.
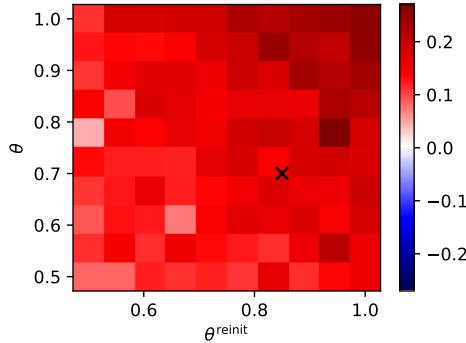

Figure 12: Heatmap of the relative difference between the average reward of the RL agent vs the best benchmark (TWAP in all the cases), averaged over 10,000 simulations.

6 Conclusion

In this work, we proposed a reinforcement-learning framework for optimal execution within the Queue-Reactive Model (QRM). By leveraging the ergodicity and microstructural features of the QRM, including its state-dependent order-flow intensities, its stochastic queue dynamics, and its endogenous liquidity replenishment mechanisms, we produced a simulation environment that captures both direct liquidity consumption and indirect order-flow responses. Training a Double Deep Q-Network (DDQN) in this setting shows that the

---

[5]Results are available upon request.

RL agent learns execution strategies that are tactically adaptive and strategically robust. Across all configurations, the agent consistently outperforms standard benchmarks such as TWAP, adjusting to local liquidity and short-term price pressure rather than following a fixed schedule. Analyses of execution timing, Q-values, and feature importance confirm that the policy internalizes subtle microstructural patterns, including nonlinear impact and volume-imbalance effects.

Possible extensions include adding limit-order placement and queue-management decisions to balance passive and aggressive execution. Extending the methodology to multi-asset execution problems would introduce cross-impact effects and portfolio-level constraints, broadening the scope of the approach. Another potential direction is to integrate predictive alpha signals into the state space so that the agent jointly optimizes execution and directional positioning, thereby connecting optimal trading and optimal execution.

Acknowledgements

References

[1] Valentina Alfi et al. "Minimal agent based model for financial markets II: statistical properties of the linear and multiplicative dynamics". In: *The European Physical Journal B* 67.3 (2009), pp. 399–417.

[2] Robert Almgren. "Optimal trading with stochastic liquidity and volatility". In: *SIAM Journal on Financial Mathematics* 3.1 (2012), pp. 163–181.

[3] Robert Almgren and Neil Chriss. "Optimal execution of portfolio transactions". In: *Journal of Risk* 3 (2001), pp. 5–40.

[4] Robert F Almgren. "Optimal execution with nonlinear impact functions and trading-enhanced risk". In: *Applied mathematical finance* 10.1 (2003), pp. 1–18.

[5] Richard Bellman. "Dynamic programming". In: *Science* 153.3731 (1966), pp. 34–37.

[6] Dimitris Bertsimas and Andrew W Lo. "Optimal control of execution costs". In: *Journal of financial markets* 1.1 (1998), pp. 1–50.

[7] Jean-Philippe Bouchaud, J. Doyne Farmer, and Fabrizio Lillo. "How Markets Slowly Digest Changes in Supply and Demand". In: *Handbook of Financial Markets: Dynamics and Evolution.* Ed. by Thorsten Hens and Klaus Reiner Schenk-Hoppé. Handbooks in Finance. San Diego: North-Holland, 2009, pp. 57–160.

[8] Jean-Philippe Bouchaud et al. "Fluctuations and response in financial markets: the subtle nature ofrandom'price changes". In: *Quantitative finance* 4.2 (2003), p. 176.

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

[10] David Byrd, Maria Hybinette, and Tucker Hybinette Balch. "ABIDES: Towards high-fidelity multi-agent market simulation". In: *Proceedings of the 2020 ACM SIGSIM Conference on Principles of Advanced Discrete Simulation.* 2020, pp. 11–22.

[11] Álvaro Cartea and Sebastian Jaimungal. "Incorporating order-flow into optimal execution". In: *Mathematics and Financial Economics* 10.3 (2016), pp. 339–364.

[12] Álvaro Cartea and Sebastian Jaimungal. "Optimal execution with limit and market orders". In: *Quantitative Finance* 15.8 (2015), pp. 1279–1291.

[13] Álvaro Cartea and Leandro Sánchez-Betancourt. "Optimal execution with stochastic delay". In: *Finance and Stochastics* 27.1 (2023), pp. 1–47.

