# Reinforcement Learning Market Makers in a Glosten-Milgrom Environment

Salim Bennouna, Stefan Jelenkovic, Sherif Karam, Joe Souaid

## ABSTRACT

We investigate whether a simple RL agent can act as an *uninformed* market maker in a Glosten–Milgrom environment, where any liquidity provider must be *subsidized*, that is, expected to lose money to informed traders, and where the central objective is therefore to *minimize* this unavoidable cost. Using a modular agent-based limit order book and tabular SARSA, we study how reward shaping and state design affect the agent's ability to manage inventory and interact with informed and uninformed flow. A combined PnL + inventory-penalization reward yields a policy that significantly outperforms our benchmark.

## 1 INTRODUCTION

In modern financial markets, buyers and sellers do not typically meet directly. Instead, trading occurs through a limit order book (LOB): data structure that aggregates outstanding buy and sell orders at different prices. At any time, the best available sell price is the ask, the best available buy price is the bid, and their difference is the bid–ask spread. Any trader needing immediate execution must cross this spread and pay for immediacy. A market maker is an agent who aims to profit from this spread by continuously posting bid and ask quotes. By doing so, it is remunerated for providing liquidity to the market. Like many trading problems, it has become increasingly automated [17], since electronic market makers run low-latency algorithms that continuously update quotes on microsecond scales. This paper investigates reinforcement learning as a methodology for constructing such market-making agents.

### 1.1 Related work.

The largest line of work treats the market maker as a risk-averse stochastic controller. Ho and Stoll's seminal optimal dealer model formulates quoting as a dynamic trade-off between expected spread capture and inventory risk [18]. Avellaneda and Stoikov refine this paradigm in continuous time, deriving optimal bid and ask quotes under price dynamics and quadratic inventory penalties [2]. Cartea et al. further generalize this inventory-based perspective within a Hamilton–Jacobi–Bellman framework, unifying market making, optimal execution, and risk management into a coherent stochastic control approach [6, 15].

More recently, a second strand leverages reinforcement learning to learn quoting policies directly from interaction with simulated markets. Spooner et al. propose Q-learning-based and actor–critic agents that rediscover inventory-aware quoting rules and benchmark favourably against Avellaneda–Stoikov-type baselines [27]. Kumar et al. investigate deep RL for high-frequency market making, focusing on richer state representations and neural policies [20]. Other formulations have also been explored: Spooner and Savani train market-making agents against worst-case opponents via adversarial RL [28], while Falces Marin et al. use RL to adapt and improve upon the Avellaneda–Stoikov heuristic in a data-driven fashion [11].

Finally, a significant line of research studies market making in *prediction markets* [4, 22, 23]. Prediction markets are automated trading venues designed not for profit maximization, but for *information aggregation*: the market maker's role is to continuously quote prices that elicit private beliefs from informed participants and cause market prices to track consensus probabilities. Although some later works consider profitability, the objective and modelling assumptions in this setting remain fundamentally different from those of financial market making.

### 1.2 Our approach.

Our idea was to combine the reinforcement-learning methods developed for financial market making with the information-elicitation principles central to prediction-market market makers. Rather than optimizing spread capture alone, we ask whether an RL agent can operate in a financial LOB while also learning to extract information from order flow in the spirit of prediction markets. Let's detail this.

In order to earn the spread, market makers are exposed to two main risks:

**1. Inventory risk.** Market makers cannot buy and sell in perfectly offsetting amounts, so they accumulate inventory over time. Holding such a position exposes them to losses when prices subsequently move in the opposite direction.

**2. Adverse selection.** The market maker typically trades at stale quotes while other traders may possess more up-to-date or superior information. If a buy (resp. sell) market order arrives just before a downward (resp. upward) price move, the market maker is "picked off" at an unfavorable moment:

$$\mathbb{E}\big[S_{t+\delta} - S_t \,\big|\, \text{trade hits the ask at } t\big] < 0.$$

Such information asymmetry erodes spread profits.

Real-world market makers mitigate these two risks by relying on (i) sophisticated pricing models to estimate the asset's "fair" value and reduce adverse selection, and (ii) explicit inventory-management mechanisms. In our work, we *deliberately remove* such pricing models: the market maker is treated as uninformed, in the spirit of the Glosten–Milgrom framework [13]. This isolates the information-elicitation channel, since any insight the market maker acquires must come solely from how informed and uninformed traders respond to its quotes. Within this setting, we can directly test whether a RL agent can both manage inventory and extract latent information from order flow, thereby blending the prediction-market emphasis on information extraction with the financial market-making problem introduced above.

In the Glosten–Milgrom environment, a subset of traders possesses noisy but superior estimates of the latent fundamental, while the market maker remains structurally disadvantaged. Informed traders will systematically trade against mispriced quotes, making profitability unattainable in expectation. Performance must therefore be evaluated by asking which policies lose the least while still providing liquidity. We study this environment because it aligns naturally with our objective: to assess whether an RL-based market maker can mitigate losses despite its informational disadvantage.

## 2 SIMULATOR DESCRIPTION

We now describe the LOB environment in which our learning market maker is trained and evaluated. The simulator is an agent-based, single-asset market with a latent "fundamental" value process, and a heterogeneous population of informed and uninformed traders. The design is inspired by existing agent-based market simulators [1, 21, 5, 19, 24]. The environment is not intended to be fully realistic; rather, its purpose is to provide a controlled and relevant framework for comparing our market makers. All calibration parameters are reported in Appendix A.

The code is available at `https://github.com/SalimBennouna/rl-market-makers-glosten-milgrom`.

### 2.1 Fundamental value and oracle

At the core of the simulator is a latent fundamental value process $r_t$ for the asset. This value is *never directly observable* to the market; it plays the role of the hidden fundamental around which transaction prices should, in principle, revolve.

We model $r_t$ as a mean-reverting Ornstein–Uhlenbeck (OU) process with occasional jumps:

$$dr_t \;=\; \kappa(\bar{r} - r_t)\,dt \;+\; \sigma\,dW_t \;+\; J_t\,dN_t,$$

where $\bar{r}$ is the long-run mean, $\kappa$ the mean-reversion speed, $\sigma$ the diffusion volatility, $W_t$ a Brownian motion, and $N_t$ a Poisson process whose jump times correspond to rare "megashocks". This specification is standard in market-making models[7]: mean-reversion creates a stable yet information-rich environment for liquidity provision, and occasional jumps generate rare but severe information shocks that accentuate adverse selection.

An *oracle* module generates and stores the entire latent path $\{r_t\}_{t\in[0,T]}$ before the simulation starts, given a random seed. The same seed therefore produces identical fundamental trajectories across runs, enabling controlled, head-to-head comparisons of different market-making policies. Informed traders can query the oracle for noisy observations $o_t \sim \mathcal{N}(r_t, \sigma_n^2)$ where $\sigma_n^2$ controls signal quality. Different agents draw independent noise realizations even when querying at the same time, capturing heterogeneous research signals. Crucially, market makers never observes $r_t$ or $o_t$; they only see publicly available order-book information. This architectural choice is a key assumption of the Glosten-Milgrom model.

