A Controlled Study of Quadratic Risk Penalties in Multi-Agent Reinforcement Learning for Limit-Order-Book Trading

Adam Ru Kun Dong
Informatics Institute, University of Amsterdam
Amsterdam, The Netherlands

Abstract

Risk penalties are widely used in reinforcement-learning trading, but their effects are rarely tested as the main experimental variable. We study two simple quadratic penalties in a shared limit-order-book environment: squared inventory for a market maker and squared remaining quantity for an execution agent. Both agents are trained together with independent proximal policy optimization in JaxMARL-HFT. We set a reference numerical scale for each penalty from validation episodes, then vary the penalties separately and in a joint grid. Each coefficient is represented by one training run, so all findings are conditional on these runs. In the separate market-maker sweep, the run at $0.25\rho^{\star}_{\mathrm{MM}}$ has 48% lower mean squared inventory while participation remains similar. Runs at $0.5\rho^{\star}_{\mathrm{MM}}$ and above have no-trade rates above 95%, indicating withdrawal in this sweep but not a seed-stable collapse point. The strongest execution-penalty run has 59% lower squared remaining quantity, raises completion from 95.3% to 99.9%, and finishes about 20 decision steps earlier. Its paired mean slippage difference is -36.5 tick$\times$share relative to the run with no added penalty. Because the 4,096 evaluation windows come from only 24 test days and no day-clustered interval is available, this difference is descriptive. The joint grid suggests that each penalty mainly affects its own agent. The observed responses are task-dependent, but multi-seed replication and day-level uncertainty are needed before making stronger claims.

Keywords: multi-agent reinforcement learning; limit order books; high-frequency trading; market making; optimal execution; quadratic risk penalties.

1 Introduction

Market making and order execution face different risks. A market maker earns the bid–ask spread but may build a large inventory after uneven fills. An execution trader must finish a parent order while choosing between immediate, costly trades and the risk of waiting. Classical models make these choices explicit. Avellaneda–Stoikov (AS) moves quotes against the dealer's inventory [3]. Almgren–Chriss (AC) balances expected execution cost against the risk of holding an unfinished position [1].

Reinforcement learning (RL) can learn policies from market states without requiring a closed-form model of prices or order arrivals. This flexibility makes reward design important. Inventory penalties are common in RL market making [14], and execution rewards often penalize delay or non-completion [11]. Yet a lower risk measure does not always mean that the task is performed better. A market maker can avoid inventory by not trading. An execution agent can finish early by using expensive, aggressive orders. The issue is harder in a multi-agent market because the liquidity provider and liquidity taker affect each other [7, 2].

We study this question in JaxMARL-HFT [10], a GPU-accelerated environment built on JAX-LOB [5] and JaxMARL [12]. One market maker and one execution agent learn together with independent PPO (IPPO) [4]. The paper makes three contributions. First, it applies the same quadratic-penalty idea to two trading tasks and sets a separate numerical scale for each. Second, it varies the two coefficients alone and together, with checks against fixed counterparties. Third, it examines how the agents reduce risk, rather than reporting risk measures alone. In the available runs, the execution penalty increases urgency, while larger market-maker penalties are associated with very low activity. Because there is one training run per coefficient, the study does not identify a seed-stable transition point.

2 Background

AS is a useful reference for active inventory control. Its reservation price moves below the mid-price when inventory is positive and above it when inventory is negative [3]. This makes one side of the quote more likely to reduce the current position. RL market makers can learn a similar response, but they may also discover that the safest action is not to quote. Prior work has shown that market-making results depend strongly on the observation space, action space, and reward [14, 8, 6].

Execution has a related but different trade-off. Passive orders may save the spread but can leave too much quantity for the end of the episode. Aggressive orders improve completion but may increase cost. RL execution studies therefore combine public book information with private variables such as time and remaining quantity [11, 9]. AC gives a simple classical pattern: greater risk aversion produces a more front-loaded schedule [1]. A learned policy has an extra choice because it can also change where it places each order.

