

Event-Based Limit Order Book Simulation under a Neural Hawkes Process: Application in Market-Making

Luca Lalor¹ and Anatoliy Swishchuk¹

¹University of Calgary, Department of Mathematics and Statistics, Calgary, AB T2N 1N4, Canada

February 25, 2025

Abstract

In this paper, we propose an event-driven Limit Order Book (LOB) model that captures twelve of the most observed LOB events in exchange-based financial markets. To model these events, we propose using the state-of-the-art Neural Hawkes process, a more robust alternative to traditional Hawkes process models. More specifically, this model captures the dynamic relationships between different event types, particularly their long- and short-term interactions, using a Long Short-Term Memory neural network. Using this framework, we construct a midprice process that captures the event-driven behavior of the LOB by simulating high-frequency dynamics like how they appear in real financial markets. The empirical results show that our model captures many of the broader characteristics of the price fluctuations, particularly in terms of their overall volatility. We apply this LOB simulation model within a Deep Reinforcement Learning Market-Making framework, where the trading agent can now complete trade order fills in a manner that closely resembles real-market trade execution. Here, we also compare the results of the simulated model with those from real data, highlighting how the overall performance and the distribution of trade order fills closely align with the same analysis on real data.

Keywords: Algorithmic and High-Frequency Trading, Limit Order Books, Deep Reinforcement Learning, Multivariate Hawkes Process, Neural Hawkes, Market Simulation.

1 Introduction

There are many works devoted to modeling limit order book (LOB) data, which often follow some form of approximation process for the midprice¹ process. A detailed overview of LOB models can be found in the survey paper by Gould et al. (2013), and also in a more recent study by Jain et al. (2024), where much of the current empirical evidence in the popular literature is presented, along with a clear indication that there are still many limitations, not limited to just the model of the price process. In this paper, we construct a midprice process from the ground up, using real LOB events to replicate its evolution in the market. In the LOB system, the three main event types are limit order entries, market order entries that lead to trade order executions, and limit order cancellations. Each time one of these event types occurs, they are given a unique timestamp, almost surely, and they occur at non-uniform discrete time points. We believe that a model that can account for this event-based structure is more in line with the realities of LOB modeling for Algorithmic and High-Frequency Trading (HFT) strategies.

Popular LOB models in the literature often rely on approximations for pricing, typically ignoring the discrete-time, event-driven nature of real LOB dynamics. This introduces limitations beyond merely misapproximating the midprice process, potentially leading to greater inaccuracies when simulating the performance of algorithmic and HFT strategies. For instance, as shown in Law and Viens (2019), many models that include some form of diffusion-based pricing model (with some popular and regularly cited works including Bertsimas and Lo (1998), Bouchard et al. (2011), Cartea et al. (2015), Guéant (2017), often lead to so-called “phantom gains” appearing in Algorithmic and HFT applications, particularly under trading strategies that involve executing numerous limit orders, such as in Market-Making (MM). When backtesting or implementing an MM strategy, it becomes evident that identifying which LOB event(s) caused a price movement is crucial. However, random-walk models like Brownian motion cannot capture this, as LOB events are entirely independent of the modeled diffusion process. One simple example that could help clarify this for the reader would be to think about a MM agent who posts limit orders on the best bid/ask. Their limit orders are much more likely to be executed when price goes against them than in their favor, which a diffusion-based model cannot account for. Thus, simulating a MM strategy within a diffusion model often overestimates the ease of obtaining favorable execution for limit orders compared to real-world conditions. Empirical evidence in line with this particular example can be found in Lalor and Swishchuk (2024b) and DeLise (2024), where the adverse selection problem in the LOB that underpins this particular trading strategy is studied. Diffusion-based pricing models are widely used in the mathematical finance, algorithmic trading, and HFT literature due to their simplicity in modeling and simulation. However, their inherent limitations become more pronounced in HFT contexts, reducing their applicability in real-world markets.