### 2.2 Limit order book and time structure

Prices are defined on a discrete grid with tick size equal to one cent. Agents submit limit and market orders; the matching engine maintains a standard price-time-priority book and updates it whenever an order arrives, is cancelled, or is executed.

Time in the simulator is event-driven. Each agent is equipped with a wake-up process (typically fixed, exponential or uniform over the session), and decisions are made only at wake-up times. At each wake-up, an agent may inspect the current state of the book (best bid/ask, depth, recent trades), optionally query the oracle if it is informed, update its internal state (beliefs, inventory, technical indicators), and then submit or cancel orders. This asynchronous interaction produces an endogenous sequence of order arrivals that mixes informed trades, liquidity-motivated trades, and technical trading, the continuous-time analogue of the Glosten-Milgrom "one trader per period" setup.

### 2.3 Agent population: informed and uninformed flow

The Glosten–Milgrom environment necessitates several families of agents that fall into two broad categories: *informed* agents whose trading direction is statistically linked to the latent fundamental $r_t$, and *uninformed* agents. Our market maker competes and interacts with this ecology.

#### 2.3.1 Informed agents.

All informed agents share the same basic structure: at wake-up, they query the oracle for a noisy observation $o_t$ and use it to update a Bayesian belief about the latent fundamental $r_t$ before deciding how and where to trade. Formally, each informed agent maintains a Gaussian prior $\mathcal{N}(\mu_{t-}, \sigma^2_{t-})$ over $r_t$ from its previous wake-up. First, it advances this prior under the discrete-time OU dynamics over the elapsed time $\Delta$:

$$\mu'_t = \left(1 - (1-\kappa)^{\Delta}\right)\bar{r} + (1-\kappa)^{\Delta}\mu_{t-},$$
$$\sigma'_t = (1-\kappa)^{2\Delta}\sigma^2_{t-} + \frac{1-(1-\kappa)^{2\Delta}}{1-(1-\kappa)^2}\,\sigma^2_s.$$

where $\sigma^2_s = \sigma^2/(2\kappa)$ is the stationary variance of the continuous-time OU proces. The agent then combines this prior with the new observation using variance weights. In our implementation, $\sigma_n$ and $\sigma'_t$ are treated directly as variances, and the posterior mean and variance are updated as

$$\mu_t = \frac{\sigma_n}{\sigma_n + \sigma'_t}\,\mu'_t + \frac{\sigma'_t}{\sigma_n + \sigma'_t}\,o_t, \qquad \sigma_t = \frac{\sigma_n\,\sigma'_t}{\sigma_n + \sigma'_t}.$$

For simplicity, posterior uncertainty $\sigma_t$ is not propagated beyond the next OU step in the code, so the belief dynamics are driven primarily by the mean $\mu_t$.

**Value-based agents.** Economically, these agents play the role of prototypical fundamental investors. They project their updated belief to the end of the session,

$$r_T = \left(1 - (1-\kappa')^{\Delta}\right)\bar{r} + (1-\kappa')^{\Delta}\,\mu_t,$$

and trade small size in the direction of perceived mispricing, placing passive or mildly aggressive limit orders around the spread. They thus embody the canonical informed traders of a Glosten–Milgrom environment.

We observed initially that Value agents traded almost exclusively near the end of each session. This behavior stemmed not from their logic but from the projection parameter $\kappa'$, whose calibration strongly influences perceived mispricing. A detailed diagnostic of this effect, together with illustrative examples, is provided in Appendix C.

**Zero-Intelligence agents.** ZI agents follow the same Bayesian update, and then draw a private valuation shock $\theta$ and a desired price improvement $R$, forming an effective valuation $v = r_T + \theta$. They set a limit price $p = v - R$ for buys or $p = v + R$ for sells. If the inside quote already offers at least this price improvement, they cross the spread; otherwise they post a passive order at $p$. Economically, ZI agents represent lightly structured but informed liquidity demanders: they trade in the correct direction on average, but their exact pricing remains noisy and myopic.

**Heuristic belief-learning agents.** HBL agents use the same Bayesian belief as ZI agents but also incorporate recent order-flow patterns. They estimate which price levels are most likely to fill profitably and choose their quotes accordingly. They represent informed traders who react to both fundamentals and microstructure signals.

#### 2.3.2 Uninformed agents.

Uninformed agents can't query the oracle. They provide the "noise trader" component that is central in the Glosten-Milgrom framework.

**Noise agents.** Noise agents generate purely exogenous order flow. Each wakes once per session, observes the inside quotes, randomly chooses buy or sell, and submits a single market order of random size. They represent non-informational trading.

**Technical agents.** These agents represent purely trend-following strategies. They compare short- and long-term moving averages of the midprice: if $\mathrm{MA}_{20} \geq \mathrm{MA}_{50}$ they buy at the inside ask, otherwise they sell at the inside bid.

**Baseline market-maker.** At each wake-up, it reads the midprice and spread, then posts fixed-size symmetric quotes: a bid at mid − spread/2 and an ask at mid + spread/2, cancelling previous quotes each time. In effect, it supplies liquidity at the current best bid and ask. This agent serves as the benchmark the RL market makers aim to outperform.

## 3 ALGORITHM DESIGN

Because the market simulator is already noisy, multi-agent, and partially observable, our guiding philosophy was to keep the RL component deliberately simple: a small, discrete state–action space trained with tabular SARSA. This ensured tractability, stability, and interpretability while still allowing the agent to react meaningfully to order-flow conditions. We imposed several structural constraints from the outset:

**Low-dimensional state space.** Our very first experiments were too ambitious in terms of dimensionality of the state space. Analyzing state visitation patterns, we found that many states were never visited, and even long training runs left most of the Q-table unfilled. From there, we always kept the state space low-dimensional. We discretized variables into a small number of bins to keep the state space fully explorable within a few simulated trading days.

**Small action space.** All agents submit fixed-size orders and use single-level quoting (at most one bid and one ask). This focuses the learning problem on *where* to quote rather than on sizing or multi-level book management. Although the exact action sets differ slightly across experiments, they always consist of a small grid of bid/ask offsets, supplemented by an optional `no-quote` action.

**On-policy TD learning (SARSA).** We opted for temporal-difference methods because they update value estimates after every interaction step and do not require waiting for episode termination. By bootstrapping from their own value estimates, TD methods learn efficiently in continuous, noisy environments such as ours. Among TD algorithms, we chose SARSA because it is on-policy and therefore less prone to overestimation and divergence than Q-learning in highly stochastic environments.

**Short-horizon objective.** A discount factor of $\gamma = 0.95$ reflects the inherently short-term nature of market making and helps stabilize TD updates.

## 4 RESULTS

In this section, we recount the sequence of modelling choices, failed experiments, and diagnostic insights that shaped our final design. All comparisons use common random numbers (CRG) to ensure variance reduction and fair, head-to-head evaluation.