[14] Anirban Chakraborti et al. "Econophysics review: II. Agent-based models". In: *Quantitative Finance* 11.7 (2011), pp. 1013–1041.

[15] Patrick Cheridito and Tardu Sepin. "Optimal trade execution under stochastic volatility and liquidity". In: *Applied Mathematical Finance* 21.4 (2014), pp. 342–362.

[16] Patrick Cheridito and Moritz Weiss. "Reinforcement Learning for Trade Execution with Market Impact". In: *arXiv preprint arXiv:2507.06345* (2025).

[17] Etienne Chevalier, Yadh Hafsi, and Vathana Ly Vath. "Optimal execution under incomplete information". In: *arXiv preprint arXiv:2411.04616* (2024).

[18] Etienne Chevalier et al. "Optimal Execution under Liquidity Uncertainty". In: *arXiv preprint arXiv:2506.11813* (2025).

[19] Andrea Coletta et al. "Learning to simulate realistic limit order book markets from data as a world agent". In: *Proceedings of the third acm international conference on ai in finance.* 2022, pp. 428–436.

[20] R. Cont et al. "Limit Order Book Simulation with Generative Adversarial Networks". In: *Social Science Research Network* (2023). DOI: 10.2139/ssrn.4512356.

[21] Gianbiagio Curato, Jim Gatheral, and Fabrizio Lillo. "Optimal execution with nonlinear transient market impact". In: *Quantitative Finance* 17.1 (2017), pp. 41–54.

[22] Marina Di Giacinto, Claudio Tebaldi, and Tai-Ho Wang. "Optimal order execution under price impact: a hybrid model". In: *Annals of Operations Research* 336.1 (2024), pp. 605–636.

[23] Jim Gatheral and Alexander Schied. "Optimal trade execution under geometric Brownian motion in the Almgren and Chriss framework". In: *International Journal of Theoretical and Applied Finance* 14.03 (2011), pp. 353–368.

[24] Gatheral, J., Schied, A., and Slynko, A. "Transient linear price impact and Fredholm integral equations". In: *Mathematical Finance: An International Journal of Mathematics, Statistics and Financial Economics* 22.3 (2012), pp. 445–474.

[25] Olivier Guéant and Charles Albert Lehalle. "General intensity shapes in optimal liquidation". In: *Mathematical Finance* 25.3 (2015), pp. 457–495.

[26] Yadh Hafsi and Edoardo Vittori. "Optimal execution with reinforcement learning". In: *arXiv preprint arXiv:2411.06389* (2024).

[27] Lynne Hamill and Nigel Gilbert. *Agent-based modelling in economics.* John Wiley & Sons, 2015.

[28] 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.

[29] 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.

[30] 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.

[31] Gur Huberman and Werner Stanzl. "Optimal liquidity trading". In: *Review of finance* 9.2 (2005), pp. 165–200.

[32]   Junyi Li et al. "Generating realistic stock market order streams". In: *Proceedings of the AAAI Conference on Artificial Intelligence.* Vol. 34. 01. 2020, pp. 727–734.

[33]   Siyu Lin and Peter A Beling. "An end-to-end optimal trade execution framework based on proximal policy optimization". In: *Proceedings of the twenty-ninth international conference on international joint conferences on artificial intelligence.* 2021, pp. 4548–4554.

[34]   Andrea Macrì and Fabrizio Lillo. "Reinforcement learning for optimal execution when liquidity is time-varying". In: *Applied Mathematical Finance* 31.5 (2024), pp. 312–342.

[35]   Volodymyr Mnih et al. "Human-level control through deep reinforcement learning". In: *nature* 518.7540 (2015), pp. 529–533.

[36]   Peer Nagy et al. "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.

[37]   Peer Nagy et al. "LOB-Bench: Benchmarking Generative AI for Finance–an Application to Limit Order Book Data". In: *arXiv preprint arXiv:2502.09172* (2025).

[38]   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.

[39]   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.

[40]   Anna A Obizhaeva and Jiang Wang. "Optimal trading strategy and supply/demand dynamics". In: *Journal of Financial markets* 16.1 (2013), pp. 1–32.

[41]   Andre F Perold. "The implementation shortfall: Paper versus reality". In: *Journal of Portfolio Management* 14.3 (1988), p. 4.