Multi-agent studies show that liquidity providers and takers adapt to each other [7, 2, 13]. JaxMARL-HFT makes this type of experiment fast enough to run on detailed order-book data. Its first experiments also show that learned market makers may trade very little, especially when inventory is penalized [10]. We build on that observation by treating the penalty coefficients as the main experimental variables and by measuring both risk and activity.

3 Method

3.1 Environment and agents

The environment replays historical market-by-order messages. Agent orders are inserted before price-time-priority matching. The market maker observes the current spread and its own

inventory. It chooses among nine two-sided quoting actions and one hold/flatten action. The execution agent observes the book, task progress, and time. It can choose no trade or place 10, 20, or 50 shares at one of four price levels, from passive to far-touch. Each episode lasts 64 decision steps. The execution task is to buy or sell 600 shares.

The agents use separate recurrent actor–critic networks and separate rewards. They still interact through the same book, so each policy changes the environment faced by the other.

3.2 Quadratic risk penalties

Only the two risk terms change across training runs. Let Q_t be the market maker's inventory after step t, and let X_t be the execution agent's remaining quantity. The rewards are

$$r_t^{\mathrm{MM}} = r_{t,\mathrm{base}}^{\mathrm{MM}} - \rho_{\mathrm{MM}} Q_t^2, \qquad r_t^{\mathrm{EX}} = r_{t,\mathrm{base}}^{\mathrm{EX}} - \rho_{\mathrm{EX}} X_t^2. \quad (1)$$

The episode sums $\sum_t Q_t^2$ and $\sum_t X_t^2$ are discrete-time analogues of integrated squared inventory or exposure. This links Eq. (1) to the quadratic inventory-risk term behind the AS certainty equivalent and to the AC variance term, which is proportional to the time integral of squared remaining holdings [3, 1]. The learned rewards keep the same risky state variables and convex form, but they are not discrete solutions of the full classical control problems.

The market-maker base reward includes spread capture, a maker rebate, and damped inventory P&L. The execution base reward compares fills with market trades and the arrival mid-price. The existing terminal treatment of unfinished quantity remains part of the base execution task. It is not added again in Eq. (1). For this reason, $\rho = 0$ means *no additional quadratic penalty*; it does not mean that the full task is economically risk-neutral.

The raw size of ρ depends on the units of the reward and the position variable. We therefore estimate a reference value from a frozen policy with no added penalty:

$$\rho_a^\star = \frac{\mathbb{E}_e\left[\sum_t |r_{e,t,\mathrm{base}}^a|\right]}{\mathbb{E}_e\left[\sum_t v_{e,t}^2\right] + \varepsilon}, \qquad v_t = Q_t \text{ or } X_t. \quad (2)$$

At $\rho_a^\star$, the added penalty would have a similar total numerical scale to the base reward on the validation episodes. This normalization is only a scaling device. It is not a calibration of economic risk aversion, and $\rho_a^\star$ is not directly comparable with the AS parameter γ or the AC parameter λ.

3.3 Classical references

AS and AC are used to explain behavior, not as directly comparable competitors. AS shows how inventory can be reduced while the agent keeps quoting. AC shows how stronger execution risk aversion moves trades earlier. The learned and classical policies use different information, actions, replay rules, and, for execution, different cost definitions. We therefore compare patterns within each policy family rather than absolute performance levels.

4 Experimental Setup

All experiments use AMZN LOBSTER data split by date into 74 training days, 25 validation days, and 24 held-out test days. Each step replays 100 historical messages. Policies are trained from scratch for up to 5×10^8 environment steps across 4,096 parallel environments. Table 1 summarizes the design, and Appendix A gives the exact dates, hyperparameters, seeds, fees, and action definitions.

Table 1: Experimental design. Each coefficient value is represented by one separately trained policy pair.