Figures are available in Appendix B.

### 4.1 Attempt 1: PnL-Only Reward

We began with an idealized setup containing only Noise agents. In this configuration, there is no adverse selection and the market is essentially driven by our own quotes. Inventory risk is negligible, since buys and sells arrive symmetrically.

A market maker in such a setting should learn to widen the spread to profit from Noise agents crossing it. We therefore defined a reward purely based on mark-to-market PnL:

$$r_t = \Delta \mathrm{PnL}_t,$$

and used a minimal state consisting only of the market spread. Actions were a fixed grid of symmetric quoting offsets: $\{\pm 2, \pm 15, \pm 50\}$.

The resulting PnL trajectory is shown in Figure 1.1. The agent earns the spread at a nearly constant rate with low variance, due to the high number of Noise agents. Inventory remains near zero (Figure 1.2), reflecting symmetric order flow.

To assess learning, we inspect the agent's action choices (Figure 1.3). After an initial exploration phase, the agent selects almost exclusively the widest-spread action, maximizing expected reward. The reward and cumulative PnL curves (Figure 1.4) show steady improvement once the policy stabilizes.

The evolution of the maximum Q-value (Figure 1.5) shows rapid growth followed by plateauing, consistent with convergence of the value estimates. We can also fix a representative state and compare terminal Q-values: the Q-value of the offset of 50 outperforms that of 15 by 48%, which itself outperforms the offset of 2 by 24% (Figure 1.6). Finally, examining the evolution of these Q-values confirms that this ordering persists over time (Figure 1.7).

Notice that up to now, the states have not been very informative, since the agent is the sole provider of liquidity in the market and therefore gains no information from observing the market spread. The state representation only becomes meaningful once we add other agents that can post quotes rather than merely take liquidity, such as the ZI agents. When we introduce even a small number of them to quantify this effect, we quickly encounter a major problem: our PnL-only rewards is extremely noisy. This high variance in returns reduces the effectiveness of learning, leading the agent to unstable behaviour and a lack of convergence even over long training horizons.

This instability is illustrated in Figure 1.8, which shows the reward averaged over a rolling window of 2000 steps (and still volatile!). Such behaviour is well known in the literature: relying solely on PnL as a reward signal typically poor learning dynamics, as documented in [29, 27, 12]. Our observations here are consistent with these findings.

### 4.2 Attempt 2: Inventory-Only Reward

To reduce variance and stabilize the learning, we add an inventory regularization term to the reward. To understand its behavior, we first consider it alone:

$$r_t = -\lambda |\mathrm{inventory}_t|^p, \qquad p \in \{2, 3\}.$$

Quadratic and cubic penalization both exist in the litterature [27, 28]. We found no major differences between the two. For the environment, we consider the full setup with all agents types in non-null quantities. We augment the state space with inventory, so that the agent can observe it. For the action set, the agent can simply "not quote", or choose actions biased toward long, biased toward short, or symmetric quoting offsets. In this setup, we focus purely on teaching the agent to manage inventory risk, and our goal is to make the agent learn that it should quote so as to reduce its position (e.g., widen the bid relative to the ask when already long).

As expected, the agent quickly learns to quote so as to bring its inventory to zero, and then stops quoting altogether to keep it at zero (Figure 2.1). The fact that it is learning can be seen through the reward trajectory (Figure 2.2), which becomes quasi-stationary around zero after a certain number of steps (up to $\epsilon$-exploration). We can also clearly observe in the action trajectories that the `no-quote` action eventually dominates (Figure 2.3).

Moreover, the agent clearly outperforms the baseline in terms of PnL (Figure 2.4), since the baseline consistently loses to informed flow by taking inventory in the wrong direction, while our agent keeps positions small and eventually stops trading altogether. This performance initially appears impressive, but it comes with an obvious drawback: a severe collapse in market liquidity. Because the agent rarely posts inside quotes, the spread widens dramatically. This effect is evident in Figure 2.5, where the spread steadily increases and then jumps sharply at the moment the agent stops quoting (around 9 pm).

This experiment revealed a fundamental issue: rewarding inventory reduction too strongly incentivizes the agent to stop functioning as a market maker.

## 4.3 Attempt 3: PnL + Inventory Reward

We now introduce both terms into the reward.

$$r_t = \Delta\text{PnL}_t - \lambda|\text{inventory}_t|^2$$

The goal is for the agent to keep its inventory close to zero while still benefiting from spread capture.

After extensive tuning of $\lambda$, we identified a range of values that yields a hybrid policy: the agent maintains an inventory close to zero but "oscillates" around it by actively posting quotes instead of relying on the `no-quote` action. This behaviour is visible in the inventory path (Figure 3.1), where after a learning phase, the oscillations around 0 are larger than those observed in Attempt 2. This is further supported by the action trajectory (Figure 3.2), where the actions skewed toward the bid and those skewed toward the ask account for more than 90% of all choices after training, while the `no-quote` action becomes nearly obsolete. This indicates that the agent has understood that it can reduce its inventory penalty while still earning spread revenue by alternating its skew.

In terms of performance, this approach allows the agent to significantly outperform the baseline: during the first 24 hours, the RL agent loses only about 1% of its bankroll, compared with roughly 8% for the benchmark (Figure 3.3). The baseline's PnL is also far more volatile, indicating greater positional risk and less consistent spread capture.

This suggests not only that the RL agent outperforms the simple MM financially, but that it may also provide *better liquidity*. To verify this, we compare the midprice in both simulations to the true latent fundamental price (Figure 3.4). Strikingly, the midprice under the RL agent (orange) tracks the latent fundamental (blue) much more closely than the midprice under the baseline (green). This demonstrates that the market is substantially more efficient when liquidity is provided by the RL agent.

In summary, reinforcement learning in this configuration allows the agent to outperform a reasonable baseline while simultaneously improving market quality. It both reduces the market maker's subsidization cost and increases price efficiency. This exceeded our expectations.

## 4.4 Attempt 4: Learning from Flow

Up to this point, we have succeeded in teaching the agent to manage inventory, but it still does not exploit the information embedded in informed order flow, nor does it learn to mitigate adverse selection. To address this, two changes are required:

**(1) State features must contain information about informed trading.** The agent must observe variables that correlate with informed participation. We introduce two such microstructure features:

- **Order-flow imbalance (OFI).** Net signed trading volume over the most recent $N$ trades:

  $$\text{OFI}_t = \sum_{k=t-N}^{t} \text{sign}(\text{trade}_k) \cdot \text{size}_k,$$

  OFI is a well-established predictor of short-horizon price movements and a proxy for informed flow [9, 25].

- **Depth imbalance at the best quotes (DI).** Using volumes at the best bid/ask, we define:

  $$\text{DI}_t = \frac{Q_{\text{bid}} - Q_{\text{ask}}}{Q_{\text{bid}} + Q_{\text{ask}}},$$

  DI captures short-term pressure from liquidity and is known to correlate with the sign of the next price move [14, 27].

