
Reinforcement Learning in Limit Order Book for Cryptocurrency Trading

Jean-Sébastien Delieau, Yanruiqi Yang

Department of Mathematics, College of Management of Technology
EPFL

Lausanne, CH-1015

jean-sebastien.delineau@epfl.ch, yanruiqi.yang@epfl.ch

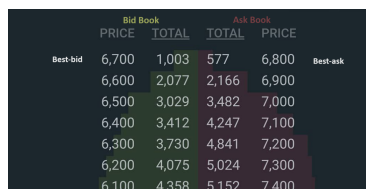
Abstract

In a model-based stock market environment, we train multiple Reinforcement Learning (RL) agents with **PPO** and **StableBaselines3** to perform in highly volatile environments like Bitcoin cryptocurrency. We implement and explain the dynamics behind adversarial RL agents. Back-testing it in different frequencies (250ms and 10ms) to evaluate its performance, we show that PPO is in most instances profitable in a mildly stochastic environment. Nevertheless, it displays limitations that we try to resolve. Our work has been pushed to the github repository.

1 Introduction

Market makers play a central role in providing liquidity and smooth trading in the financial market. They are constantly buying and selling the assets with the motivation of making profits from the price spreads. Whether they can make money in trading depends on their pricing strategies. In a market with high volatility and adversarial selection, it is always challenging to give an optimal price for the bid or ask order. Traditional market-makers construct their trading strategies based on historical data, market conditions, and volatility estimates utilizing statistical models and human expertise. In comparison, Reinforcement Learning (RL) agents can learn and adapt to unknown market regimes in real time without human intervention. In this project, we studied the Limit Order Book (LOB) dynamics and implemented training adversarial RL agents in a simulated market environment and backtesting on historical data to evaluate their performance to create optimal limit orders in different frequencies (250ms and 10ms).

The mechanism behind market making is the order book, which is a list of orders that a trading exchange uses to record the interests of the buyers and sellers of a particular financial instrument. As a specific type, a Limit Order Book (LOB) only includes limit orders. It is structured as the bid and the ask side. Buy Limit Orders (Bids) are placed by traders to buy the asset at a specified price. These orders are listed in descending order of price, with the highest bid at the top. On the other side, Sell Limit Orders (Asks) are placed by traders to sell the asset at a specified minimum price. These orders are listed in ascending order of price, with the lowest ask at the top. The difference between them is called the bid-ask spread. LOB orders are filled under a price and time priority, i.e. the first arrived lowest bid resp. the first arrived the highest ask is the first getting filled.



	Bid Book		Ask Book		
	PRICE	TOTAL	TOTAL	PRICE	
Best-bid	6,700	1,003	577	6,800	Best-ask
	6,600	2,077	2,166	6,900	
	6,500	3,029	3,482	7,000	
	6,400	3,412	4,247	7,100	
	6,300	3,730	4,841	7,200	
	6,200	4,075	5,024	7,300	
	6,100	4,358	5,152	7,400	

Figure 1: Limit order book: Bid Book & Ask Book

2 Model

Training an agent for LOB can be done in three broad approaches, a market replay approach (historical data is replayed and the learning agent interacts with it), an agent-based market simulator approach (learning agent interacts together), and a model-based approach (construct model to replicate the stock market). We use the open source `mbt_gym` (Jerome [2023]), which provides a gymnasium environment for the latter.

In the model, the agent can trade through **limit orders**; he learns to quote at any discrete time t , a bid price p_t^b and an ask price p_t^a around the "true" price S_t , which is defined as the mid-price, where the mid-price is the average sum of the best-bid and the best-ask. On the middle lower plot of Figure 2, we can see the mid-price stochastic process S_t and the agent's position p_t^a and p_t^b .

Making a limit order is the action of adding another order in the LOB, i.e. adding 1 in the volume at a given price for the bid or the ask book (cf Figure 1). Posting and removing a limit order in the order book is costless. The only obligation resulting from a limit order is a commitment to buy the security when the order is **hit**, resp. **lifted** by another order. A **market order** is an order to buy, resp. sell a stock at the best available price immediately, i.e. to execute (lift) the best-ask, resp. (hit) the best-fill.