Component	Setting
MM sweep	$\rho_{\mathrm{MM}}/\rho_{\mathrm{MM}}^\star \in \{0, .1, .25, .5, .75, 1, 2, 4\}$
EX sweep	$\rho_{\mathrm{EX}}/\rho_{\mathrm{EX}}^\star \in \{0, .25, .5, .75, 1, 2, 4\}$
Joint grid	$\{0, .25, .5, 1\} \times \{0, .25, .5, 1\}$
Main evaluation	Co-trained learned–learned pair
Checks	Learned policy vs. fixed counterparty
Test rollouts	4,096 windows from 24 test days
Learned uncertainty	Descriptive; no day-clustered interval
Training seeds	42 for sweeps; 44 for grid

Each market-making episode starts with cash $C_0 = 0$ and inventory $Q_0 = 0$. Its marked-to-market value is

$$PV_t = C_t + Q_t m_t, \quad (3)$$

where m_t is the mid-price. Final portfolio value is PV_T at step 64 and is reported in the simulator's portfolio-value units. Under the tick-price convention used in these runs, one unit is one tick×share; for AMZN, one tick is \$0.01. The no-trade rate is the fraction of decision steps on which the market maker submits no order. Market share is the market maker's filled volume divided by the total traded volume recorded in the episode. Mean squared inventory is $T^{-1}\sum_t Q_t^2$.

For execution, let m_0 be the arrival mid-price, and let p_i and q_i be the price in ticks and quantity in shares of fill i. The direction-normalized slippage proxy is

$$S = d \sum_i q_i (p_i - m_0), \qquad d = \begin{cases} +1, & \text{sell task,} \\ -1, & \text{buy task.} \end{cases} \quad (4)$$

The sum includes the fictional terminal fill for any quantity left at the end of the episode. It is measured in tick×share. Positive values mean execution better than the arrival mid-price; more negative values mean costlier execution. Mean squared remaining quantity is $T^{-1}\sum_t X_t^2$, measured before the fictional terminal fill.

The main evaluation places the two policies from the same training run together. We also use two fixed-counterparty checks. Each learned market maker is tested against the same fixed TWAP-style execution policy, and each learned execution policy is tested against the same fixed AS-style market maker. These checks show whether a pattern remains when the opponent at test time is held constant. They do not undo the interaction that took place during training.

The 4,096 sampled windows ultimately come from only 24 test days. Windows from the same day may share market conditions, so treating all episodes as independent can make an episode-level bootstrap too precise. The saved aggregate outputs used for this short paper do not contain an episode-to-day mapping, so a paired day-level or hierarchical bootstrap cannot be reconstructed without rerunning evaluation. We therefore remove the learned-policy and AC episode-bootstrap intervals and treat their comparisons as descriptive for the available frozen runs. The AS comparison remains aggregated by day and paired across the 24 test days. None of these summaries measures variation across training seeds.