[42]   Fabrizio Pomponio and Frederic Abergel. "Multiple-limit trades: empirical facts and application to lead–lag measures". In: *Quantitative Finance* 13.5 (2013), pp. 783–793.

[43]   Sergio Pulido, Mathieu Rosenbaum, and Emmanouil Sfendourakis. "Understanding the worst-kept secret of high-frequency trading". In: *arXiv preprint arXiv:2307.15599* (2023).

[44]   Martin L Puterman. *Markov decision processes: discrete stochastic dynamic programming.* John Wiley & Sons, 2014.

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

[46]   Richard S Sutton, Andrew G Barto, et al. *Reinforcement learning: An introduction.* Vol. 1. 1. MIT press Cambridge, 1998.

[47]   Damian Eduardo Taranto et al. "Linear models for the impact of order flow on prices. I. History dependent impact models". In: *Quantitative Finance* 18.6 (2018), pp. 903–915.

[48]   Hado Van Hasselt, Arthur Guez, and David Silver. "Deep reinforcement learning with double q-learning". In: *Proceedings of the AAAI conference on artificial intelligence.* Vol. 30. 1. 2016.

A Feature Importance

We draw N transitions (s,a) from the replay buffer and compute the corresponding Q-values $Q(s,a)$. For each input feature s_i, we evaluate the gradient $g_i = \frac{\partial Q(s,a)}{\partial s_i}$, take its absolute value, and average across all N sampled transitions. The resulting per-feature scores quantify how variations in each state variable influence the Q-value. This gradient-based method requires only a single backward pass per state–action pair.

B Additional Plots

B.1 Invariant Distribution

Figure 13: Invariant distribution of $Q_{\pm 1}, Q_{\pm 2}, Q_{\pm 3}$, taken from [29].

B.2 Market Impact

We provide additional details on the bid and ask refill scenarios mentioned in Section 3.4 to clarify the mechanism behind the price mean reversion observed after sending a market order that consumes the entire best ask. In a bid-refill scenario (see Figure 14a), price initially increases because the most probable subsequent event after buying the best ask is a limit order placement at the best bid, which raises the mid-price from 100.010 to 100.015. Logically, the same phenomenon happens when $\theta = 0$ and we observe that the trajectories coincide over short horizons. However, simulations indicate that this new liquidity typically vanishes rapidly, causing the mid-price to return to its previous level. When the reference price then decreases to 100.005 (with probability θ) and volumes are redrawn from the invariant distribution (with probability $\theta\theta^{\text{reinit}}$), the average mid-price settles half a tick below its initial value. This mechanism occurs frequently enough to produce systematic mean reversion, driven by the interaction between the calibrated intensities and $(\theta, \theta^{\text{reinit}})$. In contrast, when $\theta = 0$, the mid-price increases by half a tick, as expected.

(a) Bid refill following a buy order with $(\theta, \theta^{\text{reinit}}) = (0.7, 0.85)$.

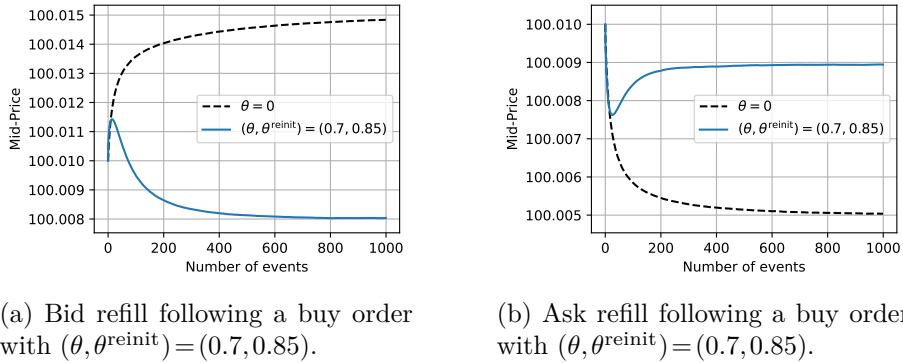(b) Ask refill following a buy order with $(\theta, \theta^{\text{reinit}}) = (0.7, 0.85)$.

Figure 14: Evolution of $\mathbb{E}[p_k]$ after buying the best ask at $k = 0$, averaged over 1,000,000 simulations. The black dotted line denotes the reference case $\theta = 0$. The mid-price before the trade is 100.005.