In this study, we propose a novel approach that models 12 LOB events using a nonlinear multivariate Hawkes process (MVHP), simulated within a standard Neural Hawkes process framework. We then apply this LOB modeling approach to a state-of-the-art deep reinforcement learning (RL) algorithmic and HFT problem, a relatively new but more robust method for analyzing such mathematical optimization problems. We model 12 types of LOB events, capturing a significant portion of real-world LOB activity. This can be extended to include additional event types available in certain markets, provided their impact on the midprice process and the intensity function governing event frequencies is

¹The midprice is the midpoint between the best bid and ask price.

well understood. Some of the more recent literature has made similar attempts, such as in Gašperov and Kostanjčar (2022) where they devised a MM framework that applies a midprice model that evolves based on the linear MVHP model formulated in Lu and Abergel (2018). However, we believe that creating a MVHP that performs a nonlinear transformation to a Linear MVHP to be more general, and more in line with how LOB pricing data should be modeled. Empirical evidence supporting this theory is provided in Lu and Abergel (2018) and Shi and Cartlidge (2022), demonstrating that a nonlinear MVHP model fits LOB data better than a linear MVHP or other traditional Hawkes models.

Building on this LOB event structure, we construct a midprice process using a neurally self-modulated multivariate point process, known as a Neural Hawkes Process (Neural HP), which was first introduced in Mei and Eisner (2017). To date, we have identified only two applications of the Neural HP in modeling LOBs, as also noted in the recent survey by Jain et al. (2024). In Shi and Cartlidge (2022), empirical evidence suggests that their LOB event model, based on four LOB event types, better simulates LOB data compared to traditional stochastic HP models and other deep learning methods. In Kumar (2024), the Neural HP is used to simulate LOB data after fitting it to specific datasets while also incorporating trader actions through an agent-based model. In addition to capturing the advantages of a nonlinear MVHP, which enables more accurate modeling of self- and cross-excitation relationships between LOB events, Neural HPs also address some of the limitations still present in traditional HP models. These limitations include the requirement for the excitation function to generate only positive values, the additive nature of past events, and the fixed decay structure of the intensity function. For example, within LOB data it is clear from the empirical evidence in Lu and Abergel (2018) that inhibition effects, rather than excitation effects, are quite common between certain LOB events, which a well-trained Neural HP model can account for. Consequently, Neural HP models are better suited for capturing these complex dependency relationships.

The main contributions of this paper can be summarized as follows:

- i. We develop a Neural Hawkes LOB simulation framework based on 12 key LOB events observed in electronic financial markets. Empirical evidence demonstrates how this event-based approach closely aligns with real-world LOB event arrivals, capturing their nonlinear dependencies. This is a crucial feature for HFT, as individual events or sequences of events carry granular information essential to short-term traders.
- ii. This event-based LOB structure allows us to develop an asset price simulation driven by a unique jump process, where jumps correspond to the probability of specific jump sizes occurring within each LOB event category. We demonstrate how this approach can be structured based on asset-specific event characteristics and applied to various HFT scenarios.
- iii. This is the first work to apply a Neural Hawkes event-based LOB model to a High-Frequency MM strategy and demonstrate how this structure enables more realistic HFT strategy backtesting. We believe this represents a significant extension of previous MM literature, as it allows for more accurate simulation of trade order fills based on the LOB events required to complete a trade. Additionally, it improves the modeling of LOB event frequencies and timing. These aspects are crucial for High-Frequency MM simulations and have received limited attention in existing research.

The rest of this paper will proceed as follows. In Section 2, we introduce the midprice model used throughout the paper. Section 2.1 provides a discussion and analysis of the included LOB events, while Section 2.2 presents an overview

of the nonlinear MVHP modeling framework. In Section 3, we describe the Neural MVHP midprice simulation framework. Section 3.1 introduces the Neural Hawkes process model, while Section 3.2 details the midprice simulation process and discusses key simulation results. In Section 4, we will then proceed to provide an application under a Deep RL framework, where we will focus on a High-Frequency MM trading problem. Here, we conduct a comparative analysis between the simulated and real LOB data across five different assets. Lastly, we will discuss our conclusions and future research recommendations.