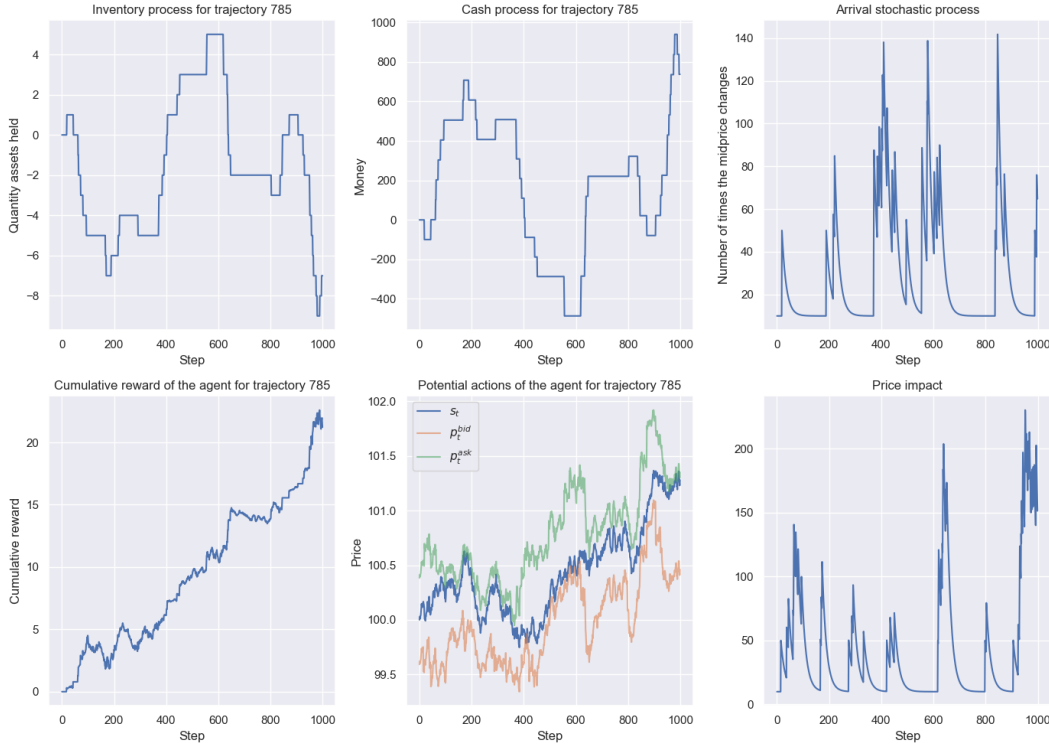


Figure 2: Markov Decision Process: Mid-price Brownian motion with volatility 1

The dynamic 'LimitOrderModelDynamics' environment models a market where our agent is but one player in the market. Under a Geometric Brownian Motion (GBM) as the mid-price, Avellaneda and Stoikov [2008] was able to derive an optimal two-step procedure for an analytic closed-form solution of optimal bid and ask quotes. First, the dealer computes a personal indifference valuation for the stock given his current inventory. Then he calibrates his bid-ask quote in the LOB by conditioning the probability of having the position filled as a function of their distance to the mid-price.

3 Trading environment

3.1 Initial settings

The initial settings are crucial as they greatly influence the regimes under which the agent is performing. Later we show that agents trained in non-volatile environments perform poorly in mildly volatile

environments. Whereas agents trained in mildly volatile environments tend to have an overall better performance because we can not predict the regime we will observe in the future, and we would like the agent to perform in any kind of regime and not in any specific one. Since cryptocurrency can be extremely volatile, we cannot train the agents every day under the regimes of the past days as they might display poor performance under unexpected events.

3.2 State space

The state space is the following six processes: cash C_t , inventory Q_t (number of assets held), time L_t , midprice S_t , arrival H_t , and price impact I_t . As depicted in Figure 2. The process is defined on the interval $I := [0, T]$ with $T = 1$. With the number of steps $n_{steps} = 1000$, the time steps are defined,

$$t_i = \frac{i \cdot T}{n_{steps}} \quad i = 0, \dots, n_{steps} \quad \text{with} \quad \Delta_t = \frac{T}{n_{steps}}.$$

3.2.1 Mid-price model

We implemented two stochastic processes for the mid-price, a Geometric Brownian Motion (GBM), which is a classic assumption (Black-Scholes), and a Brownian motion (BM) with jumps (BMJ). The jump size J is fixed and is used to model the excessive filling of orders, which results in an abrupt jump in the mid-price. In all situations, we define the drift μ to 0, as we cannot infer from external information the stock trend. σ is the volatility.

Geometric Brownian Motion Model:

$$S_{t+1} = S_t \exp \left(\left(\mu - \frac{1}{2} \sigma^2 \right) \Delta_t + \sigma \sqrt{\Delta_t} \epsilon_t \right) \quad \epsilon_t \sim \mathcal{N}(0, 1).$$

Brownian Motion (Jump) Midprice Model:

$$S_{t+1} = S_t + \mu \Delta_t + \sigma \sqrt{\Delta_t} \epsilon_t + J(\mathbb{1}_{ask}(t) - \mathbb{1}_{bid}(t)) \quad \epsilon_t \sim \mathcal{N}(0, 1).$$

$\mathbb{1}_{ask}(t)$, resp. $\mathbb{1}_{bid}(t)$ are indicator function of whether an ask, resp. a bid order is filled¹.

3.2.2 Arrival model

The arrival model describes the stochastic nature of order arrival in the market. A typical model of arrival is a Poisson process. However as displayed in Figure 3, in the financial market the Hawkes process is a more suitable assumption as it can capture the clustering effects and self-exciting nature in limit order arrivals (Pollett [2015]).

