Asynchronous Deep Double Dueling Q-learning for trading-signal execution in limit order book markets

Peer Nagy[1]*, Jan-Peter Calliess[1] and Stefan Zohren[1,2,3]

[1]Department of Engineering Science, Oxford-Man Institute of Quantitative Finance, University of Oxford, Oxford, United Kingdom, [2]Man Group, London, United Kingdom, [3]Alan Turing Institute, London, United Kingdom

We employ deep reinforcement learning (RL) to train an agent to successfully translate a high-frequency trading signal into a trading strategy that places individual limit orders. Based on the ABIDES limit order book simulator, we build a reinforcement learning OpenAI gym environment and utilize it to simulate a realistic trading environment for NASDAQ equities based on historic order book messages. To train a trading agent that learns to maximize its trading return in this environment, we use Deep Dueling Double Q-learning with the APEX (asynchronous prioritized experience replay) architecture. The agent observes the current limit order book state, its recent history, and a short-term directional forecast. To investigate the performance of RL for adaptive trading independently from a concrete forecasting algorithm, we study the performance of our approach utilizing synthetic alpha signals obtained by perturbing forward-looking returns with varying levels of noise. Here, we find that the RL agent learns an effective trading strategy for inventory management and order placing that outperforms a heuristic benchmark trading strategy having access to the same signal.

1. Introduction

Successful quantitative trading strategies often work by generating trading signals, which exhibit a statistically significant correlation with future prices. These signals are then turned into actions, aiming to assume positions in order to gain from future price changes. The higher the signal frequency and strategy turnover, the more critical is the execution component of the strategy, which translates the signal into concrete orders that can be submitted to a market. Such markets are oftentimes organized as an order ledger represented by a *limit order book (LOB)*.

Limit order book prices have been shown to be predictable over short time periods, predicting a few successive ticks into the future with some accuracy. This has been done by either utilizing the recent history of order book states (Zhang et al., 2019; Zhang and Zohren, 2021), order-flow data (Kolm et al., 2021), or market-by-order (MBO) data directly as features (Zhang et al., 2021). However, given the short time horizons over which these predictions are formed, and correspondingly small price movements, predictability does not directly translate into trading profits. Transaction costs, strategy implementation details, and time delays add up to the challenging problem of translating high-frequency forecasts into a trading strategy that determines when and which orders to send to the exchange. In addition,

different predictive signals have to be traded differently to achieve optimal results, depending on the forecast horizon, signal stability, and predictive power.

In this paper, we use asynchronous off-policy reinforcement learning (RL), specifically *Deep Dueling Double Q-learning* with the *APEX* architecture (Horgan et al., 2018), to learn an optimal trading strategy, given a noisy directional signal of short-term forward mid-quote returns. For this purpose, we developed an OpenAI gym (Brockman et al., 2016) limit order book environment based on the ABIDES (Byrd et al., 2020a) market simulator, similar to Amrouni et al. (2021). We use this simulator to replay NASDAQ price-time priority limit order book markets using message data from the LOBSTER data set (Huang and Polak, 2011).

If a financial trader wants to transact a number of shares of a financial security, such as shares of cash equities, for example Apple stock, they need to send an order to an exchange. Most stock exchanges today accept orders electronically and in 2022 in US equity markets approximately 60–73% of orders are submitted by algorithms, not human traders (Mordor Intelligence, 2022). A common market mechanism to efficiently transact shares between buyers and sellers is the *limit order book (LOB)*. The LOB contains all open limit orders for a given security and is endowed with a set of rules to clear marketable orders. A more detailed introduction to LOBs can be found in Section 3.1.

High-frequency trading (HFT) is an industry with high barriers to entry and a regulatory landscape varying by geographical region. Generally, trading on both sides of the LOB simultaneously by submitting both limit buy and limit sell orders is the purview of *market makers*. While their principal role is to provide liquidity to the market, they also frequently take directional bets over short time periods. Transaction costs for such trading strategies consist of two main components, *explicit costs*, such as trading fees, and *implicit costs*, such as market impact (Harris, 2003). Trading fees vary by institution but can be negligible for institutional market makers, with some fee structures even resulting in zero additional costs if positions are closed out at the end of the day. The main consideration of transaction costs should thus be given to market impact. Our simulation environment models market impact by injecting new orders into historic order flow, thereby adding to or consuming liquidity in the market. One limitation of this approach is that adverse selection and, generally, the reactions of other market participants are not modeled.

We study the case of an artificial or synthetic signal, taking the future price as known and adding varying levels of noise, allowing us to investigate learning performance and to quantify the benefit of an RL-derived trading policy compared to a baseline strategy using the same noisy signal. This is not an unrealistic setup when choosing the correct level of noise. Practitioners often have dedicated teams researching and deriving alpha signals, often over many years, while other teams might work on translating those signals into profitable strategies. Our aim is to focus on the latter problem, which becomes increasingly more difficult as signals become faster. It is thus interesting to see how an RL framework can be used to solve this problem. In particular, we show that the RL agent learns policies superior to the baselines, both in terms of strategy return and Sharpe ratio. Machine learning methods, such as RL, have become increasingly important to automate trade

execution in the financial industry in recent years (Nagy et al., 2023), underlining the practical use of research in this area.

We make a number of contributions to the existing literature. By defining a novel action and state space in a LOB trading environment, we allow for the placement of limit orders at different prices. This allows the agent to learn a concrete high-frequency trading strategy for a given signal, trading either aggressively by crossing the spread, or conservatively, implicitly trading off execution probability and cost. In addition to the timing and level placement of limit orders, our RL agent also learns to use limit orders of single units of stock to manage its inventory as it holds variably sized long or short positions over time. More broadly, we demonstrate the practical use case of RL to translate predictive signals into limit order trading strategies, which is still usually a hand-crafted component of a trading system. We thus show that simulating limit order book markets and using RL to further automate the investment process is a promising direction for further research. To the best of our knowledge, this is also the first study applying the APEX (Horgan et al., 2018) algorithm to limit order book environments.

The remaining paper is structured as follows: Section 2 surveys related literature, Section 3 explains the mechanics of limit order book markets and the APEX algorithm, Section 4 details the construction of the artificial price signal, Section 5 showcases our empirical results, and Section 6 concludes our findings.

2. Related work

Reinforcement learning has been applied to learn different tasks in limit order book market environments, such as optimal trade execution (Nevmyvaka et al., 2006; Dabérius et al., 2019; Karpe et al., 2020; Ning et al., 2021; Schnaubelt, 2022), market making (Abernethy and Kale, 2013; Kumar, 2020), portfolio optimization (Yu et al., 2019), or trading (Kearns and Nevmyvaka, 2013; Wei et al., 2019; Briola et al., 2021). The objective of optimal trade execution is to minimize the cost of trading a predetermined amount of shares over a given time frame. Trading direction and the number of shares is already pre-defined in the execution problem. Market makers, on the other hand, place limit orders on both sides of the order book and set out to maximize profits from capturing the spread, while minimizing the risk of inventory accumulation and adverse selection. We summarize using the term "RL for trading" such tasks which maximize profit from taking directional bets in the market. This is a hard problem for RL to solve as the space of potential trading strategies is large, leading to potentially many local optima in the loss landscape, and actionable directional market forecasts are notoriously difficult due to arbitrage in the market.