Together, they should allow the agent to observe directional pressure in both trades and quotes, enabling it to recognize *"toxic"* flow.

**(2) A reward that penalizes adverse selection.** To learn from informed flow, the agent must recognize that being "picked off" is costly. Instantaneous PnL does not capture this well. A commonly used technique is to incorporate a *markout* term [30].

When the agent trades at price $p_t$, we evaluate the midprice $m_{t+\tau}$ at a short future horizon $\tau$, and define the markout as:

$$\mathrm{MO}_t = \begin{cases} m_{t+\tau} - p_t, & \text{(buy)} \\ p_t - m_{t+\tau}, & \text{(sell).} \end{cases}$$

A negative markout indicates that the agent traded just before the price moved against it,a signature of adverse selection [30, 31].

We therefore modify the reward to

$$r_t = \Delta\mathrm{PnL}_t \; - \; \alpha\,\mathrm{MO}_t \; - \; \lambda|\mathrm{inventory}_t|^2,$$

This formulation rewards profitable spread capture while discouraging trades that systematically lose to informed flow.

We expected Attempt 4 to produce a policy that meaningfully improved upon Attempt 3. However even after extensive tuning this was a failure. Below are our thoughts on how to explain this.

**(a) Markout term variance.** In principle, the markout should signal adverse selection. In practice, however, it introduces very high variance: many unrelated agent actions occur between the market maker's trade and the evaluation horizon $\tau$, so the same state–action pair produces widely different outcomes across episodes. Moreover, as the market maker's policy evolves, the surrounding order-flow distribution shifts, making the transition dynamics strongly non-stationary.

These factors break the association between a quoting decision and its delayed markout, and SARSA fails to form stable value estimates. In OFI- and DI-sensitive states, Q-values oscillated or failed to converge across seeds, consistent with the instability expected in multi-agent, partially observable settings with noisy delayed rewards.

**(b) Structurally conflicting three-objective reward.** The reward bundles three objectives that are *not* aligned for an uninformed market maker:

1. **Spread capture** encourages frequent quoting close to the inside.

2. **Adverse-selection avoidance** encourages quoting *less*,or much wider,precisely when order flow becomes informative.

3. **Inventory control** encourages symmetric quoting and sometimes withdrawing liquidity.

We believe this is a case of *scalarization failure*: compressing multiple conflicting objectives into a single scalar reward leads to a brittle optimization landscape. Across a wide range of $(\alpha, \lambda)$, we observed knife-edge behaviour: slightly increasing $\alpha$ collapsed the policy into the "do-not-quote" regime (similar to Attempt 2). Slightly decreasing $\alpha$ made the markout term effectively irrelevant (similar to Attempt 3).

There was no broad parameter region where the TD updates converged to a stable policy that improved all objectives simultaneously. The agent faces incompatible gradients: the reward signal pushes the value function in different directions depending on the local balance of spread capture, markouts, and inventory pressure.

## 5 CONCLUSIONS

This work examined whether a simple, uninformed RL market maker can operate effectively in a Glosten–Milgrom environment. Through a sequence of experiments, we found that tabular SARSA can learn useful behaviours, such as widening the spread in pure-noise markets and maintaining near-zero inventory while still benefiting from the spread when combining PnL and inventory penalties. In this configuration, the agent outperformed our baseline both in terms of subsidization costs and price efficiency.

However, extending the framework to learn from informed flow proved unsuccessful. As documented by [8] and [26], TD methods struggle in high-noise, partially observable market-making settings. Our results confirm these limitations: delayed markout rewards introduced substantial variance, and the scalar reward blended incompatible objectives (spread capture, inventory control, adverse-selection avoidance), ultimately leading to unstable learning.

Future work includes several directions motivated by our findings:

1. **Revisiting Attempt 4**: redesigning the adverse-selection penalty to reduce reward noise, incorporating belief-tracking or latent-state inference to handle partial observability.

2. **Testing alternative tabular RL algorithms:** Expected SARSA, Double Q-learning...

3. **Moving beyond coarse discretization:** linear function approximation, tile coding, deep RL methods...

4. **Exploring multi-objective RL formulations.**

## Appendix A: Main Calibration Parameters

| Symbol / Name | Description | Value / Default |
|---|---|---|
| **Latent fundamental (OU + jumps)** | | |
| $\bar{r}$ | Long-run mean of fundamental value | 100,000 |
| $\kappa$ | Mean-reversion speed | $1 \times 10^{-14}$ |
| $\sigma$ | OU diffusion volatility | $5 \times 10^{-9}$ |
| $\sigma_s$ | OU stationary std. ($\sigma_s^2 = \sigma^2/(2\kappa)$) | $3.53 \times 10^{-2}$ |
| $\lambda_J$ | Jump (megashock) arrival intensity | $2.7 \times 10^{-18}$ |
| $\mu_J$ | Mean jump size | $10^3$ |
| $\sigma_J$ | Jump-size std. | $\approx 224$ |
| $\sigma_n$ | Oracle noise std. in $o_t \sim \mathcal{N}(r_t, \sigma_n^2)$ | agent-dependent (see below) |
| $J_t$ | Jump-size process | $\mathcal{N}(\mu_J, \sigma_J^2)$ |
| $N_t$ | Poisson jump-count process | $\text{Poisson}(\lambda_J)$ |
| **Time and market microstructure** | | |
| Time resolution | Internal clock | nanosecond |
| Tick size | Price grid spacing | \$0.01 (integer cents) |
| **Agent population (varies by experiment, typical setup)** | | |
| $N_{\text{MM}}$ | Market makers | 1 |
| $N_{\text{Value}}$ | Value-based informed agents | 100 |
| $N_{\text{ZI}}$ | ZI informed agents | 50 |
| $N_{\text{HBL}}$ | HBL informed agents | 50 |
| $N_{\text{Noise}}$ | Noise agents | 5000 |
| $N_{\text{Mom}}$ | Technical agents | 25 |
| **RL market maker** | | |
| $\gamma$ | Discount factor | 0.95 |
| $\alpha$ | Learning rate | 0.10 |
| $\epsilon_0$ | Initial exploration rate | 0.10 |
| $\epsilon(t)$ | Decaying exploration | exponential; half-life 5 h |
| $Q_{\text{init}}$ | Initial Q-value | 0 |
| Base size | Limit order size | Varies. Typically 10 shares |
| Wake cadence | RLMM wake-up frequency | every second |
| Inventory limit | Hard inventory limit | 100 shares |
| Inventory clip | $\|q_t\|$ clipping level | 100 shares |
| Inventory bin | Inventory bin size | 10 |
| Spread clip | Spread clipping | model-dependent |
| Spread bin | Spread bin size | model-dependent |
| $w_{\text{MO}}$ | Markout weight | 0.1 |
| $\tau$ | Markout horizon | 30 s |
| OFI lookback $N$ | Trades used for OFI | 50 trades |
| OFI bin size | OFI discretization step | 100 |
| OFI clip | OFI clipping | 5000 |
| Depth-imbalance bins | DI discretization | $\{-0.5, -0.1, 0.1, 0.5\}$ |