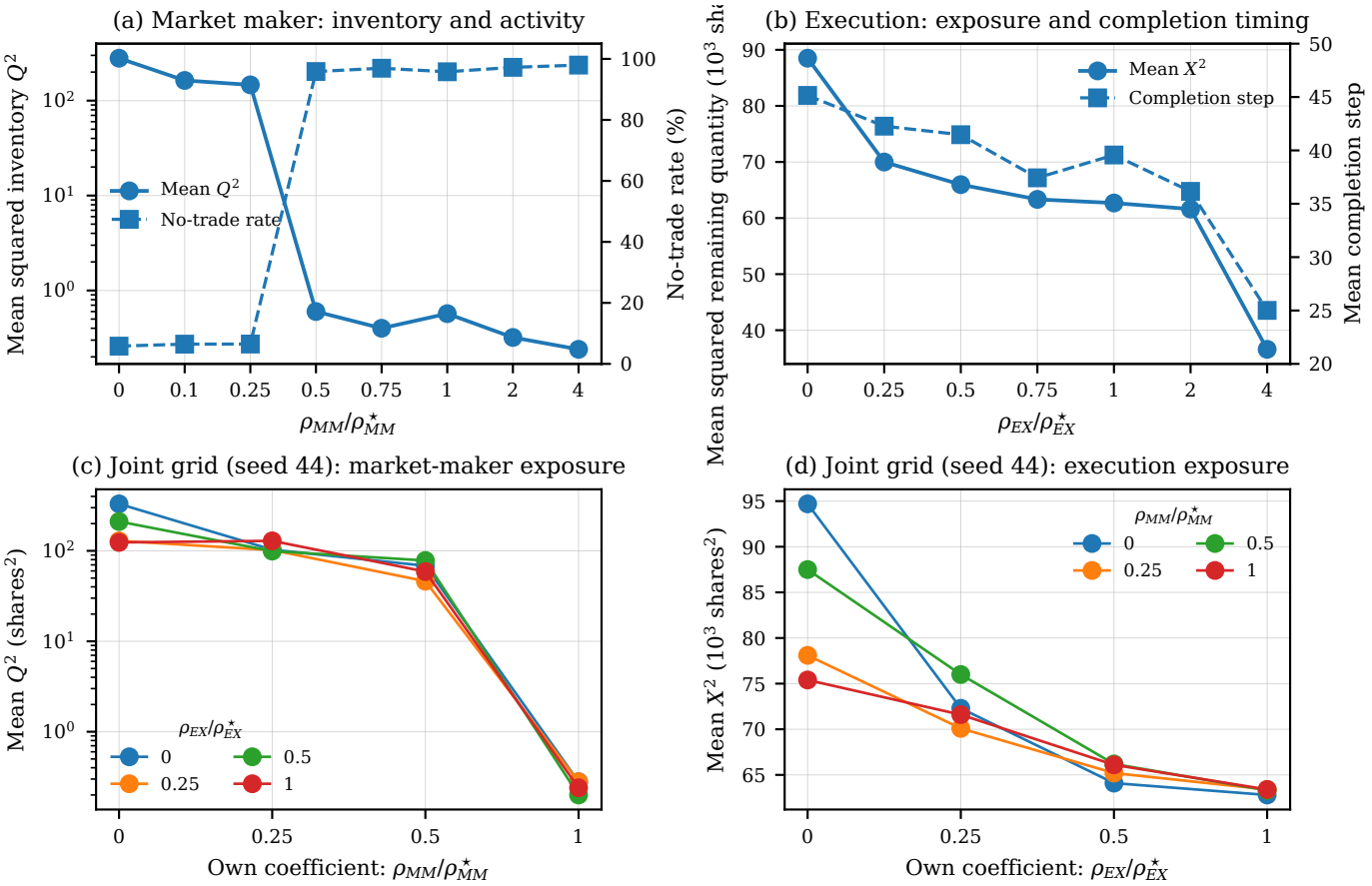


Figure 1: Descriptive means on held-out data. Each sweep point and each grid cell comes from one training run. (a) In the separate market-maker sweep, the runs at $0.5\rho^\star_{\mathrm{MM}}$ and above have very low activity; this does not locate a seed-stable transition. (b) Larger execution penalties are associated with lower remaining quantity and earlier completion in the available runs. (c–d) Within the seed-44 joint grid, each risk measure responds mainly to its own coefficient.

5 Results

Figure 1 summarizes the separate sweeps and the joint grid.

5.1 Market maker: control first, low activity later

In the available separate-sweep runs, moving from 0 to $0.25\rho^\star_{\mathrm{MM}}$ lowers mean squared inventory from 279.9 to 146.6 shares^2, a reduction of 48%. Mean maximum drawdown falls from 389.1 to 281.4 portfolio-value units. The no-trade rate stays near 6%, and market share remains above 7.6%. Mean final portfolio value is 17.5 portfolio-value units higher on the matched test windows. Quote skew also moves against inventory: long positions push quotes down, and short positions push them up. This is the same direction as the AS rule. These differences are conditional on the two single training runs being compared.