2 Midprice Process Modeling

In this section we proceed to discuss the formulation of our Midprice modeling process, under a nonlinear MVHP framework. Over the last 15-20 years, HPs have seen many applications in finance, where a broad survey can be found in Bacry et al. (2015), and more recently also in Algorithmic and HFT. A HP is a self-exciting point process where the occurrence of an event increases the likelihood of future events occurring within a short time frame. This makes it well-suited for modeling clustered event arrivals, such as LOB activity in financial markets. Examples of applications in the LOB modeling space include Abergel and Jedidi (2015) where they study the long-term behavior of applying HP to LOBs, Cartea et al. (2018b) where a order flow metric is modeled using a HP, Swishchuk and Huffman (2020) and Swishchuk et al. (2019) which formulates a selection of General Compound HP models that can be applied within certain trading problems, where different applications can be found in Roldan Contreras and Swishchuk (2022) and Lalor and Swishchuk (2024a). This is by no means an extensive review of HP applications in LOBs, for this we refer the reader to the survey paper by Jain et al. (2024). However, it is clear from much of the recent literature to date that HPs are a suitable choice for modeling LOB events, particularly from a point process standpoint.

The rest of this section will proceed as follows. In Section 2.1, we provide an overview of the 12 LOB events examined in this paper, along with an analysis of their occurrence frequencies in real LOB data. In Section 2.2, we will describe the nonlinear MVHP dynamics used to model these events, along with our midprice process that represents price moves based on these events.

2.1 *Limit Order Book Events*

In selecting which LOB events to model, we follow a similar set of events as used in Lu and Abergel (2018), Law and Viens (2019) and Gašperov and Kostanjčar (2022). In these works, the focus is on how each respective LOB event affects the midprice i.e., whether these events caused the midprice to increase, decrease or remain unchanged. Irrespective of their effect on the midprice, they all affect the MVHP intensity function that will be introduced in Section 2.2, through the self- and cross-excitation effects that can be modeled by nonlinear MVHP models. Subsequently, these LOB events can be introduced as follows:

- i. Aggressive Limit Buy Order (LB^+): A limit order that moves the midprice up by posting at a higher bid price than the previous best bid.
- ii. Aggressive Limit Sell Order (LS^-): A limit order that moves the midprice down by posting at a lower ask price than the previous best ask.
- iii. Aggressive Market Buy Order (MB^+): A market order that moves the midprice up and depletes at least one ask queue.

- iv. Aggressive Market Sell Order (MS^-): A market order that moves the midprice down and depletes at least one bid queue.
- v. Aggressive Limit Buy Cancellation (BC^-): A canceled buy limit order that leads to a decrease in the midprice as the queue size at the previous best bid price is emptied.
- vi. Aggressive Limit Sell Cancellation (SC^+): A canceled sell limit order that leads to an increase in the midprice as the queue size at the previous best ask price is emptied.
- vii. Non-aggressive Limit Buy Order (LB^0): A limit buy order that leaves the midprice unchanged.
- viii. Non-aggressive Limit Sell Order (LS^0): A limit sell order that leaves the midprice unchanged.
- ix. Non-aggressive Market Buy Order (MB^0): A market buy order that leaves the midprice unchanged.
- x. Non-aggressive Market Sell Order (MS^0): A market sell order that leaves the midprice unchanged.
- xi. Non-aggressive Limit Buy Cancellation (BC^0): A limit buy order cancellation that leaves the midprice unchanged.
- xii. Non-aggressive Limit Sell Cancellation (SC^0): A limit sell order cancellation that leaves the midprice unchanged.

In this paper, we used LOBSTER (2025) data, which is a website partnered with Nasdaq and the Universität Wien that offers high-frequency data on multiple stocks, upon which many researchers now rely on in algorithmic and HFT research. They offer some free data on one day, June 21, 2012, throughout the full trading day (9:30 AM–4:30 PM EST). This free data includes LOB data up to 10 price levels, along with the transaction message feeds, for stocks listed on the NASDAQ exchange. Table 1 presents the frequency of each previously described LOB event across five stock tickers (Apple-AAPL, Amazon-AMZN, Alphabet-GOOG, Intel Corp.-INTC, and Microsoft-MSFT), which were derived using all 10 LOB levels. The last column of Table 1 displays the probability of each LOB event occurring, conditional on the occurrence of any LOB event, based on data from these five stocks.