In the ask-refill scenario (see Figure 14b), the opposite sequence occurs. These two price response patterns are consistently observed across all tested parameter combinations. However, the aggregate results (see Figure 5) vary according to the relative weighting of each of these bid-ask refill scenarios, determined by the probabilities $(\theta, \theta^{\text{reinit}})$.
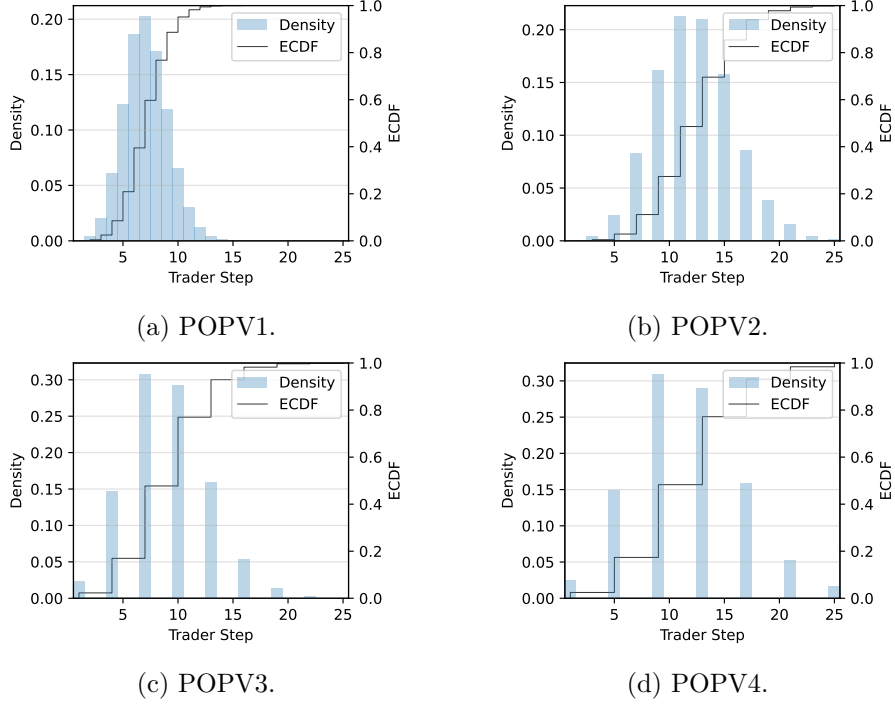
B.3 POPV Benchmarks

(a) POPV1.

(b) POPV2.

(c) POPV3.

(d) POPV4.

Figure 15: Episode length distribution of the POPV benchmarks.

C Additional Experiments with Reduced State Spaces

We evaluate multiple state-space configurations to examine how the dimensionality of the input representation influences the agent's performance. These configurations help isolate the contribution of individual market features such as best ask volume and bid-ask imbalance. The following results correspond to the case of a binary action space, $\mathcal{A} = \{0\%, 100\%\}$. For feature importance, we report only the input-gradient analysis, as similar patterns were observed with the SHAP-value analysis. The following figures report results averaged over 20,000 test episodes.

C.1 Reduced Model: 3D State and Binary Action Space

In this first configuration, we consider a minimal 3-dimensional state space that includes the remaining inventory, the time and the best ask price. We report the results of the trained RL agent and of the benchmarks in Table 4 and show the reward distribution in Fig. 16. The RL agent has the second best average reward after TWAP. We observe that TWAP exhibits a much larger variance in rewards compared to DDQN. This stems from the fact that TWAP is entirely agnostic to price and order book dynamics: it performs poorly when prices trend upward and favorably when they decline, resulting in high variability in realized rewards. Thus, the performance of the RL agent is not satisfactory: we would expect it to perform better than TWAP as it is better informed.

	POPV1	POPV2	POPV3	POPV4	TWAP	DDQN
Mean	-0.413	-0.408	-0.400	-0.399	$-\mathbf{0.365}^{**}$	-0.386
Std	**0.279**	0.342	0.388	0.437	0.652	0.473

Table 4: Reward results. POPV1, POPV2, POPV3, TWAP and DDQN fully execute on all episodes. POPV4 has 2.51 shares remaining in average on 53 episodes (0.265%).

Feature	Gradient (Action 0%)	Gradient (Action 100%)
Inventory	0.45	0.42
Ask Price	0.25	0.26
Time	0.19	0.10

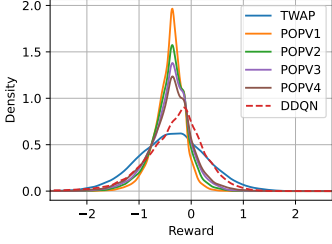Table 5: Input-gradient for actions 0% and 100%.