The work of Kearns and Nevmyvaka (2013) is an early study of RL for market microstructure tasks, including trade execution and predicting price movements. While the authors achieve some predictive power of directional price moves, forecasts are determined to be too erroneous for profitable trading. The most similar work to ours is Briola et al. (2021) that provides the first end-to-end DRL framework for high-frequency trading, using PPO (Schulman et al., 2017) to trade Intel stock. To model

price impact, Briola et al. (2021) use an approximation, moving prices proportionately to the square-root of traded volume. The action space is essentially limited to market orders, so there is no decision made on limit prices. The trained policy is capable of producing a profitable trading strategy on the evaluated 20 test days. However, this is not compared to baseline strategies and the resulting performance is not statistically tested for significance. In contrast, we consider a larger action space, allowing for the placement of limit orders at different prices, thereby potentially lowering transaction costs of the learned HFT strategy. For a broader survey of deep RL (DRL) for trading, including portfolio optimization, model-based and hierarchical RL approaches the reader is referred to Kumar (2020).
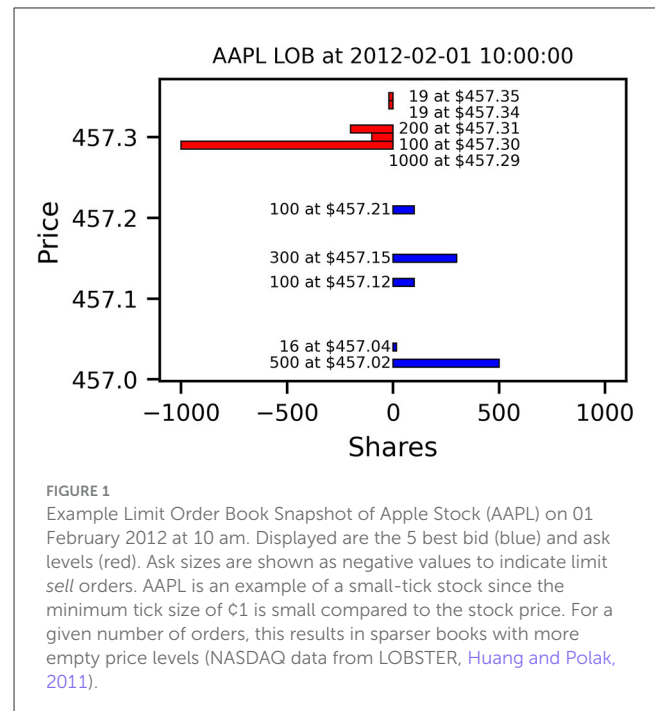
One strand of the literature formulates trading strategies in order-driven markets, such as LOBs, as optimal stochastic control problems, which can often even be solved analytically. A seminal work in this tradition is Almgren and Chriss (2001), which solves a simple optimal execution problem. More recently, Cartea et al. (2018) use order book imbalance, i.e., the relative difference in volume between buy and sell limit orders, to forecast the direction of subsequent market orders and price moves. They find that utilizing this metric can thereby improve the performance of trading strategies. In addition to market orders, LOB imbalance has also been found to be predictive of limit order arrivals, and illegal manipulation of the LOB imbalance by *spoofing* can be a profitable strategy (Cartea et al., 2020). Stochastic models, assuming temporary and permanent price impact functions have also found that using order flow information can reduce trading costs when trading multiple assets (Cartea and Jaimungal, 2016; Cartea et al., 2019). In contrast to the stochastic modeling literature, we employ a purely data-driven approach using simulation. This allows us to make fewer assumptions about market dynamics, such as specific functional forms and model parameters. The stochasticity of the market is captured in large samples of concrete data realizations.

3. Background

3.1. Limit order book data

Limit order books (LOBs) are one of the most popular financial market mechanisms used by exchanges around the world (Gould et al., 2013). Market participants submit limit buy or sell orders, specifying a maximum (minimum) price at which they are wiling to buy (sell), and the size of the order. The exchange's LOB then keeps track of unfilled limit orders on the buy side (bids) and the sell side (asks). If an incoming order is marketable, i.e., there are open orders on the opposing side of the order book at acceptable prices, the order is matched immediately, thereby removing liquidity from the book. The most popular matching prioritization scheme is price-time priority. Here, limit orders are matched first based on price, starting with the most favorable price for the incoming order, and then based on arrival time, starting with the oldest resting limit order in the book, at each price level.

Figure 1 shows an example snapshot of an Apple LOB, traded at the NASDAQ exchange, on Wednesday 01 February 2012 at 10 am. Shown are the best 5 price levels on the bid and ask side and the aggregated available volume at each price. In this example,

FIGURE 1
Example Limit Order Book Snapshot of Apple Stock (AAPL) on 01 February 2012 at 10 am. Displayed are the 5 best bid (blue) and ask levels (red). Ask sizes are shown as negative values to indicate limit *sell* orders. AAPL is an example of a small-tick stock since the minimum tick size of ¢1 is small compared to the stock price. For a given number of orders, this results in sparser books with more empty price levels (NASDAQ data from LOBSTER, Huang and Polak, 2011).

the best bid lies at $457.21 for 100 shares, which is the maximum instantaneous price a potential seller would be able to trade at, for example using a market order. Conversely, a potential buyer could receive up to 1,000 shares at a price of $457.29. Trading larger quantities on either side would use up all volume at the best price and consume additional liquidity at deeper levels. Submitting a buy limit order at a price below the best ask would not be marketable immediately and instead enter the LOB as new volume on the bid side. For a more comprehensive review of limit order book dynamics and pertaining models, we refer the reader to Gould et al. (2013).

In this paper, we consider equity limit order book data from the NASDAQ exchange (Huang and Polak, 2011), which also uses a price-time priority prioritization. Our market simulator keeps track of the state of the LOB by replaying historical message data, consisting of new incoming limit orders, order cancellations, or modifications. The RL agent can then inject new messages into the order flow and thereby, change the LOB state from its observed historical state.

Our simulator reconstructs LOB dynamics from message data, so every marketable order takes liquidity from the book and thus has a direct price impact. Beyond that, we make no further assumptions on permanent market impact or reactions of other agents in the market, which we leave to future work.

3.2. Deep reinforcement learning

We model the trader's problem as a Markov Decision Process (MDP) (Bellman, 1957; Puterman, 1990), described by the tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, r, \gamma \rangle$. $\mathcal{S}$ denotes a state space, $\mathcal{A}$ an action space, $\mathcal{T}$ a stochastic transition function, r a reward function, and γ a discount factor. Observing the current environment state $s_t \in \mathcal{S}$ at time

t, the trader takes action $a_t \in \mathcal{A}$, which causes the environment to transition state according to the stochastic transition function $\mathcal{T}(s_{t+1}|s_t, a_t)$. After transitioning from s_t to s_{t+1}, the agent receives a reward $r_{t+1} = r(s_t, a_t, s_{t+1})$.