The observed behavior changes sharply between $0.25\rho^\star_{\mathrm{MM}}$ and $0.5\rho^\star_{\mathrm{MM}}$ in this sweep. At $0.5\rho^\star_{\mathrm{MM}}$, mean squared inventory is 0.60 shares^2, but the no-trade rate is 95.9% and market share is 0.18%. Larger-coefficient runs stay in this low-activity range. Their low drawdown and near-zero inventory mainly reflect limited liquidity provision, not improved active market making.

The same split appears when every learned market maker faces the same fixed execution policy, so it is not caused only by a different opponent during evaluation. The joint grid, trained with seed 44 rather than 42, places the change elsewhere: $0.5\rho^\star_{\mathrm{MM}}$ remains partly active, while $\rho^\star_{\mathrm{MM}}$ has about 98% no-trade. The present runs therefore suggest a withdrawal response at sufficiently large penalties, but they do not identify its exact location or show that it is stable across training seeds.

5.2 Execution: earlier completion, descriptive cost change

In the available execution-sweep runs, moving from 0 to $4\rho^\star_{\mathrm{EX}}$ lowers mean squared remaining quantity from 88,505 to 36,593 shares^2, a reduction of 59%. Terminal unfinished quantity falls from 4.5 to 0.3 shares. Completion rises from 95.3% to 99.9%, and the mean completion step moves from 45.2 to 25.0.

The policy becomes more urgent rather than less active. Passive placements fall from 51% to 26%. Mid-price and far-touch placements together rise from 12% to 50%, while the no-trade rate stays below 5%. The paired mean slippage difference between the $4\rho^\star_{\mathrm{EX}}$ and zero-penalty runs is -36.5 tick$\times$share. Under Eq. (4), this means that the strongest-penalty run obtains execution that is, on average, 36.5 tick$\times$share less favorable across a 600-share episode. It does *not* mean 36.5 ticks per share: it is about 0.061 tick per parent-order share, or \$0.365 in aggregate at a \$0.01 tick. Without a day-clustered interval, this point estimate does not establish a systematic increase or decrease in cost. The fixed-counterparty check gives closely similar risk and timing means.

The observed timing pattern agrees with AC: stronger execution risk aversion moves the schedule earlier. The learned policy is already somewhat front-loaded when $\rho_{\mathrm{EX}} = 0$ because unfinished quantity is costly in the base task. Its response

is small through the middle of the sweep and becomes large again at $4\rho^\star_{\mathrm{EX}}$. This is a comparison of behavior only, since the AC and learned policies use different replay and slippage conventions.

5.3 Joint grid

Within the seed-44 joint grid, the two penalties mostly act on their own risk variables. In every ρ_{EX} column, larger ρ_{MM} is associated with lower market-maker inventory. In every ρ_{MM} row, larger ρ_{EX} is associated with lower execution exposure. Across the $\rho^\star_{\mathrm{MM}}$ row, mean market-maker Q^2 stays below 0.3 shares^2. Across the $\rho^\star_{\mathrm{EX}}$ column, execution X^2 changes by only about 1%. The other agent's penalty matters most near the no-penalty corner, and these smaller cross-effects are not monotone. The grid is a separate experiment, not a multi-seed replication of every sweep point.

5.4 Comparison with AS and AC

The classical policies show what risk reduction looks like when the task remains active. At its validation-selected setting, AS lowers mean squared inventory by 22% relative to symmetric quoting, from 61.4 to 47.7 shares^2, while quoting at every decision step. The paired day-level portfolio-value difference is $+0.001$ USD per episode, with a 95% interval of $[-0.007, 0.009]$. The learned market maker follows the same inventory-skewing direction only while it remains active. Its lower inventory at larger coefficients comes from a behavior that AS does not allow: leaving the market.