As it is clear that LOBs evolve based on the above type of LOB events, we believe it is paramount that any midprice simulation process account for this, specifically when studying high-frequency data and short-term trading strategies. As in Lu and Abergel (2018) and Gašperov and Kostanjčar (2022), we will proceed to show how each of these LOB events can be modeled using a nonlinear MVHP model, as MVHPs have been commonly used to model the relationship between these events. In the HFT strategy simulation literature, only a Linear MVHP has been used to simulate the LOB data, thus we aim to extend previous work by simulating a nonlinear MVHP. Nonlinear MVHPs offer greater flexibility in modeling complex dependencies between events, as they allow for non-additive, state-dependent excitation effects and can capture inhibition or saturation effects that linear models cannot. This makes them better suited for applications like LOB modeling, where event interactions are highly dynamic and not strictly additive.

Event-Type	Stock Ticker					Probability
	AAPL	AMZN	GOOG	INTC	MSFT	
LB^+	16,805	6,611	6,496	739	951	0.01497
LS^-	17,474	6,993	6,381	869	1,078	0.01554
MB^+	5,876	1,871	1,748	725	931	0.00528
MS^-	6,227	2,368	2,111	695	888	0.00582
BC^-	8,969	5,013	3,904	95	104	0.00857
SC^+	8,999	4,701	3,444	94	108	0.00822
LB^0	65,714	56,792	27,626	162,421	153,972	0.22101
LS^0	91,021	61,588	30,755	140,761	173,565	0.23577
MB^0	12,385	3,644	4,121	17,939	15,784	0.02552
MS^0	10,502	3,535	3,697	13,123	15,811	0.02211
BC^0	65,830	56,413	27,839	153,903	144,480	0.21245
SC^0	90,588	60,248	29,793	132,675	161,092	0.22474

Table 1: The total number of occurrences of each LOB event type from up to ten LOB levels in the LOBSTER dataset on June 21st, 2012, along with the probability of each event occurring based on these 5 stocks.

2.2 Multivariate Hawkes Process Midprice Modeling

Consider $(N(t) = (N_1(t), N_2(t), \dots, N_m(t)))$ to be a simple multivariate counting process, where only one event can occur at any given time. This assumption is realistic for LOB data due to its high-resolution nature, where each LOB event has a unique timestamp, almost surely. The conditional intensity process of this simple multivariate counting process is $\lambda(t) = (\lambda_1(t), \lambda_2(t), \dots, \lambda_m(t))$, representing the number of LOB events of type i occurring up to time t , for $i = 1, \dots, m$. In our case $m = 12$, corresponding to the number of different LOB event types. The intensity function for each $\lambda_i(t)$, for the i -th event at time t , can be defined as follows:

$$\lambda_i(t) = \phi_i \left(\lambda_i + \int_{(0,t)} \sum_{j=1}^m \mu_{ij}(t-s) dN_j(s) \right), \quad (1)$$

where λ_i is the baseline intensity of the i -th event which may be time dependent, μ_{ij} is a kernel function (often an exponential or power-law function in the literature due to their simplicity) describing the influence of past events of type j on type i events, and ϕ_i is a nonlinear function that transforms the linear MVHP into a nonlinear MVHP. For our purposes, we will select the exponential kernel function for simplicity, and so we set $\mu_{ij} = \alpha_{ij} e^{-\beta_{ij} t}$. Equation (1) can then be rewritten as follows:

$$\lambda_i(t) = \phi_i \left(\lambda_i + \int_{(0,t)} \sum_{j=1}^m \alpha_{ij} e^{-\beta_{ij}(t-s)} dN_j(s) \right), \quad (2)$$

where α and β govern the self($i = j$)- and cross($i \neq j$)- excitation effects between each of the events being modeled.