Figure 16: Reward distribution.

Figure 17: Average inventory trajectory.

Figure 18: Episode length distribution.

C.2 Reduced Model: 4D State and Binary Action Space

In this second configuration, we consider a 4-dimensional state space that includes the remaining inventory, the time, the best ask price and best ask volume. The rationale for including the best ask volume is that it provides the agent with information about the number of shares that would be executed if it decided to consume the entire best ask volume. This feature is particularly relevant because the distribution of best ask volumes is highly right-skewed and exhibits a heavy tail. Thus, the agent can take advantage of unusually large volumes to execute substantial trades in a single action. The results are reported in Table 6: the RL agent outperforms all the benchmarks in terms of average IS. Figure 20 shows that the episodes are longer compared to those of the 3-dimensional state-space agent, indicating that the agent has learned to act more patiently in order to exploit more favorable future prices.

	POPV1	POPV2	POPV3	POPV4	TWAP	DDQN
Mean	-0.413	-0.408	-0.400	-0.399	-0.365	$-\mathbf{0.325}^{***}$
Std	**0.279**	0.342	0.388	0.437	0.652	0.594

Table 6: Reward results. POPV1, POPV2, POPV3 and TWAP fully execute on all episodes. POPV4 has 2.51 shares remaining in average on 53 episodes (0.265%). DDQN has 6.77 shares remaining in average on 31 episodes (0.155%).

Feature	Gradient (Action 0%)	Gradient (Action 100%)
Inventory	0.33	0.31
Ask Price	0.25	0.26
Time	0.12	0.07
Ask Volume	0.06	0.06

Table 7: Input-gradient for actions 0% and 100%.
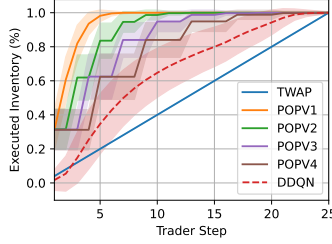

Figure 19: Reward distribution.
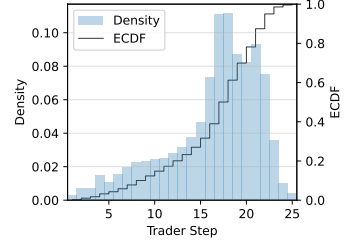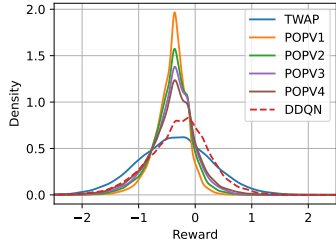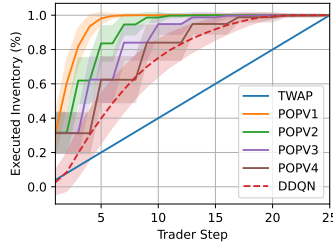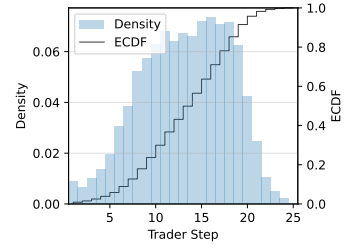
Figure 20: Average inventory trajectory.

Figure 21: Episode length distribution.

C.3 Reduced Model: 5D State and Binary Action Space

In this third configuration, we consider a 5-dimensional state space that includes the remaining inventory, the time, the best ask price and the best bid-ask volume. The rationale behind adding the best bid volume is that it informs the agent of the volume imbalance, a well-established short-term predictor of price movements in market microstructure. The results are reported in Table 8: the RL agent outperforms all the benchmarks.

	POPV1	POPV2	POPV3	POPV4	TWAP	DDQN
Mean	-0.413	-0.408	-0.400	-0.399	-0.365	$-\mathbf{0.290}$***
Std	**0.279**	0.342	0.388	0.437	0.652	0.541

Table 8: Reward results. POPV1, POPV2, POPV3 and TWAP fully execute on all episodes. POPV4 has 2.51 shares remaining in average on 53 episodes (0.265%). DDQN has 2.33 shares remaining in average on 3 episodes (0.015%).