This means the occurrence of an order increases the likelihood of subsequent orders arriving shortly thereafter. The Hawkes conditional intensity is defined as,

$$\lambda_{t+1} = \lambda_t + \gamma * (\lambda_0 - \lambda_t) \Delta_t + \alpha * \mathbb{1}_{arrivals}(t), \quad \text{s.t.} \quad \mathbb{1}_{arrivals}(t) = \mathbb{1}_{bid}(t) + \mathbb{1}_{ask}(t).$$

where $\lambda_0 = 10$ is the baseline intensity, $\gamma = 60$ is the mean reversion speed, and $\alpha = 40$ is the orders jump size representing the increase in arrival rate due to new arrivals.

3.2.3 Price Impact model

The price impact model describes how trades affect the market price of an asset. In a realistic market, large trades can move the market, affecting the execution prices of subsequent trades.

Let us assume an order to buy Q units of a stock arrives at time t . Then the first lowest ask orders until Q units are filled, which causes a market impact as the transaction is at a different price than the mid-price. Let p_t^Q be the price of the highest ask order filled, then the price impact of this trade is defined as

$$\Delta p_t := p_t^Q - s_t \quad \text{Our order is filled if} \quad \delta_t^a < \Delta p_t.$$

When Q is large, more low-priced ask orders are filled, and the limit order book will rise. This is however not included in the `LimitOrderModelDynamics` environment for model simplification.

¹Their implementation is a bit trickier, $\mathbb{1}_{ask/bid}(t) = \mathbb{1}_{(\exp(-\kappa \delta^a/b) > U[0,1])}(t) \cdot \mathbb{1}_{(\Delta_t \cdot \lambda_t > U[0,1])}(t)$

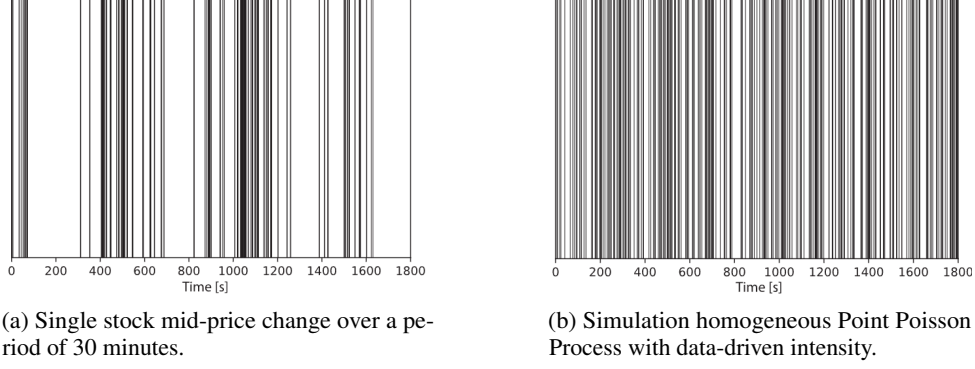


Figure 3: Display heavy clustering around mid-price change (Mark et al. [2022])

3.3 Order Filling model

Relying on Econophysics results to model the probability of having an order fill. Under constant frequency of arrivals, it has been derived that the price impact function is logarithmic (Marc Potters [2002]). The probability of having an order filled is proportional to the distance to the mid-price. Therefore, a market buy order will lift our agent's sell limit orders at a rate,

$$\lambda^a(\delta_t^a) := \exp(-\kappa \delta_t^a) \quad \kappa > 0.$$

where δ is the fill depth and $\kappa = 1.5$ is the fill exponent.

3.4 Action space

The action space is a continuous two-dimensional space composed of $[\delta_t^b, \delta_t^a]$, the distance between the mid-price and the bid/ask price.

$$\delta_t^b := s_t - p_t^b, \quad \delta_t^a := p_t^a - s_t.$$

3.5 Reward function

In market-making, two typical reward functions are Profit and Loss (PnL) and Running inventory penalty (RIP). These reward functions help in assessing and guiding the agent's trading strategy to ensure profitability and control the risk. The market maker faces two types of risk, the inventory risk intrinsic to the stock uncertainty and the asymmetric information risk arising with better-informed traders. In this project, we focus on the inventory risk and therefore adopt RIP as the reward function, because it helps the agent minimize the risk of holding inventory while pursuing larger profits.

- **Profit and Loss (PNL):** The PNL reward function measures the net profit or loss generated by the agent over a given period, considering the "mark-to-market" value of the agent's portfolio. It calculates the difference between the market value of the agent's portfolio at the current and next states.

$$PnL_t = S_t \cdot Q_t + C_t - (S_0 \cdot Q_0 + C_0).$$

- **Running Inventory Penalty (RIP):** Based on PNL, Running Inventory Penalty aims to manage the risk associated with holding large positions in the market. It adds the inventory aversion terms to penalize the agent for maintaining a high inventory, thus encouraging it to keep a balanced position.

$$RIP_t = PNL_t - \Delta t \cdot \phi \cdot Q_t^i - \alpha \cdot Q_T^i \cdot \mathbb{1}_T(t),$$

where ϕ is the constant per-step inventory aversion coefficient, α is the terminal inventory aversion coefficient, $i = 2$ is the inventory exponent, and $\mathbb{1}_{\{T\}}(t)$ is an indicator function for the terminal step T . The two penalty terms discourage the agent from holding excessive inventory and prefer a strategy of balanced buying and selling activities to reduce the inventory-holding risk.