Solving the MDP amounts to finding an optimal policy $\pi^*: \mathcal{S} \rightarrow \mathcal{A}$, which maximizes the discounted expected sum of future rewards $\sum_{i=t+1}^{T} \gamma^{i-t} \mathbb{E}_t r_i$ between current time t and terminal time T, given a discount factor $\gamma \in (0, 1]$. As the transition kernel $\mathcal{T}$ is unknown, we use reinforcement learning (RL) to learn an optimal policy from observed trajectories of state-action transitions. RL algorithms fall broadly within two categories: value-based methods, which learn representations of value functions, and policy-based methods, which learn explicit representations of the agent's policy.[1] In this paper, we are using a value-based RL algorithm, based on *Q-learning*, which explicitly approximates the action-value function Q^* for the optimal policy π^*. The action-value function for a given policy π is defined recursively as $Q^\pi(s, a) = \mathbb{E}_t r_{t+1} + \gamma \max_{a'} Q^\pi(s_{t+1}, a')$. One benefit of this class of algorithms is that they are *off-policy* learners, which means that they approximate the optimal value function Q^* using transitions sampled from the environment using a, potentially sub-optimal, exploratory policy π. This allows for computational efficiency due to asynchronous sampling and learning steps, as described in the next section. Modern Deep RL algorithms, such as DQN (Mnih et al., 2013), use neural networks as function approximators, in this case, of the Q-function. This way algorithms make use of the generalization abilities provided by deep learning, necessitated by large or continuous state spaces. Network parameters are updated using *temporal difference learning* with gradient-based optimizers, such as stochastic gradient descent or the popular Adam optimizer (Kingma and Ba, 2014). For a comprehensive treatment of RL, we refer the interested reader to Sutton and Barto (2018), and for a survey of recent progress in RL for financial applications to Hambly et al. (2023).

3.3. Double DQN with distributed experience replay

We use Deep Double Q-learning (Van Hasselt et al., 2016) with a dueling network architecture (Wang et al., 2016) to approximate the optimal Q-function $Q^*(s, a) = \mathbb{E}[r_{t+1} + \gamma \max_{a'} Q^*(s_{t+1}, a')|a_t = a, s_t = s]$. To speed up the learning process we employ the *APEX* training architecture (Horgan et al., 2018), which combines asynchronous experience sampling using parallel environments with off-policy learning from experience replay buffers. Every episode i results in an experience trajectory $\tau_i = \{s_t, a_t\}_{t=1}^{T}$, many of which are sampled from parallel environment instances and are then stored in the replay buffer. The environment sampling is done asynchronously using parallel processes running on CPUs. Experience data from the buffer is then sampled randomly and batched to perform a policy improvement step of the Q-network on the GPU. Prioritized sampling from the experience buffer has proven to degrade performance in our noisy

problem setting, hence we are sampling uniformly from the buffer.[2] After a sufficient number of training steps, the new policy is then copied to every CPU worker to update the behavioral policy.

Double Q-learning (Hasselt, 2010; Van Hasselt et al., 2016) stabilizes the learning process by keeping separate Q-network weights for action selection (main network) and action validation (target network). The target network weights are then updated gradually in the direction of the main network's weight every few iterations. Classical Q-learning without a separate target network can be unstable due to a positive bias introduced by the max operator in the definition of the Q-function, leading to exploding Q-values during training. The dueling network architecture (Wang et al., 2016) additionally uses two separate network branches (for both main and target Q-networks). One branch estimates the value function $V(s) = \max_a Q(s, a)$, while the other estimates the advantage function $A(s, a) = Q(s, a) - V(s)$. The benefit of this architecture choice lies therein that the advantage of individual actions in some states might be irrelevant, and the state value, which can be learnt more easily, suffices for an action-value approximation. This leads to faster policy convergence during training.

4. Framework

4.1. Artificial price signal

The artificial directional price signal $d_t \in \Delta^2 = \{x \in \mathbb{R}^3 : x_1 + x_2 + x_3 = 1, \ x_i \geq 0 \text{ for } i = 1, 2, 3\}$ the agent receives is modeled as a discrete probability distribution over 3 classes, corresponding to the averaged mid-quote price decreasing, remaining stable, or increasing over a fixed future time horizon of $h \in \mathbb{N}_+$ seconds. To achieve realistic levels of temporal stability of the signal process, d_t is an exponentially weighted average, with persistence coefficient $\phi \in (0, 1)$, of Dirichlet random variables ϵ_t. The Dirichlet parameters α depend on the realized smoothed future return $\overline{r_{t+h}}$, specifically on whether the return lies within a neighborhood of size k around zero, or above or below. Thus we have:

$$
\begin{aligned}
d_t &= \phi d_{t-1} + (1 - \phi)\epsilon_t \\
\epsilon_t &= \text{Dirichlet}\left(\alpha(\overline{r_{t+h}})\right) \\
\overline{r_{t+h}} &= \frac{\overline{p_{t+h}} - p_t}{p_t} \quad \text{where} \quad \overline{p_{t+h}} = \frac{1}{h}\sum_{i=1}^{h} p_{t+i}
\end{aligned} \tag{1}
$$

and prices p_t refer to the mid-quote price at time t. The Dirichlet distribution is parametrized, so that, in expectation, the signal d_t updates in the direction of future returns, where a^H and a^L determine the variance of the signal. The Dirichlet parameter vector

---

1 Actor-critic algorithms fall between the two as they keep explicit representations of both policy (actor) and value functions (critic).

---

2   In many application domains prioritized sampling, whereby we resample instances more frequently where the model initially performs poorly tends to aide learning. However, in our low signal-to-noise application domain, we noted poor performance. Investigating the matter, we found that prioritized sampling caused more frequent resampling of highly noisy instances where learning was particularly difficult, thus degrading performance.

is thus:

$$\alpha(r_{t+h}) = \begin{cases} (a^H, a^L, a^L) & \text{if } \overline{r_{t+h}} < -k \\ (a^L, a^H, a^L) & \text{if } -k \leq \overline{r_{t+h}} < k \\ (a^L, a^L, a^H) & \text{if } k \leq \overline{r_{t+h}}. \end{cases} \quad (2)$$

4.2. RL problem specification

At each time step t, the agent receives a new state observation s_t. s_t consists of the time left in the current episode $T - t$ given the episode's duration of T, the agent's cash balance C_t, stock inventory X_t, the directional signal $d_t \in \Delta^2$, encoded as probabilities of prices decreasing, remaining approximately constant, or increasing; and price and volume quantities for the best bid and ask (level 1), including the agent's own volume posted at bid and ask: $o_{b,t}$ and $o_{a,t}$ respectively. In addition to the most current observable variables at time t, the agent also observes a history of the previous l values, which are updated whenever there is an observed change in the LOB. Putting all this together, we obtain the following state observation:

$$s_t = \begin{pmatrix} T - u \\ C_u \\ X_u \\ (d_u^1, d_u^2, d_u^3)' \\ (p_{a,u}, v_{a,u}, o_{a,u}, p_{b,u}, v_{b,u}, o_{b,u})' \end{pmatrix}_{u=\{t-l,\dots,t\}}.$$