Feature	Gradient (Action 0%)	Gradient (Action 100%)
Inventory	0.44	0.40
Ask Price	0.29	0.30
Time	0.12	0.06
Ask Volume	0.07	0.07
Bid Volume	0.06	0.03

Table 9: Input-gradient for actions 0% and 100%.

Figure 22: Reward distribution.

Figure 23: Average inventory trajectory.
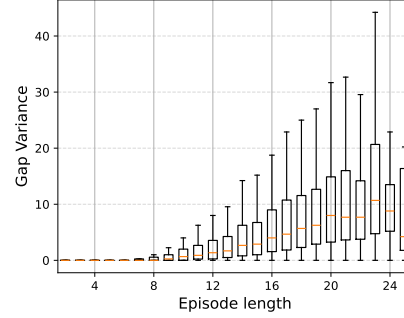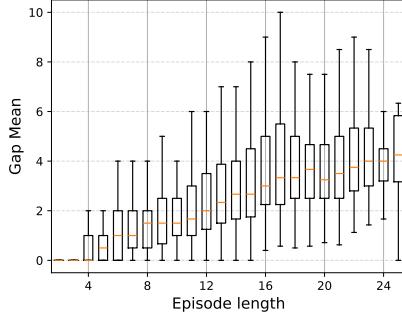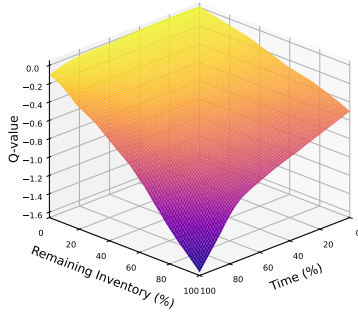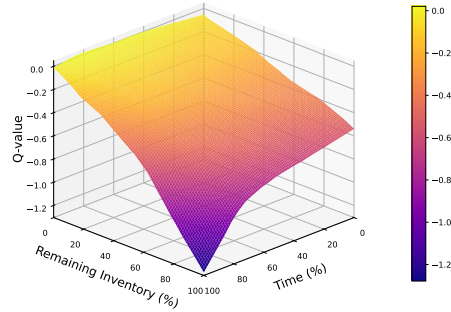
Figure 24: Episode length distribution.

Figure 25: Boxplots of average gaps (left) and gap variance (right) between consecutive executions per episode length.

(a) Q-values for action 0%.
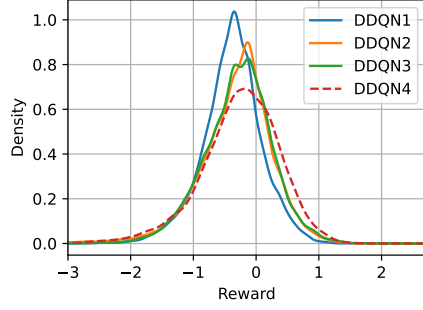
(b) Q-values for action 100%.

Figure 26: Q-value surfaces when the price equals P_0 and the best bid/ask volumes are equal to their means.

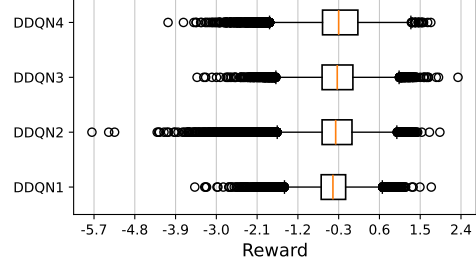## C.4 Comparing Models with Different State and Action Spaces

We summarize and compare the RL performances across all combinations of state and action spaces explored. For the 2-dimensional action space $\mathcal{A} = \{0\%, 100\%\}$, the algorithms DDQN1, DDQN2, and DDQN3 correspond to state spaces of dimensions 3, 4, and 5, respectively. The model denoted DDQN4 refers to the best-performing agent trained with a 3-dimensional action space $\mathcal{A} = \{0\%, 50\%, 100\%\}$ and a 5-dimensional state representation. Our study shows that the RL agent's performance gradually increases by extending the state and action space.

	TWAP	DDQN1	DDQN2	DDQN3	DDQN4
Mean	-0.365	-0.386	-0.325	-0.290	$-\mathbf{0.259}$***
Std	0.652	**0.473**	0.594	0.541	0.631

Table 10: Reward results for different state and action space dimensions.

(a) Histogram.   (b) Boxplot showing the mean.

Figure 27: Reward distribution.