| Symbol / Name | Description | Value / Default |
|---|---|---|
| **Per-agent calibration** | | |
| *ValueTrader* | | |
| $c_0$ | Starting cash | 103,000 |
| $\sigma_n$ | Signal noise std | 9,500 |
| $\bar{r}$ | Belief long-run mean | 98,500 |
| $\kappa$ | Belief mean-reversion | 0.047 |
| $\sigma_s$ | Belief stationary std | 98,000 |
| $\lambda_a$ | Wake intensity | 0.0048 |
| `aggr_ratio` | Active vs passive routing | 0.12 |
| Lot size | Trade size | $\in \{1, \dots, 5\}$ |
| *ZITrader* | | |
| $c_0$ | Starting cash | 102,000 |
| $\sigma_n$ | Signal noise std | 950 |
| $\bar{r}$ | Belief long-run mean | 99,000 |
| $\kappa$ | Belief mean-reversion | 0.048 |
| $\sigma_s$ | Belief stationary std | 95,000 |
| $q_{\max}$ | Max inventory target | 11 |
| $\sigma_{pv}$ | Private valuation std | $4.8 \times 10^6$ |
| $R_{\min}$ | Min price improvement | 1 tick |
| $R_{\max}$ | Max price improvement | 245 ticks |
| $\eta$ | Decay of target queue depth | 0.95 |
| $\lambda_a$ | Wake intensity | 0.0045 |
| *HBLTrader (inherits ZITrader)* | | |
| $c_0$ | Starting cash | 101,000 |
| $\bar{r}$ | Belief long-run mean | 99,500 |
| $\kappa$ | Belief mean-reversion | 0.047 |
| $\sigma_s$ | Belief stationary std | 98,000 |
| $\eta$ | Depth-decay parameter | 0.96 |
| $L$ | Order-flow lookback window | 9 |
| *Baseline market maker* | | |
| $c_0$ | Starting cash | 101,200 |
| Quote size | Fixed limit size | 18 |
| Wake cadence | Deterministic | every `1.1s` |
| Inventory cap | Hard inventory limit | 120 shares |
| *TechnicalTrader* | | |
| $c_0$ | Starting cash | 102,500 |
| Order size | Market order size | $\in [15, 85]$ |
| Wake cadence | Deterministic | every 58 s |
| MA windows | Moving-average lookback | fast = 20, slow = 50 trades |

## Appendix B: Additional Figures for Section 3

### Attempt 1

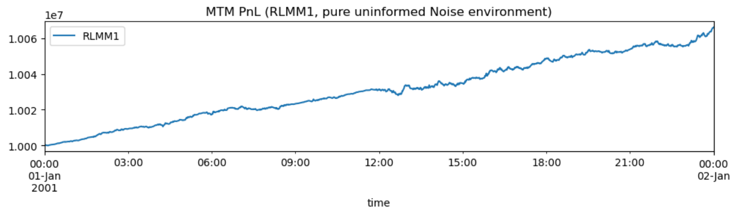

Figure 1.1:Mark-to-market PnL.

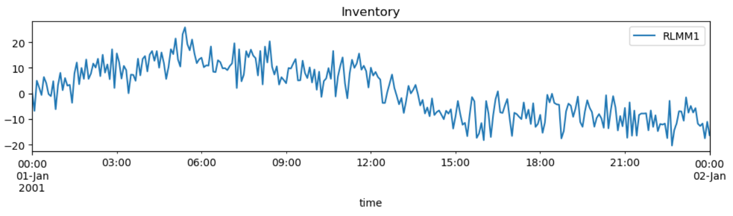

Figure 1.2:Inventory path.

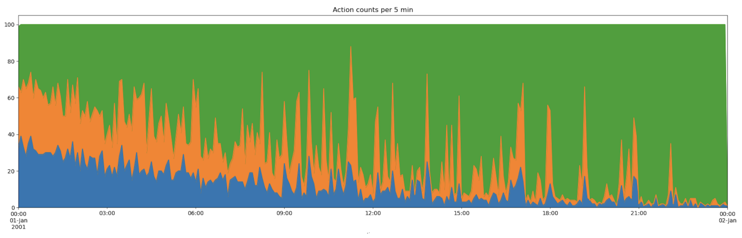

Figure 1.3:Action choices over time.

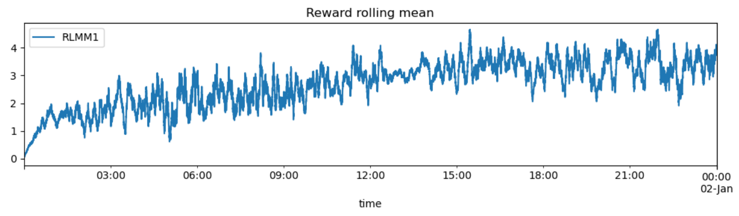

Figure 1.4:Instantaneous reward.

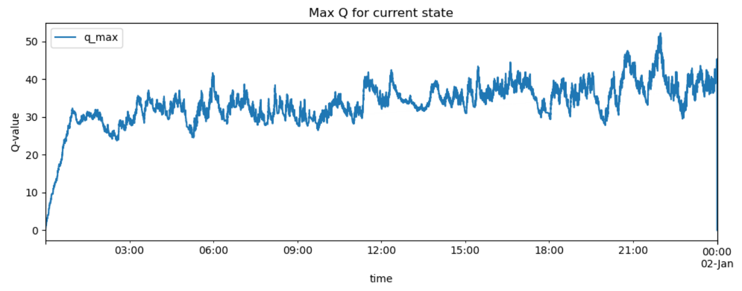

Figure 1.5:Max Q-value over time.

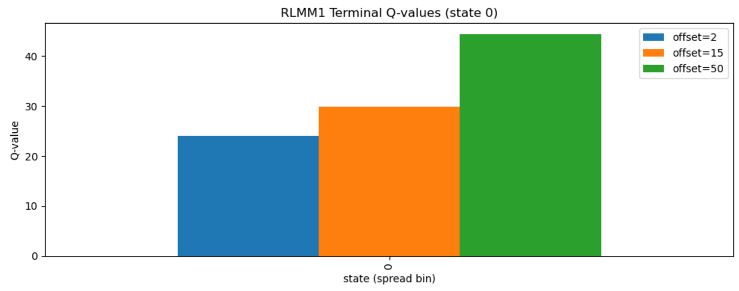

Figure 1.6:Q-values by action for a fixed state.