4 Training Algorithm

Proximal Policy Optimization (PPO) is an efficient and robust policy gradient algorithm using a clipped surrogate loss function for policy updates. It enhances earlier methods like Trust Region Policy Optimization (TRPO) by simplifying implementation and improving computational performance because it eliminates the need for second-order derivatives and complex constrained optimization.

4.1 Modification MLP Network

We modify the network such that PPO constructs separate policy and value functions using networks with two hidden layers of 256 neurons and Rectified Linear Unit (RELU) as activation for its sparse representation. This allows different parts of the network to learn distinct features of the input space.

4.2 Rollout phase

In our experiment, the agent interacts with the environment over 1,000 timesteps to collect rewards, which generates trajectories of states, actions, rewards, and next states. After every 1,000 timesteps, old transitions are discarded, and new ones are gathered using the updated policy. This makes on-policy methods like PPO less transition-efficient compared to off-policy methods like DQN, which reuse transitions from a replay buffer.

4.3 Learning phase

4.3.1 Advantage estimation

Once sufficient experience has been gathered, the policy update process begins with Generalized Advantage Estimation (GAE) by computing the advantage function \hat{A}_t for each timestep:

$$\hat{A}_t = \sum_{l=0}^{T-t-1} (\gamma\lambda)^l \delta_{t+l}, \quad \text{where} \quad \delta_t = r_t + \gamma V(s_{t+1}) - V(s_t).$$

With γ the discount factor, and λ the GAE parameter. When $\lambda = 1$, GAE considers the sum of all future rewards, similar to Monte Carlo methods. This method effectively estimates the advantage by considering the temporal differences across multiple future steps. To stabilize training and improve learning efficiency, the advantages are then normalized at the minibatch level.

4.3.2 Policy update

The probability ratio $r_t(\theta)$ between the new and old policies is then computed for each action taken during the trajectory:

$$r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}.$$

The surrogate objective function of PPO is:

$$L^{CLIP}(\theta) = \mathbb{E}_t \left[\min \left(r_t(\theta) \hat{A}_t, \text{clip}(r_t(\theta), 1 - \epsilon, 1 + \epsilon) \hat{A}_t \right) \right],$$

where $r_t(\theta)$ is the ratio of the probability of taking an action under the new policy to the probability of taking the same action under the old policy, \hat{A}_t is the advantage function, and ϵ is a hyperparameter that defines the clipping range. This clipping mechanism prevents the new policy from deviating too far from the old policy, ensuring more stable and reliable updates.

Using this ratio and the clipped surrogate loss defined above, the actual policy update is performed through a stochastic gradient descent (SGD), typically involving several epochs of mini-batch updates within each iteration with Adam optimizer. The number of epochs and size of the minibatch are hyperparameters. The update rule can be expressed as:

$$\theta_{\text{new}} = \theta_{\text{old}} + \alpha \nabla_\theta L^{CLIP}(\theta),$$

where α is the learning rate. After completing the optimization epochs, the parameters of the old policy are updated to those of the new policy.

This iterative process ensures continuous refinement of the policy while maintaining stability through controlled updates.

5 Results

The main question is: What is the agent learning? More precisely, what would we like the agent to learn? We are training agents on a complex model that is still not able to perfectly replicate the entire dynamic of the market. The agent aims to perform in a very short-term horizon, making it unable to distinguish particular trends. This is good news, as the agent cannot predict long-term trends. On top of this, Mark et al. [2022] has shown that the Bitcoin market suggests an endogeneity of about 80%, which is also comforting news as we only make trades by observing the market. Even if the mid-price model is a martingale, the agent is still able to have profits. This means that under the complex framework, there is more at play than just the prediction of a martingale's next value. Most likely, the agent is leveraging the volatility of the mid-price. The agent will fall short in the prediction task, but it can utilize the "buy low, sell high" principle, which, in a theoretical framework, is again a martingale. But with the implementation of a Hawkes process for the mid-price arrivals, the agent learns the chain reaction following an order execution and can get in and out with a bonus spread.

5.1 Training

The mean reward plot in Figure 4a hints that the agents will show better performing results with increased steps in their learning stage, which is the case. Nevertheless, the improvement looks to be asymptotically logarithmic and bounded! Figure 4b displays that the explained variance (Equation 1), is reaching a cap as shown on the black and blue curve of Figure 4b. It gets to a point where it cannot predict any better and starts to lose past information, which is a common issue for PPO when a large number of training steps are used. The predictive ability of the agent is high when there is no jump, but as soon as small jumps are introduced its predictive performance is nearly divided by half. For all jump processes, the explained variance converges in the range of $[0.4, 0.5]$.