For execution, the most front-loaded AC schedule finishes at step 24 rather than step 60 under TWAP. Its normalized squared remaining quantity falls from 30.5% to 5.4%, and the standard deviation of cost falls from 12.15 to 5.21 ticks per share. Mean cost rises by 0.22 ticks per share relative to TWAP. We report this as a descriptive difference because the original AC interval was based on episode windows rather than day clusters. The learned execution agent also finishes earlier as its penalty rises. These numbers should not be used for a direct ranking because the two policy families use different engines and cost definitions.

6 Discussion

In the observed policies, the same quadratic form leads to different behavior because the tasks have different ways to avoid risk. The execution agent cannot solve its problem by doing nothing: unfinished quantity remains costly. Larger ρ_{EX} is therefore associated with earlier and more aggressive execution in these runs. The market maker has no similar requirement to keep quoting. In the low-activity runs, not trading becomes the easiest way to avoid inventory.

This result also shows why risk measures must be read together with checks that the task is still being performed. For market making, inventory and drawdown should be reported with no-trade rate, market share, and action frequencies. For execution, remaining quantity and completion should be reported with slippage and order aggressiveness. Scaling the coefficient makes the sweep easier to interpret, but it does not prevent an undesirable solution and should not be read as an economic calibration.

A better market-making objective may need separate terms or constraints for risk and participation. Examples include minimum quoting requirements, a market-share constraint, or constrained RL with an inventory-risk budget. A simple activity bonus is not necessarily enough, because it could reward unprofitable volume. The evaluation should continue to measure both inventory control and useful liquidity provision.

The execution result needs a different caution. The point estimate for the slippage change is modest relative to the size of the execution task, but no valid day-clustered interval is available. The experiment therefore does not show that extra urgency is cost-free, nor does it establish a systematic cost increase. A claim of practical equivalence would also require a pre-defined margin for what counts as an acceptably small cost change.

There are four main limits. First, each coefficient was trained once. The present runs show low market-making activity at some larger coefficients, but they do not establish an exact collapse point, a seed-stable ordering, or training stability. Second, the study uses one stock and one period, so the same coefficient multiples may behave differently in another asset or market regime. Third, the historical background order flow does not react to the agents' orders. The replay captures detailed book mechanics, but it does not model a fully responsive market. Fourth, the 4,096 evaluation windows come from 24 days and may be correlated within day. Episode-level intervals were therefore removed from this version. A future evaluation should retain day labels and use a paired day-level or hierarchical bootstrap. Such intervals would still not measure variation across training seeds.

The most important next experiment is to repeat the key coefficient values $\{0, 0.25, 0.5, 1\}\rho^\star$ with several seeds. A stronger study would also test more liquid and less liquid assets, report a pre-set equivalence margin for execution cost, and compare reward penalties with explicit constraints. This would show whether the market maker can be kept active without replacing inventory risk with forced, uneconomic trading.

7 Conclusion

Quadratic risk penalties provide a simple link between classical exposure control and multi-agent RL, but the evidence here is conditional on one training run per coefficient. In the available execution runs, larger remaining-quantity penalties are associated with earlier and more reliable completion. In the market-making runs, a moderate penalty lowers inventory while activity remains similar, whereas some larger-penalty runs provide almost no liquidity. The results suggest task-dependent responses, but they do not establish a seed-stable market-maker transition or a precise execution-cost effect. Multi-seed replication and day-level or hierarchical uncertainty are needed for stronger conclusions.

A Reproducibility

The code, configuration files, and analysis scripts are available in the project repository. The repository and the paper source contain the complete hyperparameters, coefficient values, action mappings, and run records.

Data. We use AMZN LOBSTER message and ten-level order-book files during regular trading hours (09:30–16:00), with a

$0.01 tick. The chronological splits contain 74 training days (12 Nov 2025–2 Mar 2026), 25 validation days (3 Mar–7 Apr 2026), and 24 test days (8 Apr–11 May 2026). Each episode has 64 decision steps and 100 historical messages per step. The execution task is 600 shares.