After receiving the state observation, the agent then chooses an action a_t. It can place a buy or sell limit order of a single share at bid, mid-quote, or ask price; or do nothing and advance to the next time step. Actions, which would immediately result in positions outside the allowed inventory constraints $[pos_{min}, pos_{max}]$ are disallowed and do not trigger an order. Whenever the execution of a resting limit order takes the inventory outside the allowed constraints, a market order in the opposing direction is triggered to reduce the position back to pos_{min} for short positions or pos_{max} for long positions. Hence, we define

$$a_t \in \mathcal{A} = (\{-1, 1\} \times \{-1, 0, 1\}) \cup \{skip\}$$

so that in total there are 7 discrete actions available, three levels for both buy and sell orders, and a skip action. For the six actions besides the "skip" action, the first dimension encodes the trading direction (sell or buy) and the second dimension the price level (bid, mid-price, or ask). For example, $a = (1, 0)$ describes the action to place a buy order at the mid price, and $a = (-1, 1)$ a sell order at best ask. Rewards R_{t+1} consist of a convex combination of a profit-and-loss-based reward R_{t+1}^{pnl} and a directional reward R_{t+1}^{dir}. R_{t+1}^{pnl} is the log return of the agent's mark-to-market portfolio value M_t, encompassing cash and the current inventory value, marked at the mid-price. The benefit of log-returns is that they are additive over time, rather than multiplicative like gross returns, so that, without discounting ($\gamma = 1$) the total profit-and-loss return $\sum_{s=t+1}^{T} R_s^{pnl} = M_T - M_t$. The directional reward term R_{t+1}^{dir} incentivizes the agent to hold inventory in the direction of the signal and penalizes the agent for inventory positions opposing the signal direction. The size

of the directional reward can be scaled by the parameter $\kappa > 0$. R_{t+1}^{dir} is positive if the positive prediction has a higher score than the negative ($d_{t,3} > d_{t,1}$) and the current inventory is positive; or if $d_{t,3} < d_{t,1}$ and $X_t < 0$. Further, if the signal $[-1, 0, 1] \cdot d_t$ has an opposite sign than inventory X_t, R_{t+1}^{dir} is negative. This can be summarized as follows:

$$\begin{aligned} \text{Mark-to-Market Value} \quad & M_t = C_t + X_t p_t^m \\ & \Delta M_t = \Delta C_t + X_{t-1}\Delta p_t^m + \Delta x_t p_t^m \\ \text{PnL Reward} \quad & R_{t+1}^{pnl} = \ln(M_t) - \ln(M_{t-1}) \quad (3) \\ \text{Directional Reward} \quad & R_{t+1}^{dir} = \kappa[-1, 0, 1] \cdot d_t X_t \\ \text{Total Reward} \quad & r_{t+1} = w^{dir} R_{t+1}^{dir} + (1 - w^{dir}) R_{t+1}^{pnl} \end{aligned}$$

The weight on the directional reward $w^{dir} \in [0, 1)$ is reduced every learning step by a factor $\psi \in (0, 1)$,

$$w^{dir} \leftarrow \psi w^{dir}$$

so that initially the agent quickly learns to trade in the signal direction. Over the course of the learning process, R_t^{pnl} becomes dominant and the agent maximizes its mark-to-market profits.

5. Experimental results

We train all RL policies using the problem setup discussed in Section 4.2 on 4.5 months of Apple (AAPL) limit order book data (2012-01-01 to 2012-05-16) and evaluate performance on 1.5 months of out-of-sample data (2012-05-17 to 2012-06-31). We only use the first hour of every trading day (09:30 to 10:30) as the opening hour exhibits higher-than-average trading volume and price moves. Each hour of the data corresponds to a single RL episode. After analyzing the results, we also performed a robustness check by repeating the training and analysis pipeline on more recent AAPL data from 2022. Results are reported in Section 5.2 and confirm the main conclusions based on earlier data.

Our neural network architecture consists of 3 feed-forward layers, followed by an LSTM layer, for both the value- and advantage stream of the dueling architecture. The LSTM layer allows the agent to efficiently learn a memory-based policy with observations including 100 LOB states.

We compare the resulting learned RL policies to a baseline trading algorithm, which receives the same artificially perturbed high-frequency signal of future mid-prices. The baseline policy trades aggressively by crossing the spread whenever the signal indicates a directional price move up or down until the inventory constraint is reached. The signal direction in the baseline algorithm is determined as the prediction class with the highest score (down, neutral, or up). When the signal changes from up or down to neutral, indicating no immediate expected price move, the baseline strategy places a passive order to slowly reduce position size until the inventory is cleared. This heuristic utilizes the same action space as the RL agent and yielded better performance than trading using only passive orders
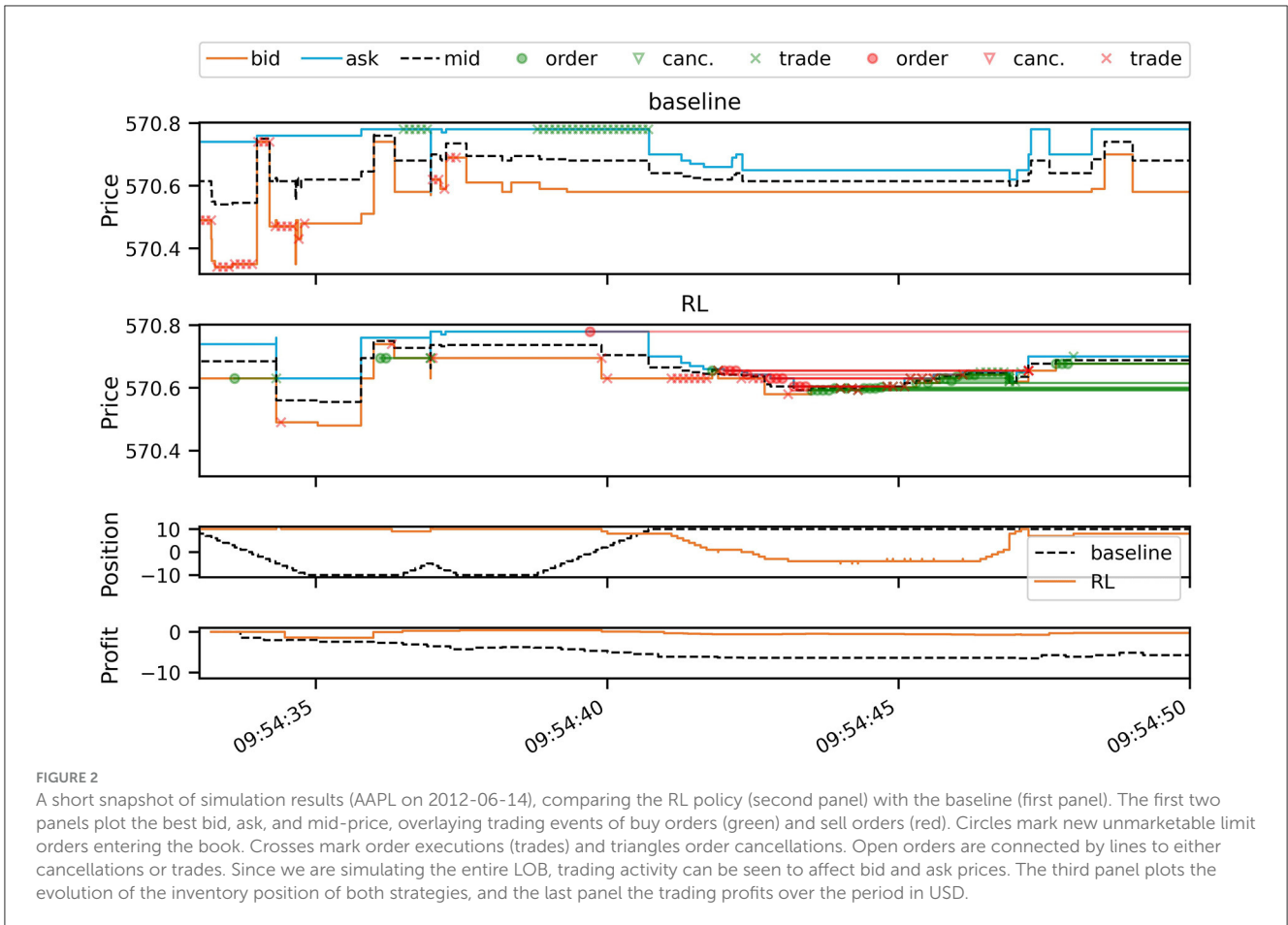
FIGURE 2
A short snapshot of simulation results (AAPL on 2012-06-14), comparing the RL policy (second panel) with the baseline (first panel). The first two panels plot the best bid, ask, and mid-price, overlaying trading events of buy orders (green) and sell orders (red). Circles mark new unmarketable limit orders entering the book. Crosses mark order executions (trades) and triangles order cancellations. Open orders are connected by lines to either cancellations or trades. Since we are simulating the entire LOB, trading activity can be seen to affect bid and ask prices. The third panel plots the evolution of the inventory position of both strategies, and the last panel the trading profits over the period in USD.