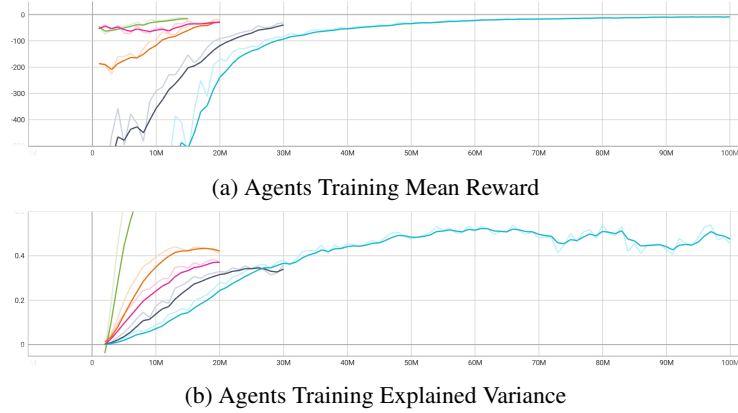


Figure 4: Pink = $\text{GBM}_{\sigma=0.1}$, Green = $\text{BM}_{\sigma=2}$, Orange = $\text{BM}_{\sigma=0.1}^{J=1}$, Black = $\text{BM}_{\sigma=1}^{J=4}$, Blue = $\text{GBM}_{\sigma=3}^{J=8}$.

5.2 Performance

After the training of the agent, we tracked the performance of the agent over a thousand trajectories. As we can see in Figure 5, the RIP performs worse than the PnL due to the added penalization. In all the models presented, the distribution of the end cumulative reward and end PnL is highly skewed with a fat left tail. This means that in certain situations the agent performs terribly, which worsens the statistics of the performance. As strange as it first sounds, with increased jump size, it has improved performance. This is due to the randomness of the mid-price, over a thousand trajectories it can happen for the agent to encounter a trajectory for the mid-price far different than the one usually observed. Given that the agent with a larger jump size is more easily penalized, he will take less risk and not issue such losses. This is a positive insight as the standard deviation of these two metrics is therefore an indicator of robustness under different regimes.

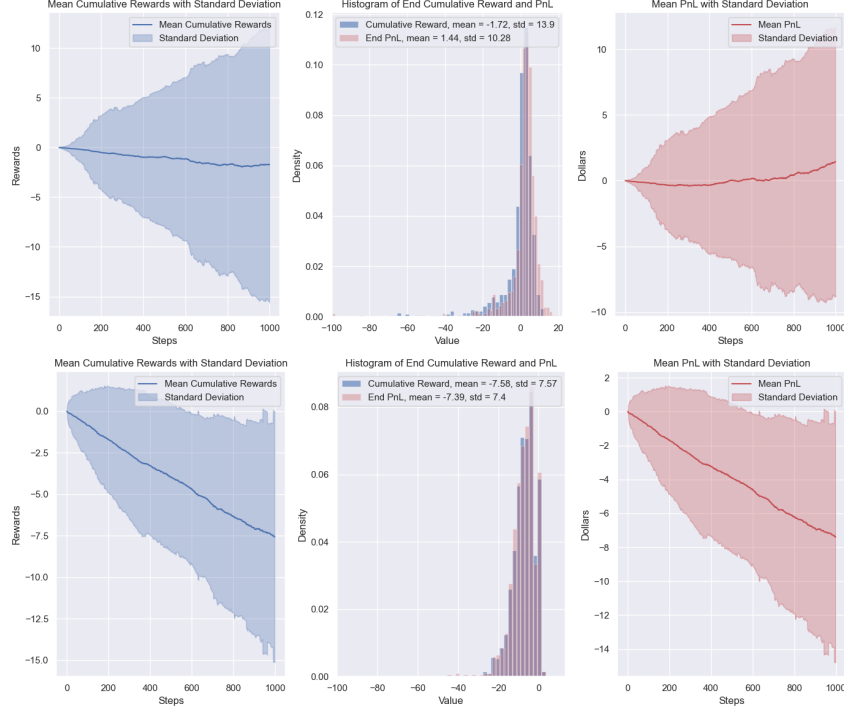


Figure 5: Statistics of performance for $\text{GBM}_{\sigma=1}^{J=4}$ and $\text{GBM}_{\sigma=3}^{J=8}$

5.3 BackTest

Given the peculiarity of our state space and more precisely its unpredictability, we should not rely on the agent's performance in its training settings nor on its reward function as we implemented the RIP to hedge against risk. Therefore the most important metric on Table 1 is the PnL.

The backtest data is the Bitcoin trading data of 1.04.2024 in UTC. The first data set represents the first ten minutes and the second one the first hour. The step size of the first one is 0.25 (0.25 seconds, 250 ms), and of the second is 0.01 (0.01 seconds, 10 ms), which indicates the decision-making interval. The fill probability is not data-driven and is kept as exponential as is in the training environment.

Implementing a mid-price without jumps with $\text{BM}_{\sigma=0}^{J=0}$ for the mid-price or with only jumps $\text{BM}_{\sigma=0.1}^{J=1}$ does not provide significant results as displayed in Table 1.