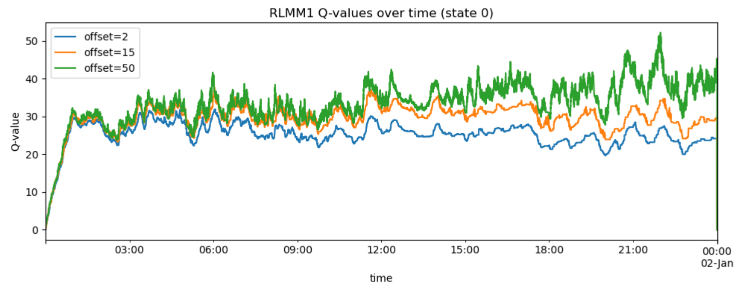

Figure 1.7:Terminal Q-values.

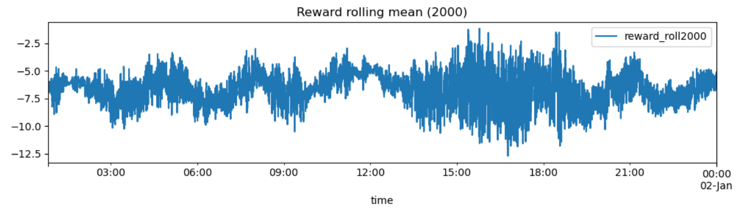

Figure 1.8:2000-step reward rolling window (with ZI agents).

## Attempt 2

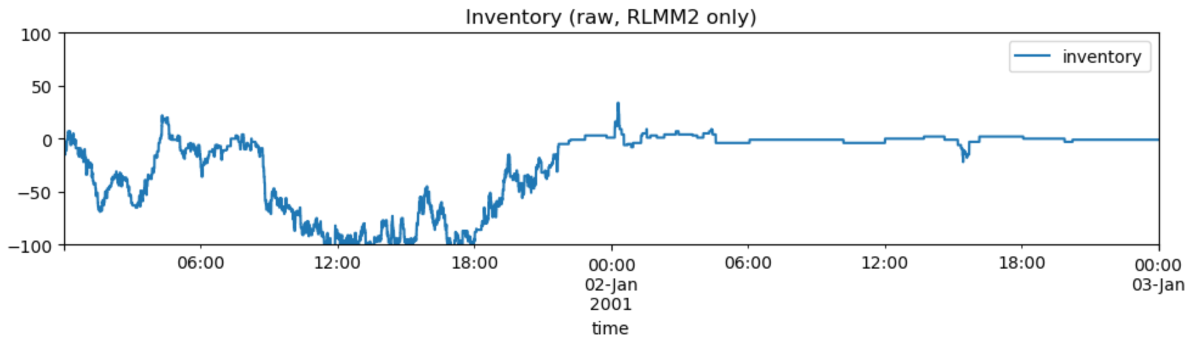

Figure 2.1:Inventory path.

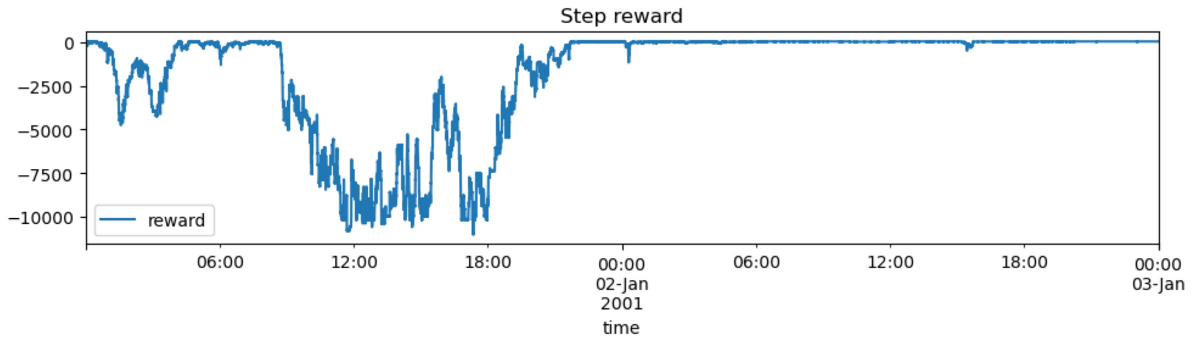

Figure 2.2:Reward trajectory.

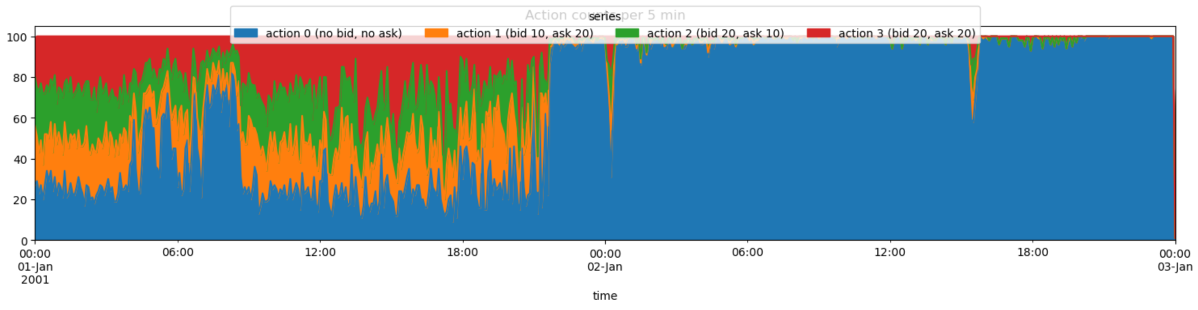

Figure 2.3:Action choices over time.

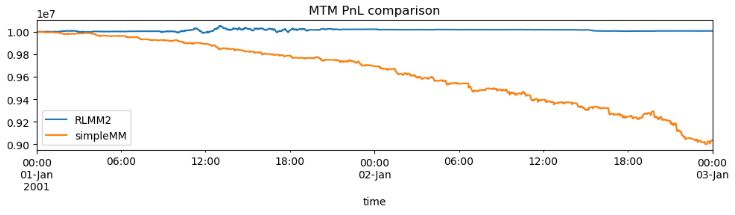

Figure 2.4:PnL comparison with baseline MM.

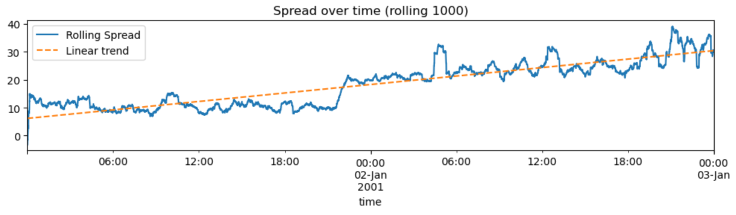

Figure 2.5:Market spread collapse.