(placed at the near touch), or only aggressive orders (at the far touch).
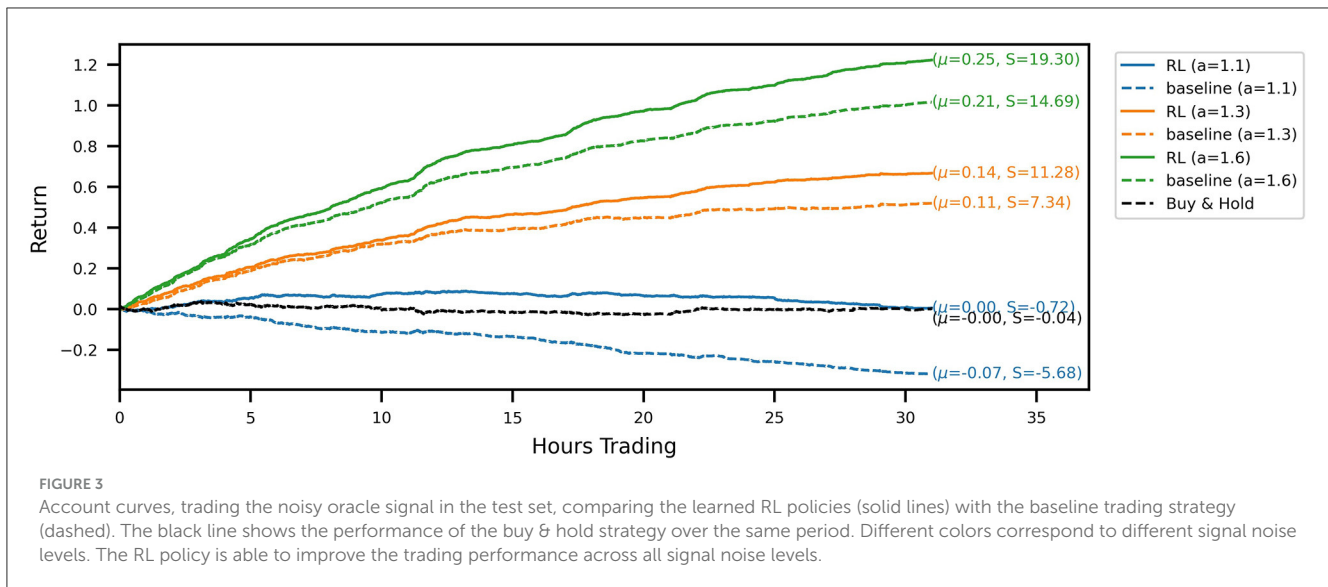
Figure 2 plots a 17 second simulation window from the test period, comparing the simulated baseline strategy with the RL strategy. It can be seen that prices in the LOB are affected by the trading activity as both strategies inject new order flow into the market, in addition to the historical orders, thereby consuming or adding liquidity at the best bid and ask. During the plotted period, the baseline strategy incurs small losses due to the signal switching between predicting decreasing and increasing future prices. This causes the baseline strategy to trade aggressively, paying the spread with every trade. The RL strategy, on the other hand, navigates this difficult period better by trading more passively out of its long position, and again when building up a new position. Especially in the second half of the depicted time period, the RL strategy adds a large number of passive buy orders (green circles in the second panel of Figure 2). This is shown by the green straight lines, which connect the orders to their execution or cancellation, some of which occur after the depicted period.

5.1. Oracle signal

The RL agent receives a noisy oracle signal of the mean return $h = 10$ seconds into the future (see Equation 1). It chooses an action every 0.1s, allowing a sufficiently quick build-up of long

or short positions using repeated limit orders of single stocks. The algorithm is constrained to keep the stock inventory within bounds of $[pos_{min}, pos_{max}] = [-10, 10]$. To change the amount of noise in the signal, we vary the a^H parameter of the Dirichlet distribution, keeping $a^L = 1$ constant in all cases. To keep the notation simple, we hence drop the H superscript and refer to the variable Dirichlet parameter a^H simply as a. We consider three different noise levels, parametrising the Dirichlet distribution with $a = 1.6$ (low noise), $a = 1.3$ (mid noise), and $a = 1.1$ (high noise). A fixed return classification threshold $k = 4 \cdot 10^{-5}$ was chosen to achieve good performance of the baseline algorithm, placing around 85% of observations in the up or down category. The signal process persistence parameter is set to $\phi = 0.9$.

Out-of-sample trading performance is visualized by the account curves in Figure 3. The curves show the evolution of the portfolio value for a chronological evaluation of all test episodes. Every account curve shows the mean episodic log-return μ and corresponding Sharpe ratio S next to it. We show that all RL-derived policies are able to outperform their respective baseline strategies for the three noise levels investigated. Over the 31 test episodes, the cumulative RL algorithm out-performance over the baseline strategy ranges between 14.8 ($a = 1.3$) and 32.2 ($a = 1.1$) percentage points (and 20.7 for $a = 1.6$). In the case of the signal with the lowest signal-to-noise ratio ($a = 1.1$), for which the baseline strategy incurs a loss for the test period, the RL agent has learned a trading strategy with an approximately zero mean return.

FIGURE 3
Account curves, trading the noisy oracle signal in the test set, comparing the learned RL policies (solid lines) with the baseline trading strategy (dashed). The black line shows the performance of the buy & hold strategy over the same period. Different colors correspond to different signal noise levels. The RL policy is able to improve the trading performance across all signal noise levels.

Temporarily, the strategy even produces positive gains. Overall, it produces a sufficiently strong performance to not lose money while still trading actively and incurring transaction costs. Compared to a buy-and-hold strategy over the same time period, the noisy RL strategy similarly produces temporary out-performance, with both account curves ending up flat with a return around zero. Inspecting Sharpe ratios, we find that using RL to optimize the trading strategy is able to increase Sharpe ratios significantly. The increase in returns of the RL strategies is hence not simply explained by taking on more market risk.

Figure 4A compares the mean return between the buy & hold, baseline, and RL policies for all out-of-sample episodes across the three noise levels. A single dashed gray line connects the return for a single test episode across the three trading strategies: buy & hold, baseline, and the RL policy. The solid blue lines representing the mean return across all episodes. Error bars represent the 95% bootstrapped confidence intervals for the means. Testing for the significance of the differences between RL and baseline returns across all episodes (t-test) is statistically significant ($p \ll 0.1$) for all noise levels. Differences in Sharpe ratios are similarly significant. We can thus conclude that the high frequency trading strategies learned by RL outperform our baseline strategy for all levels of noise we have considered.