We then wonder if the most stochastic training environment is the best. To explain this, we can make an analogy with the training of a regional football team. If the team only trains against children (under $\text{BM}_{\sigma=0}^{J=0}$), they will not learn under constraints and can roam freely and take personal initiatives. When they are confronted by the national team (Backtest data), they will be highly punished for their disorganized behavior. As shown in Figure 9 with the quantities of assets held. Therefore, we can ask if they will show better results if they are trained against the World Champion team $\text{BM}_{\sigma=3}^{J=8}$? The answer is that in this instance, they will be too highly punished to display great results against the national team, as we can see in Figure 7. The best performance comes from training against a slightly stronger team than they will face, as we can see in Figure 8. The agent $\text{BM}_{\sigma=1}^{J=4}$ is the one taking the most actions and being the most rewarded. It never holds or sells more than three stocks. Whereas $\text{BM}_{\sigma=4}^{J=8}$ was trained in a more volatile environment and never holds or sells more than one stock! Hence, in an even more volatile training environment, we can expect to see the agent taking no action (as if the football team stayed in front of the goal to avoid conceding a goal).

We also see that depending on the training settings the agent will ask for a larger spread, which corroborates Glosten-Milgrom's theory. The pronounced jump in the mid-price is a result of important orders that can signal the presence of informed traders. Therefore the agent will increase his spread to hedge against this risk.

We trained the agents with a mid-price oscillating around 100, which made the agents quote their

mean spread in the range of [0.5,3]. To obtain better backtest results, one should calibrate the starting price of the mid-price to obtain a spread aligning with the market one. Overall the performance of the agents with high jump sizes are quite satisfactory on the 250ms data. Such a result happens when we are confronted to stochastic processes that increase in volatility when scaled. Meaning that under our performance, we can only withstand a certain threshold of volatility.

	$\text{GBM}_{\sigma=0.1}$	$\text{BM}_{\sigma=2}^{J=0}$	$\text{BM}_{\sigma=0.1}^{J=1}$	$\text{BM}_{\sigma=1}^{J=4}$	$\text{BM}_{\sigma=3}^{J=8}$
Mean PnL	15.89	17.73	1.44	-18.29	-7.39
Mean CumReward	-15.39	5.35	-1.72	-19.2	-7.58
Sdev PnL	87.54	11.89	10.28	13.92	7.4
Sdev CumReward	98.54	26.91	13.9	14.57	7.57
Mean PnL	4811.78	-620.59	673.04	910.79	201.81
Mean CumReward	-736,160.7	-12678.93	-958.42	422.95	65.04
Sdev PnL	402.81	211.54	82.97	116.12	19.58
Sdev CumReward	92715.72	3524.97	122.58	118.71	21.03
Mean PnL	-17,841.6	-5780.27	-1201.18	42.24	19.62
Mean CumReward	-1,927,709.9	-31,680.1	-2843.91	-66.85	-6.02
Sdev PnL	2004.37	243.25	148.61	47.52	21.03
Sdev CumReward	899,779.09	7534.73	316.15	55.74	19.84

Table 1: Comparison Agents Performance: Training Settings/ Backtest 250ms/ Backtest 10ms

6 Future work

As described before with the Explained Variance and Mean Episodic Reward plot we are encountering a bound of performance. As demonstrated, taking a more volatile environment will not be the solution to our problem. We would like to have the most adversarial agent without conceding on its reward. This is why it could be necessary to implement other RL algorithms.

Robust Adversarial Reinforcement Learning (RARL) is designed to train agents to operate effectively under destabilizing adversarial forces. This method uses a zero-sum, minimax objective function, where an adversary is trained alongside the agent to learn optimal policies that destabilize the agent's performance. The adversary applies disturbance forces to the system, creating a more robust and stable training environment for the RL agent. The effectiveness of RARL, like any RL algorithm, depends on the specific task and environment. Therefore, it could be performing poorly in our environment.

7 Conclusion

An RL approach for market making was explored in cryptocurrency trading using Limit Order Book (LOB) dynamics. Agents are trained with the PPO algorithm in the trading environment simulated with `mbt_gym` and backtested with historical bitcoin data in different frequencies (10 ms and 250 ms). The study shows that PPO agents, particularly those trained in more mildly stochastic environments, outperform the ones trained in nonstochastic environments.

References

- Marco Avellaneda and Sasha Stoikov. High-frequency trading in a limit order book. *Quantitative Finance*, 8(3):217–224, 2008. doi: 10.1080/14697680701381228.
- Joseph Jerome. Mbt-gym: Reinforcement learning for model-based limit order book trading. In *Proceedings of the ACM Conference*, 2023. doi: 10.1145/3604237.3626873. URL <https://doi.org/10.1145/3604237.3626873>. arXiv preprint arXiv:2209.07823.
- Jean-Philippe Bouchaud Marc Potters. More statistical properties of order books and price impact. *The European Journal of Finance*, 2002. doi: <https://arxiv.org/abs/cond-mat/0210710>.

Michael Mark, Jan Sila, and Thomas A. Weber. Quantifying endogeneity of cryptocurrency markets. *The European Journal of Finance*, 28(7):784–799, 2022. doi: 10.1080/1351847X.2020.1791925.

Patrick J. Laub Thomas Taimre Philip K. Pollett. Hawkes processes. 2015. URL <https://arxiv.org/abs/1507.02822>.