## Attempt 3

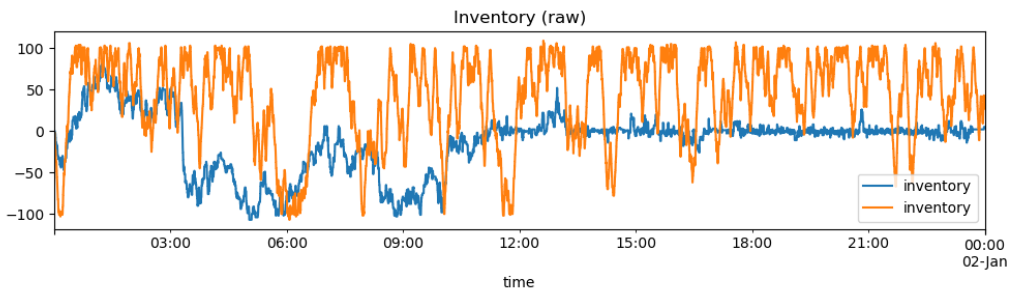

Figure 3.1:Inventory path.

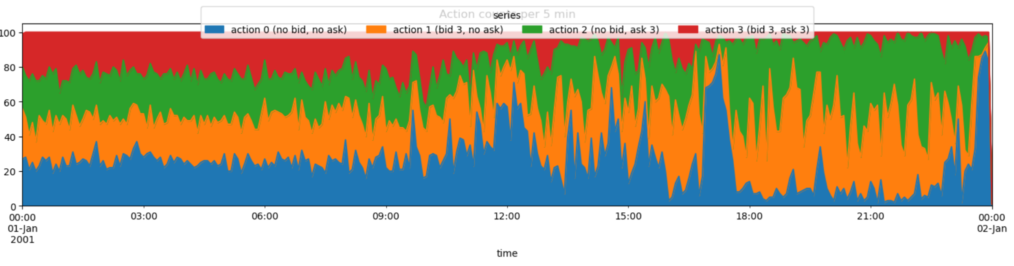

Figure 3.2:Action frequencies over time.

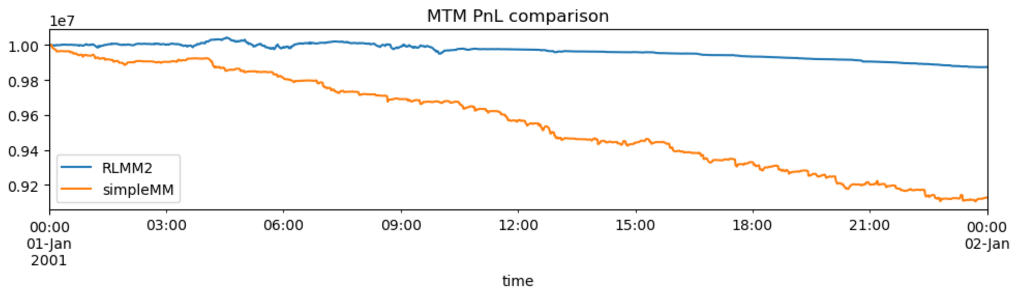

Figure 3.3:PnL comparison with baseline MM.

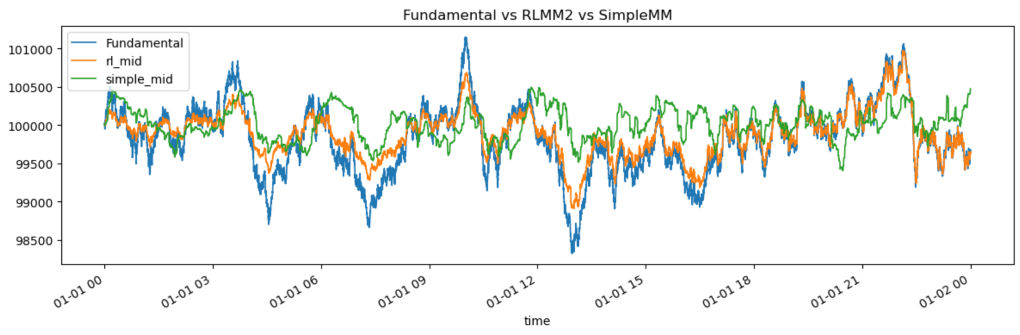

Figure 3.4:Midprice vs latent fundamental.

## Appendix C. Effect of the Projection Parameter $\kappa'$

This appendix expands on the trading behaviour of the Value-based informed agents discussed in Section 2. In early experiments, these agents traded almost exclusively near the end of each session. The issue did not lie in their trading logic but in the projection step used to map their belief $\mu_t$ to a terminal value $r_T$.

When $\kappa'$ is calibrated too large, the projection becomes overly mean-reverting: the projected terminal value is pulled strongly toward the long-run mean for most of the session and reacts only weakly to changes in $\mu_t$. As a result, the agent perceives almost no mispricing during the bulk of the trading day, and meaningful trading activity appears only when the time-to-horizon becomes small enough for the projection to move appreciably.

Reducing $\kappa'$ restores the intended behaviour: $r_T$ tracks $\mu_t$ more closely throughout the session, and Value agents trade continuously rather than only at the close.

Figures below illustrate this effect for three representative values of $\kappa'$.

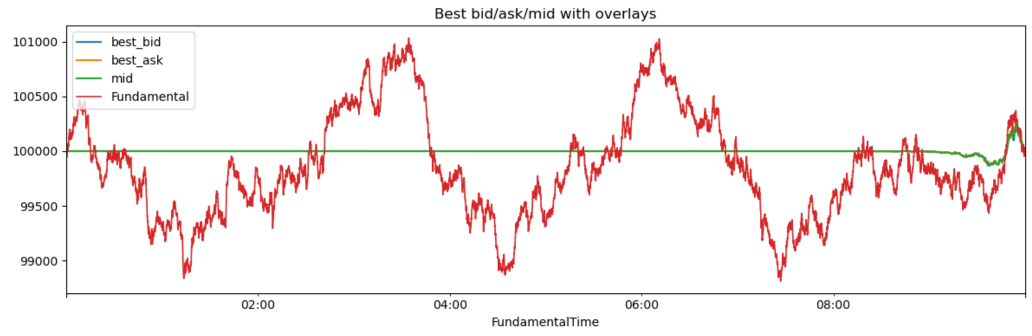

Figure C-1: Midprice vs. fundamental value for a relatively large value of $\kappa'$.

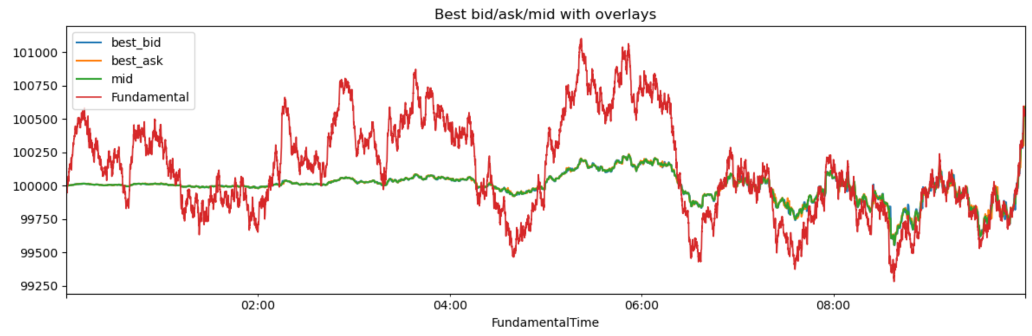

Figure C-2: Midprice vs. fundamental value for an intermediate value of $\kappa'$.