It is also informative to compare the amount of trading activity between the baseline and RL strategies (see Figure 4B). The baseline turnover decreases with an increasing signal-to-noise ratio (higher a), as the signal remains more stable over time, resulting in fewer trades. In contrast, the turnover of the RL trading agent increases with a higher signal-to-noise ratio, suggesting that the agent learns to trust the signal more and reflecting that higher transaction costs, resulting from the higher trading activity, can be sustained, given a higher quality signal. In the high noise case ($a = 1.1$), the RL agent learns to reduce trading activity relative to the other RL strategies, thereby essentially filtering the signal. The turnover is high in all cases due to the high frequency of the signal and the fact that we are only trading a small inventory. Nonetheless, performance is

calculated net of spread-based transaction costs as our simulator adequately accounts for the execution of individual orders.

Table 1 lists action statistics for all RL policies, including how often actions are skipped, and the price levels at which limit orders are placed, grouped by buy and sell orders. With the least informative signal, the strategy almost exclusively uses marketable limit orders, with buy orders being placed at the bid and sell orders at the ask price. With better signals being available ($a = 1.3$ and $a = 1.6$), buy orders are more often placed at the mid-quote price, thereby trading less aggressively and saving on transaction costs. Overall, the strategies trained on different signals all place the majority of sell orders at the best bid price, with the amount of skipped actions varying considerably across the signals.

5.2. Robustness evaluation on recent data

To evaluate our results on more recent LOB data, we also trained RL policies using 4.5 months of AAPL LOB data from the second half of 2022 (2022-07-01 to 2022-11-13) and evaluated on 1.5 months of held-out test data (2022-11-14 to 2022-12-31). Figure 5 shows account curves over the evaluation period for all 3 noise levels. For low ($a = 1.6$) and medium ($a = 1.3$) noise levels, the RL policies, again, beat the baselines by a significant margin and increase profits significantly. For a low-quality signal with high noise levels ($a = 1.1$), the RL policy performs similarly to the losing baseline strategy and also doesn't make a profit. Interestingly, the strategy also does not learn to stop trading altogether, which would represent a superior policy in this case. This could be due to a common problem with RL: effective exploration. The local optimum of not trading could not be found using an epsilon-greedy exploration policy in this case. Overall, even though macroeconomic and financial market conditions in 2022 differed markedly from 2012, our results support
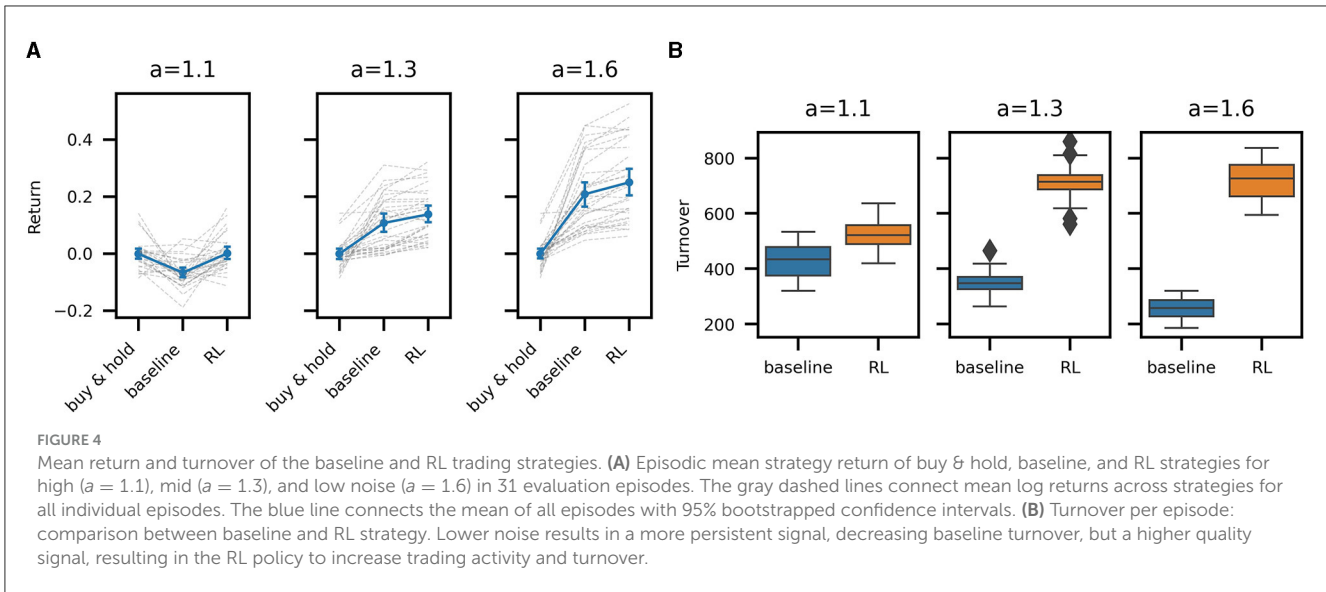
FIGURE 4
Mean return and turnover of the baseline and RL trading strategies. **(A)** Episodic mean strategy return of buy & hold, baseline, and RL strategies for high ($a = 1.1$), mid ($a = 1.3$), and low noise ($a = 1.6$) in 31 evaluation episodes. The gray dashed lines connect mean log returns across strategies for all individual episodes. The blue line connects the mean of all episodes with 95% bootstrapped confidence intervals. **(B)** Turnover per episode: comparison between baseline and RL strategy. Lower noise results in a more persistent signal, decreasing baseline turnover, but a higher quality signal, resulting in the RL policy to increase trading activity and turnover.

TABLE 1 Actions taken by RL policy for the three different noise levels: the first row shows how often the policy chooses the "skip" action.
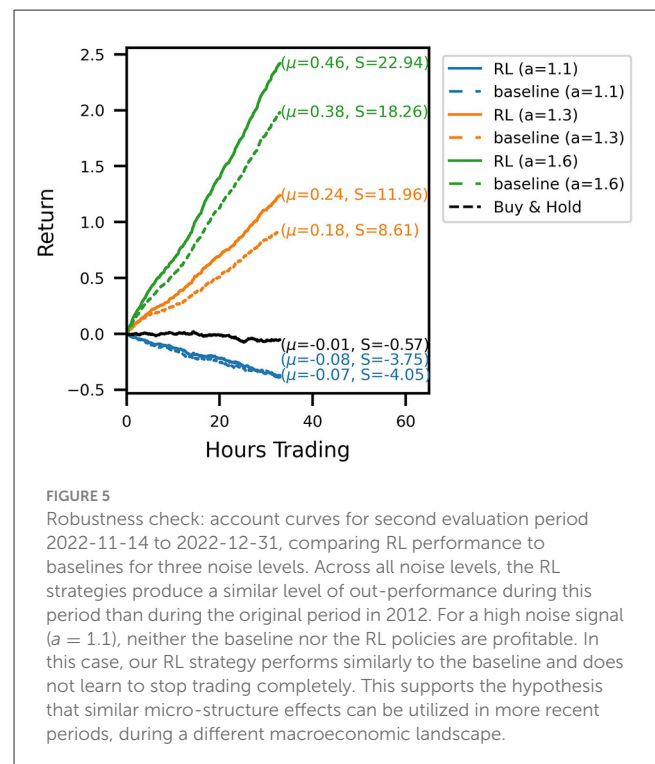
	$a = 1.1$	$a = 1.3$	$a = 1.6$
Action skipped [%]	24.5	43.8	7.8
Sell levels (bid, mid, ask) [%]	(95.4, 3.1, 1.5)	(94.6, 2.8, 2.65)	(97.2, 1.9, 0.9)
Buy levels (bid, mid, ask) [%]	(1.1, 1.3, 97.5)	(1.6, 52.9, 45.5)	(1.7, 13.0, 85.3)

Not choosing this action does however not necessarily result in an order being placed, e.g., if inventory constraints are binding. The last two rows show the relative proportion of limit order placement levels for sell orders, and buy orders, respectively.

the conclusion that the RL policies can utilize similar micro-structure effects in both periods to improve the execution of a trading strategy based on a price forecast signal.