8 Appendix

8.1 Explained Variance

$$\text{Explained variance} = 1 - \frac{\text{Var}[\text{Empirical Return} - \text{Predicted Value}]}{\text{Var}[\text{Empirical Return}]}. \quad (1)$$

It indicates how well the value function predicts returns. Values close to 1 indicate perfect predictions, while values less than or equal to 0 suggest poor performance.

8.2 Backtest Results

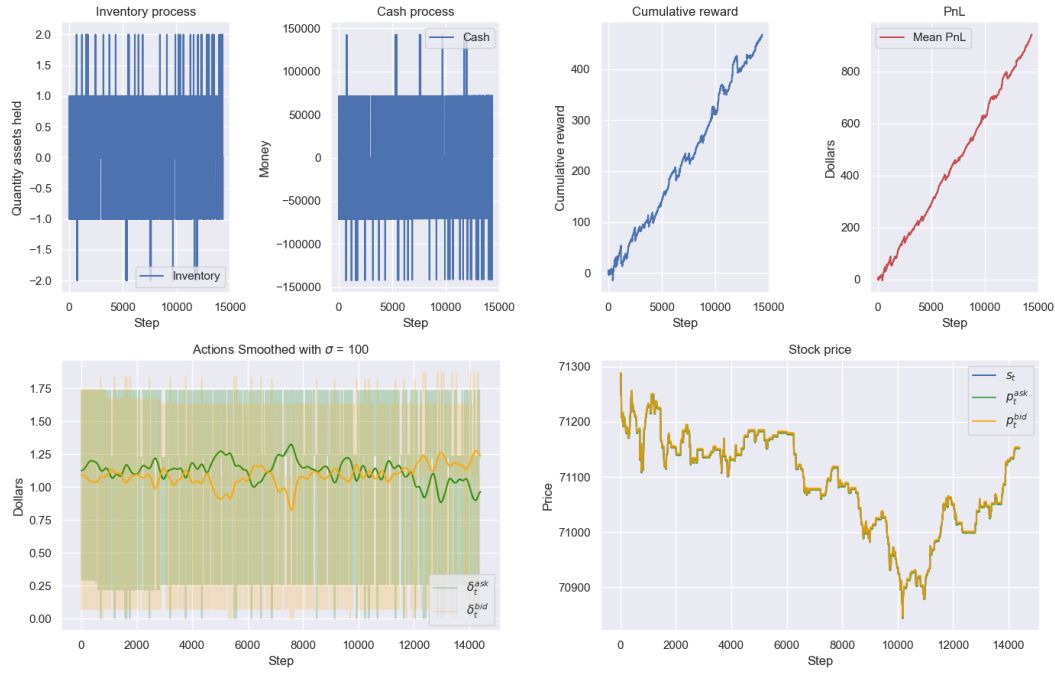