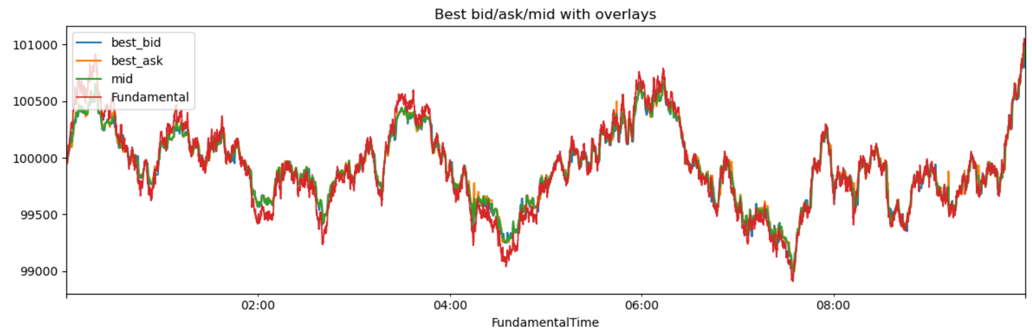

Figure C-3: Midprice vs. fundamental value for a very small value of $\kappa'$.

## References

[1] Arthur, W. B., Holland, J. H., LeBaron, B., Palmer, R., & Tayler, P. (1997). Asset pricing under endogenous expectations in an artificial stock market. In *The Economy as an Evolving Complex System II* (pp. 15–44). Addison–Wesley.

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

[3] Benth, F. E., Kallsen, J., & Meyer-Brandis, T. (2007). A non-Gaussian Ornstein–Uhlenbeck model for electricity spot price dynamics. *Applied Mathematical Finance, 14*(2), 153–169.

[4] Brahma, A., Chakraborty, M., Das, S., Lavoie, A., & Magdon-Ismail, M. (2012). A Bayesian market maker. In *Proceedings of the ACM Conference on Electronic Commerce* (pp. 215–232).

[5] Byrd, D., Hybinette, M., & Balch, T. (2019). ABIDES: Towards high-fidelity market simulation for AI research. arXiv:1904.12066.

[6] Cartea, Á., Jaimungal, S., & Penalva, J. (2015). *Algorithmic and High-Frequency Trading*. Cambridge University Press.

[7] Chakraborty, C., & Kearns, M. (2011). Market making and mean reversion. In *Proceedings of the 12th ACM Conference on Electronic Commerce* (pp. 307–314).

[8] Chan, N. T., & Shelton, C. R. (2001). An electronic market-maker. MIT AI Memo 2001–005.

[9] Cont, R., Kukanov, A., & Stoikov, S. (2014). The price impact of order book events. *Journal of Financial Econometrics, 12*(1), 47–88.

[10] Deng, S. (2005). Stochastic models of energy commodity prices and their applications: Mean reversion with jumps and spikes. Working paper, Georgia Institute of Technology.

[11] Falces Marin, J., Díaz Pardo de Vera, D., & Lopez Gonzalo, E. (2022). A reinforcement learning approach to improve the performance of the Avellaneda–Stoikov market-making algorithm. *PLOS ONE, 17*(12), e0277042.

[12] Fernández Vicente, M., Guijarro, F., & Peris, A. (2023). Automated market maker inventory management with deep reinforcement learning. *Applied Intelligence, 53*, 22249–22266.

[13] Glosten, L. R., & Milgrom, P. R. (1985). Bid, ask and transaction prices in a specialist market with heterogeneously informed traders. *Journal of Financial Economics, 14*(1), 71–100.

[14] Gould, M. D., & Bonart, J. (2015). Predicting price moves from order book imbalance. In *Market Microstructure: Confronting Many Viewpoints*. Springer.

[15] Guilbaud, F., & Pham, H. (2013). Optimal high-frequency trading with limit and market orders. *Quantitative Finance, 13*(1), 79–94.

[16] Hambly, B., Howison, S., & Kluge, W. (2009). Modelling spikes and pricing swing options in electricity markets. *Quantitative Finance, 9*(8), 937–949.

[17] Hasbrouck, J., & Saar, G. (2013). Low-latency trading. *Journal of Financial Markets, 16*(4), 646–679.

[18] Ho, T., & Stoll, H. R. (1981). Optimal dealer pricing under transactions and return uncertainty. *Journal of Financial Economics, 9*(1), 47–73.

[19] Klingert, F. M. A., & Meyer, M. (2018). Comparing prediction market mechanisms: An experiment-based and micro-validated multi-agent simulation. *Journal of Artificial Societies and Social Simulation, 21*(1).

[20] Kumar, P. (2023). Deep reinforcement learning for high-frequency market making. In *Proceedings of the 14th Asian Conference on Machine Learning* (pp. 531–546).

[21] Lux, T., & Marchesi, M. (1999). Scaling and criticality in a stochastic multi-agent model of a financial market. *Nature, 397*, 498–500.

[22] Othman, A. (2012). *Automated Market Making: Theory and Practice* (Ph.D. thesis). Carnegie Mellon University.

[23] Othman, A., Pennock, D. M., Reeves, D. M., & Sandholm, T. (2013). A practical liquidity-sensitive automated market maker. *ACM Transactions on Economics and Computation, 1*(3), 1–25.

[24] Restocchi, V., McGroarty, F., Gerding, E., & Johnson, J. E. V. (2018). It takes all sorts: A heterogeneous agent explanation for prediction market mispricing. *European Journal of Operational Research, 270*(2), 556–569.

[25] Sadighian, A. (2019). *Deep Reinforcement Learning Market Maker* (Master's thesis). University of Amsterdam.

[26] Sherstov, A. A., & Stone, P. (2004). Three automated stock-trading agents: A comparative study. In *Agent-Mediated Electronic Commerce VI* (pp. 173–187). Springer.

[27] Spooner, T., Fearnley, J., Savani, R., & Koukorinis, A. (2018). Market making via reinforcement learning. In *Proceedings of the 17th International Conference on Autonomous Agents and Multiagent Systems* (pp. 434–442).

[28] Spooner, T., & Savani, R. (2020). Robust market making via adversarial reinforcement learning. In *Proceedings of the 19th International Conference on Autonomous Agents and Multiagent Systems*.

[29] Tsantekidis, A., Passalis, N., Tefas, A., Kanniainen, J., Gabbouj, M., & Iosifidis, A. (2020). Price trailing for financial trading using deep reinforcement learning. *IEEE Transactions on Neural Networks and Learning Systems, 32*(7), 2837–2846.

[30] Wah, E., Kelly, M., & Krishnamurthy, A. (2017). Market fragmentation and venue choice: Evidence from market markouts. Working paper.

[31] Zhao, X., & Linetsky, V. (2021). High-frequency market making with inventory constraints and adverse selection. arXiv:2107.12345.