6. Conclusions

Using Deep Double Dueling Q-learning with asynchronous experience replay, a state-of-the-art off-policy reinforcement learning algorithm, we train a limit order trading strategy in an environment using historic market-by-order (MBO) exchange message data. For this purpose we develop an RL environment based on the ABIDES (Byrd et al., 2020a) market simulator, which reconstructs order book states dynamically from MBO data. Observing an artificial high-frequency signal of the mean return over the following 10 seconds, the RL policy successfully transforms a directional signal into a limit order trading strategy. The policies acquired by RL outperform our baseline trading algorithm, which places marketable limit orders to trade into positions and passive limit orders to exit positions, both in terms of mean return and Sharpe ratio. We investigate the effect of different levels of noise in the alpha signal on the RL performance. Unsurprisingly, more accurate signals lead to higher trading returns but we also find that

FIGURE 5
Robustness check: account curves for second evaluation period 2022-11-14 to 2022-12-31, comparing RL performance to baselines for three noise levels. Across all noise levels, the RL strategies produce a similar level of out-performance during this period than during the original period in 2012. For a high noise signal ($a = 1.1$), neither the baseline nor the RL policies are profitable. In this case, our RL strategy performs similarly to the baseline and does not learn to stop trading completely. This supports the hypothesis that similar micro-structure effects can be utilized in more recent periods, during a different macroeconomic landscape.

RL provides a similar added benefit to trading performance across all noise levels investigated.

The task of converting high-frequency forecasts into tradeable and profitable strategies is difficult to solve as transaction costs, due to high portfolio turnover, can have a prohibitively large impact on the bottom line profits. We suggest that RL can be a useful tool to perform this translational role and learn optimal strategies for a specific signal and market combination. We have shown that tailoring strategies in this way can significantly improve performance, and eliminates the need for manually fine-tuning execution strategies for different markets and signals. For practical

applications, multiple different signals could even be combined into a single observation space. That way the problem of integrating different forecasts into a single coherent trading strategy could be directly integrated into the RL problem.

A difficulty for all data-driven simulations of trading strategies relying on market micro-structure is accurately estimating market impact. We address this partially by injecting new orders into historical order streams, thereby removing liquidity from the LOB. If liquidity at the best price is used up, this would automatically increase transaction costs by consuming deeper levels of the book. This mechanism accurately models temporary *direct* market impact, but cannot take account of the *indirect* or permanent component due to other market participants' reactions to our orders. By only allowing trading a single stock each time step, we posed the problem in a way to minimize the potential effect indirect market impact would have on the performance of the RL strategy. The strategy trades small quantities on both sides of the book without accumulating large inventories. Such trading strategies are *capacity* constrained by the volume available in the book at any time and belong to a different class than *impact* constrained strategies, which build up large inventories by successively submitting orders in only one direction. Furthermore, we measure trading performance relative to a baseline strategy, which makes the same assumptions on market impact. However, accurately modeling the full market impact of high-frequency trading in LOB markets in a data-driven approach is an interesting direction for future research and would allow evaluating strategies with larger order sizes. Recent attempts in this vein have used agent-based models (Byrd et al., 2020b) or generative models (Coletta et al., 2021, 2022, 2023).

We chose to focus our investigations in this paper on AAPL stock as a challenging test case of a small-tick stock, i.e., one where the minimum tick size is small relative to the stock price, with a high trading volume. Showing that we can train an RL agent to improve the profitability of an alpha signal in this example, indicates that similar performance improvements could be possible in larger-tick stocks with less trading activity. Although results are limited to a single company due to computational constraints, functional relationships in the micro-structure of the market have been found to be stable over time and across companies in prior work. In a large-scale study of order flow in US equity LOBs (Sirignano and Cont, 2019) found a *universal* and *stationary* relationship between order flow and price changes, driven by robust underlying supply and demand dynamics. Similarly, supervized training of deep neural networks to predict the mid-price direction a few ticks into the future has been shown to work for a wide range of stocks (Zhang et al., 2019). In contrast to lower-frequency trading strategies, whose performance often varies with market conditions, such as the presence of price trends or macroeconomic conditions, high-frequency strategies don't suffer the same degree of variability. Nonetheless, a systematic investigation of potential changes in LOB dynamics due to crisis periods or rare events could be an interesting avenue for future research.

While we here show an interesting use case of RL in limit order book markets, we also want to motivate the need for further research in this area. There are many years of high-frequency market data available, which ought to be utilized to make further progress in LOB-based tasks and improve RL in noisy environments. This, together with the newest type of neural network architectures, such as attention-based transformers (Vaswani et al., 2017; Child et al., 2019), enables learning tasks in LOB environments directly from raw data with even better performance. For the task we have considered in this paper, future research could enlarge the action space, allowing for the placement of limit orders deeper into the book and larger order sizes. Allowing for larger sizes however would require a realistic model of market impact, considering the reaction of other market participants.

Data availability statement

The data analyzed in this study is subject to the following licenses/restrictions: Lobster data (https://lobsterdata.com/index.php) is available by subscription from the vendor for research purposes. Requests to access these datasets should be directed to https://lobsterdata.com/info/HowToJoin.php.

Author contributions

PN performed the experiments and wrote the manuscript with input from all authors. All authors devised the project, the conceptual ideas, and the design of the experiments. All authors contributed to the article and approved the submitted version.

Funding

Acknowledgments

Conflict of interest

SZ was employed by Man Group.

The remaining authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Publisher's note

All claims expressed in this article are solely those of the authors and do not necessarily represent those of

their affiliated organizations, or those of the publisher, the editors and the reviewers. Any product that may be evaluated in this article, or claim that may be made by its manufacturer, is not guaranteed or endorsed by the publisher.

Supplementary material

The Supplementary Material for this article can be found online at: https://www.frontiersin.org/articles/10.3389/frai.2023.1151003/full#supplementary-material

References

Abernethy, J. D., and Kale, S. (2013). "Adaptive market making via online learning," in *Advances in Neural Information Processing Systems*, 26.

Almgren, R., and Chriss, N. (2001). Optimal execution of portfolio transactions. *J. Risk* 3, 5–40. doi: 10.21314/JOR.2001.041

Amrouni, S., Moulin, A., Vann, J., Vyetrenko, S., Balch, T., and Veloso, M. (2021). "ABIDES-gym: gym environments for multi-agent discrete event simulation and application to financial markets," in *Proceedings of the Second ACM International Conference on AI in Finance* (ACM) 1–9. doi: 10.1145/3490354.3494433

Bellman, R. (1957). A Markovian decision process. *J. Math. Mech.* 6, 679–684. doi: 10.1512/iumj.1957.6.56038

Briola, A., Turiel, J., Marcaccioli, R., and Aste, T. (2021). Deep reinforcement learning for active high frequency trading. *arXiv preprint arXiv:2101.07107*.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., et al. (2016). Openai gym. *arXiv preprint arXiv:1606.01540*.