Figure 6: $\text{BM}_{\sigma=1}^{J=4}$ Backtest Performance 250ms

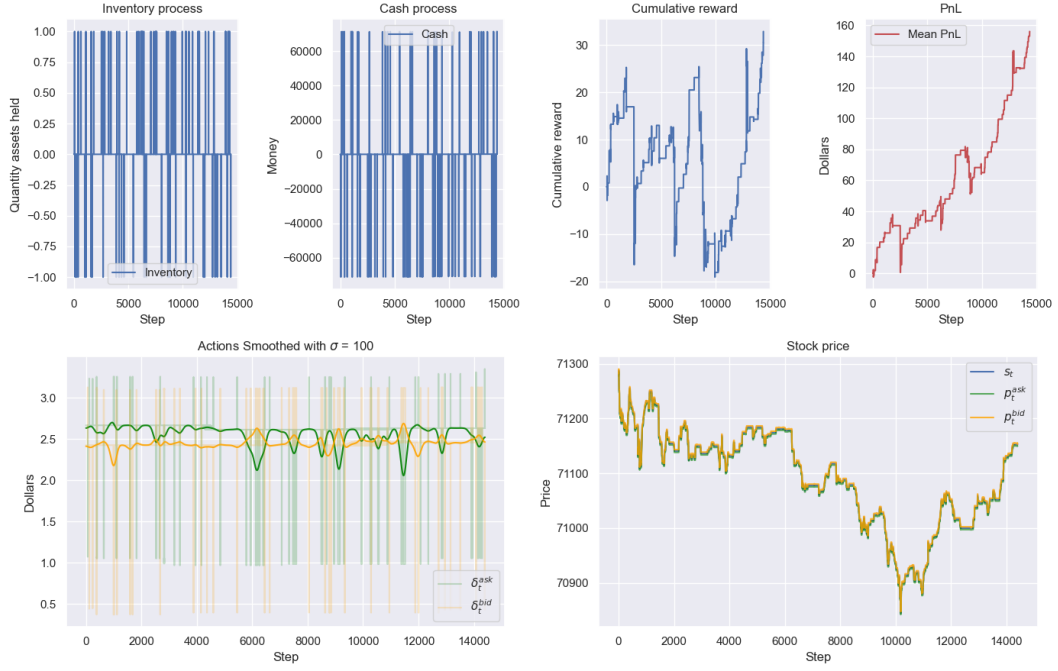


Figure 7: $\text{BM}_{\sigma=3}^{J=8}$ Backtest Performance 250ms

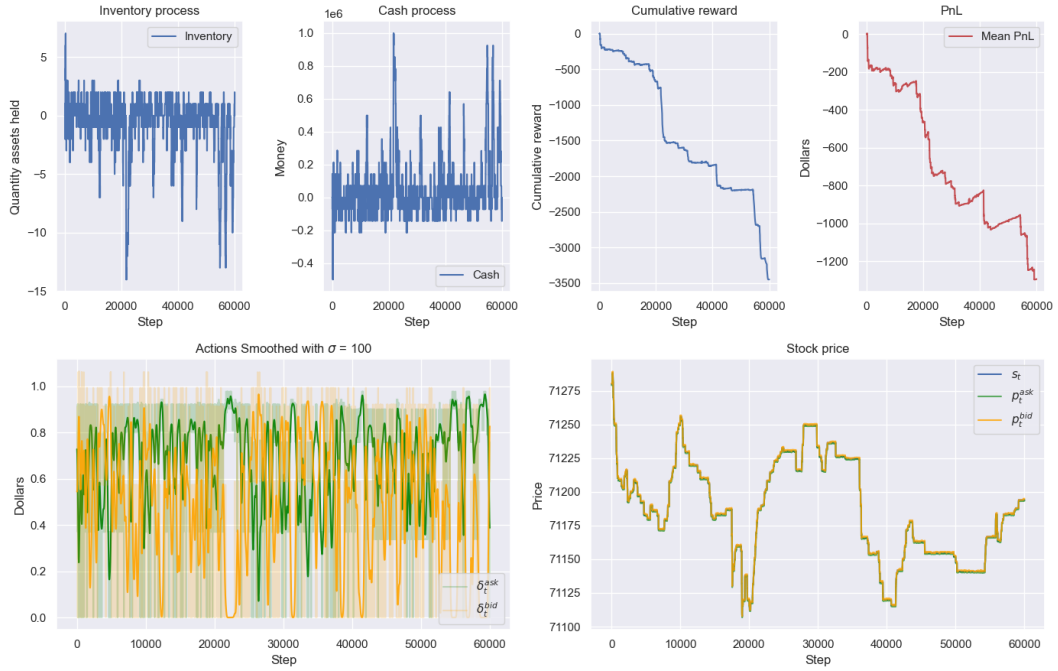


Figure 8: $\text{BM}_{\sigma=0.1}^{J=1}$ Backtest Performance 10ms

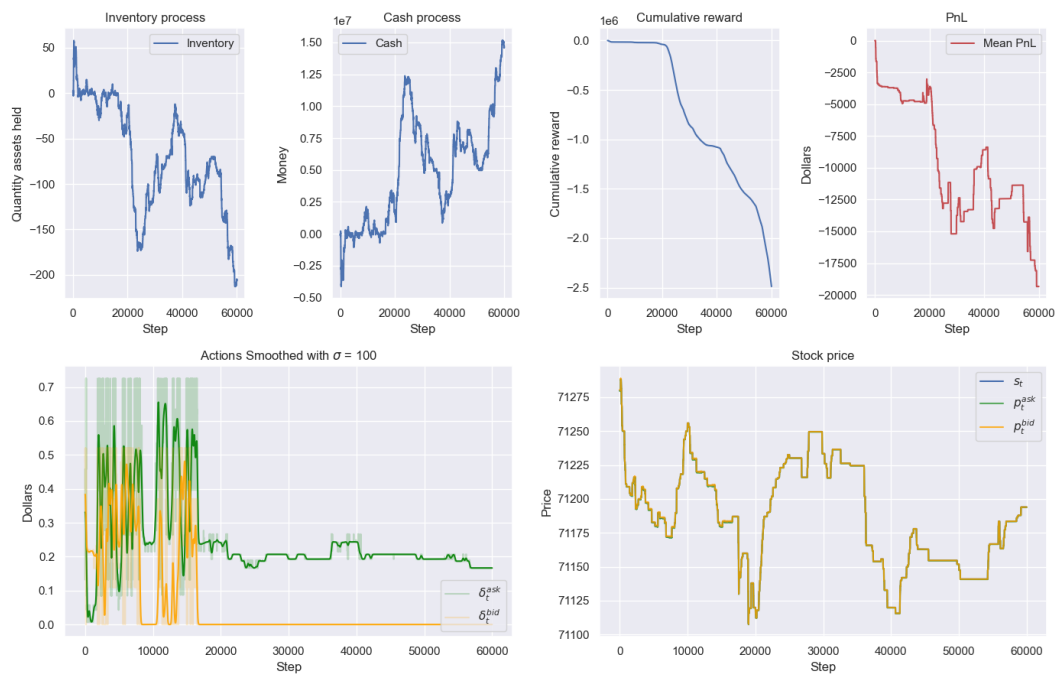


Figure 9: $\text{GBM}_{\sigma=0.1}$ Backtest Performance 10ms