Training. Policies are trained with recurrent IPPO in 4,096 parallel environments for up to 5×10^8 environment steps. Each PPO update uses four epochs and four minibatches, with a clip value of 0.2. Training uses adaptive moment estimation with $\epsilon = 10^{-5}$ and an initial learning rate of 2.5×10^{-4} that decreases linearly to zero. Each agent uses a 256-unit input layer, a 256-unit GRU, and separate actor and critic heads. The separate coefficient sweeps use training seed 42, while the joint grid uses seed 44. There is one training run per coefficient.

Trading setup. The market maker chooses between fixed 10-share two-sided quotes and a no-quote action. The execution agent chooses 10, 20, or 50 shares at four price levels. Unfinished quantity is assigned a terminal fill three ticks beyond the far touch. The market-maker reward includes a 0.4-basis-point maker rebate, and no fee parameter is varied. Exact action definitions and remaining settings are provided in the repository. The licensed LOBSTER files are not redistributed.

References

[1] R. Almgren and N. Chriss. Optimal execution of portfolio transactions. *Journal of Risk*, 3:5–39, 2000.

[2] L. Ardon, N. Vadori, T. Spooner, M. Xu, J. Vann, and S. Ganesh. Towards a fully RL-based market simulator. In *Proc. ACM ICAIF*, 2021.

[3] M. Avellaneda and S. Stoikov. High-frequency trading in a limit order book. *Quantitative Finance*, 8(3):217–224, 2008.

[4] C. S. de Witt, T. Gupta, D. Makoviichuk, V. Makoviychuk, P. H. S. Torr, M. Sun, and S. Whiteson. Is independent learning all you need in the StarCraft multi-agent challenge? arXiv:2011.09533, 2020.

[5] S. Frey, K. Li, P. Nagy, S. Sapora, C. Lu, S. Zohren, J. Foerster, and A. Calinescu. JAX-LOB: A GPU-accelerated limit order book simulator. In *Proc. ACM ICAIF*, 2023.

[6] B. Gašperov, S. Begušić, P. Posedel Šimović, and Z. Kostanjčar. Reinforcement learning approaches to optimal market making. *Mathematics*, 9(21):2689, 2021.

[7] M. Karpe, J. Fang, Z. Ma, and C. Wang. Multi-agent reinforcement learning in a realistic limit order book market simulation. In *Proc. ACM ICAIF*, 2020.

[8] Y.-S. Lim and D. Gorse. Reinforcement learning for high-frequency market making. In *Proc. ESANN*, 2018.

[9] S. Lin and P. A. Beling. An end-to-end optimal trade execution framework based on proximal policy optimization. In *Proc. IJCAI*, pages 4548–4554, 2020.

[10] V. Mohl, S. Frey, R. Leyland, K. Li, G. Nigmatulin, M. Cucuringu, S. Zohren, J. Foerster, and A. Calinescu. JaxMARL-HFT: GPU-accelerated large-scale multi-agent reinforcement learning for high-frequency trading. arXiv:2511.02136, 2025.

[11] Y. Nevmyvaka, Y. Feng, and M. Kearns. Reinforcement learning for optimized trade execution. In *Proc. ICML*, pages 673–680, 2006.

[12] A. Rutherford et al. JaxMARL: Multi-agent RL environments and algorithms in JAX. In *Advances in Neural Information Processing Systems*, 2024.

[13] N. Vadori, L. Ardon, S. Ganesh, T. Spooner, S. Amrouni, J. Vann, M. Xu, Z. Zheng, T. Balch, and M. Veloso. Towards multi-agent reinforcement learning-driven over-the-counter market simulations. *Mathematical Finance*, 34(2):262–347, 2024.

[14] T. Spooner, J. Fearnley, R. Savani, and A. Koukorinis. Market making via reinforcement learning. In *Proc. AAMAS*, pages 434–442, 2018.