Byrd, D., Hybinette, M., and Balch, T. H. (2020a). Abides: Towards high-fidelity market simulation for AI research. arXiv preprint arXiv:1904.12066.

Byrd, D., Hybinette, M., and Balch, T. H. (2020b). "Abides: Towards high-fidelity multi-agent market simulation," in *Proceedings of the 2020 ACM SIGSIM Conference on Principles of Advanced Discrete Simulation*, 11–22. doi: 10.1145/3384441.3395986

Cartea, A., Donnelly, R., and Jaimungal, S. (2018). Enhancing trading strategies with order book signals. *Appl. Mathem. Finance* 25, 1–35. doi: 10.1080/1350486X.2018.1434009

Cartea, Á., Gan, L., and Jaimungal, S. (2019). Trading co-integrated assets with price impact. *Mathem. Finan.* 29, 542–567. doi: 10.1111/mafi.12181

Cartea, Á., and Jaimungal, S. (2016). Incorporating order-flow into optimal execution. *Mathem. Finan. Econ.* 10, 339–364. doi: 10.1007/s11579-016-0162-z

Cartea, Á., Jaimungal, S., and Wang, Y. (2020). Spoofing and price manipulation in order-driven markets. *Appl. Mathem. Finan.* 27, 67–98. doi: 10.1080/1350486X.2020.1726783

Child, R., Gray, S., Radford, A., and Sutskever, I. (2019). Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*.

Coletta, A., Jerome, J., Savani, R., and Vyetrenko, S. (2023). Conditional generators for limit order book environments: Explainability, challenges, and robustness. *arXiv preprint arXiv:2306.12806*.

Coletta, A., Moulin, A., Vyetrenko, S., and Balch, T. (2022). "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*, 428–436. doi: 10.1145/3533271.3561753

Coletta, A., Prata, M., Conti, M., Mercanti, E., Bartolini, N., Moulin, A., et al. (2021). "Towards realistic market simulations: a generative adversarial networks approach," in *Proceedings of the Second ACM International Conference on AI in Finance*, 1–9. doi: 10.1145/3490354.3494411

Dabérius, K., Granat, E., and Karlsson, P. (2019). *Deep Execution-Value and Policy Based Reinforcement Learning for Trading and Beating Market Benchmarks*. Rochester, NY: Social Science Research Network.

Gould, M. D., Porter, M. A., Williams, S., McDonald, M., Fenn, D. J., and Howison, S. D. (2013). Limit order books. *Quant. Finan.* 13, 1709–1742. doi: 10.1080/14697688.2013.803148

Hambly, B., Xu, R., and Yang, H. (2023). Recent advances in reinforcement learning in finance. *Mathem. Finan.* 33, 437–503. doi: 10.1111/mafi.12382

Harris, L. (2003). *Trading and Exchanges: Market Microstructure for Practitioners*. New York, NY: OUP USA.

Hasselt, H. (2010). "Double Q-learning," in *Advances in Neural Information Processing Systems*, 23.

Horgan, D., Quan, J., Budden, D., Barth-Maron, G., Hessel, M., Hasselt, H., et al. (2018). Distributed prioritized experience replay. *arXiv preprint arXiv:1803.00933*.

Huang, R., and Polak, T. (2011). *Lobster: Limit Order Book Reconstruction System*. Available online at: SSRN 1977207.

Karpe, M., Fang, J., Ma, Z., and Wang, C. (2020). "Multi-agent reinforcement learning in a realistic limit order book market simulation," in *Proceedings of the First ACM International Conference on AI in Finance* (ACM). doi: 10.1145/3383455.3422570

Kearns, M., and Nevmyvaka, Y. (2013). "Machine learning for market microstructure and high frequency trading," in *High Frequency Trading: New Realities for Traders, Markets, and Regulators*, eds. D. Easley, L. de Prado, M. Mailoc, and M. O'Hara (London, UK: Risk Books).

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

Kolm, P. N., Turiel, J., and Westray, N. (2021). *Deep Order Flow Imbalance: Extracting Alpha at Multiple Horizons From the Limit Order Book*. Available online at: SSRN 3900141.

Kumar, P. (2020). "Deep reinforcement learning for market making," in *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*, 1892–1894.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., et al. (2013). Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.

Mordor Intelligence (2022). *Algorithmic trading market share, size trading statistics*. Available online at: https://www.mordorintelligence.com/industry-reports/algorithmic-trading-market (accessed August 2, 2023).

Nagy, P., Powrie, J., and Zohren, S. (2023). "Machine learning for microstructure data-driven execution algorithms," in *The Handbook on AI and Big Data Applications in Investments* (CFA Institute Research Foundation).

Nevmyvaka, Y., Feng, Y., and Kearns, M. (2006). "Reinforcement learning for optimized trade execution," in *Proceedings of the 23rd International Conference on Machine Learning*, 673–680. doi: 10.1145/1143844.1143929

Ning, B., Lin, F. H. T., and Jaimungal, S. (2021). Double deep Q-learning for optimal execution. *Appl. Mathem. Finan.* 28, 361–380. doi: 10.1080/1350486X.2022.2077783

Puterman, M. L. (1990). Markov decision processes. *Handb. Operat. Res. Manage. Sci.* 2, 331–434. doi: 10.1016/S0927-0507(05)80172-0

Schnaubelt, M. (2022). Deep reinforcement learning for the optimal placement of cryptocurrency limit orders. *Eur. J. Operat. Res.* 296, 993–1006. doi: 10.1016/j.ejor.2021.04.050

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. (2017). Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Sirignano, J., and Cont, R. (2019). Universal features of price formation in financial markets: perspectives from deep learning. *Quantit. Finan.* 19, 1449–1459. doi: 10.1080/14697688.2019.1622295

Sutton, R. S., and Barto, A. G. (2018). *Reinforcement Learning: An Introduction*. London: MIT press.

Van Hasselt, H., Guez, A., and Silver, D. (2016). "Deep reinforcement learning with double Q-learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 30. doi: 10.1609/aaai.v30i1.10295

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., et al. (2017). "Attention is all you need," in *Advances in Neural Information Processing Systems*, 30.

Wang, Z., Schaul, T., Hessel, M., Hasselt, H., Lanctot, M., Freitas, N. (2016). "Dueling network architectures for deep reinforcement learning," in *International Conference on Machine Learning* (PMLR), 1995–2003.

Wei, H., Wang, Y., Mangu, L., and Decker, K. (2019). Model-based reinforcement learning for predictions and control for limit order books. *arXiv preprint arXiv:1910.03743*.

Yu, P., Lee, J. S., Kulyatin, I., Shi, Z., and Dasgupta, S. (2019). Model-based deep reinforcement learning for dynamic portfolio optimization. *arXiv preprint arXiv:1901.08740*.

Zhang, Z., Lim, B., and Zohren, S. (2021). Deep learning for market by order data. *Appl. Mathem. Finan.* 28, 79–95. doi: 10.1080/1350486X.2021.1967767

Zhang, Z., and Zohren, S. (2021). Multi-horizon forecasting for limit order books: Novel deep learning approaches and hardware acceleration using intelligent processing units. *arXiv preprint arXiv:2105.10430*.

Zhang, Z., Zohren, S., and Roberts, S. (2019). DeepLOB: Deep convolutional neural networks for limit order books. *IEEE Trans. Signal Proc.* 67, 3001–3012. doi: 10.1109/TSP.2019.2907260