Next, we divide our list of LOB events in Section 2.1 into three categories, based on their respective midprice movements. Notice the shorthand notation we gave these events in Section 2.1, where the exponents $+$, $-$, and 0 refer to LOB events that lead to an increase, a decrease, or no change in the midprice. These LOB event categories can be described as follows.

- $O_u = \{MB^+, LB^+, SC^+\}$ represents the LOB events that cause the mid-price to increase.
- $O_d = \{MC^-, LS^-, BC^-\}$ represents the LOB events that cause the mid-price to decrease.
- $O_n = \{MB^0, MS^0, LB^0, LS^0, BC^0, SC^0\}$ represents the LOB events that do not cause the midprice to change.

Note also that the LOB events in the sets O_u and O_d are referred to as aggressive events since they cause the midprice to change, whereas events in the set O_n are referred to as non-aggressive events as they cause no change in the midprice.

As the LOB evolves based on these types of events occurring discretely, we now define the evolution of a midprice process in discrete-time as follows:

$$\begin{aligned}
V(t + \Delta t) &= V(t) + \frac{\Delta}{2} \times \left[\sum_{k \in O_u} \mathbb{I}_k(t) a(X_k) - \sum_{l \in O_d} \mathbb{I}_l(t) b(X_l) \right. \\
&\quad \left. + \sum_{m \in O_n} \mathbb{I}_m(t) c(X_m) \right] \\
&= V(t) + \frac{\Delta}{2} \times \left[\sum_{k \in O_u} \mathbb{I}_k(t) a(X_k) - \sum_{l \in O_d} \mathbb{I}_l(t) b(X_l) \right].
\end{aligned} \tag{3}$$

Note that the third term vanishes because the jump sizes are always zero for the non-aggressive LOB events present in the set O_n . The right-hand side of Equation (3) can be summarized as follows:

- $V(t)$ represents the value of the asset at time t .
- Δt represents the time step, which here is non-uniform as LOB events occur at irregular time intervals.
- Δ represents the tick size/bid-ask spread of the asset being modeled, where $\frac{\Delta}{2}$ represents the half-spread.
- $\sum_{k \in O_u} \mathbb{I}_k(t) a(X_k)$ accounts for the upward movements (represented by the LOB events in the set O_u), where,
 - $\mathbb{I}_k(t)$ checks if an event k occurs or not at time t ,
 - $a(X_k)$ describes the mapping from the current state of the asset to a price move, where X_k represents the type of price movement. This price movement can be modeled where there are many possible tick movements i.e., 1 tick, 2 ticks, 3 ticks, etc.
- $\sum_{l \in O_d} \mathbb{I}_l(t) b(X_l)$ and $\sum_{m \in O_n} \mathbb{I}_m(t) c(X_m)$ can be described similarly for downward and non-change movements, respectively.
- Recall that events in the category $O_u(t)$ ($O_d(t)$) cause the midprice to increase (decrease), while events in the category $O_n(t)$ cause no change in the midprice, but still affect the intensity function in Equations (1)-(2).

This is a more general version of the mid-price process given in Lu and Abergel (2018), where they assumed that the jump size for each + and – event is always one tick. If we simplified our model to their model, the jump size for each + or – LOB event would be equal to 1, that is, $a(X_k) = b(X_k) = 1$ at each time step. This would lead us to get the following.

$$V(t + \Delta t) = V(t) + \frac{\Delta}{2} \times \left[\sum_{k \in O_u} \mathbb{I}_k(t) - \sum_{l \in O_d} \mathbb{I}_l(t) \right] \quad (4)$$

which is the same as midprice process as in Lu and Abergel (2018), but in discrete-time.

3 Midprice Simulation via a Neural Hawkes Process

The Neural HP extends traditional HP’s by formulating continuous-time recurrent neural networks (RNNs) to model event sequences with adaptive temporal dependencies, as are apparent in nonlinear MVHP models. This approach overcomes many of the limitations that appear in most of the classical HP models, such as assuming a fixed parametric form for the intensity function. The Neural HP model in the original paper, by Mei and Eisner (2017), is built using a Long Short-Term Memory (LSTM) RNN, which learns the intensities of specific events over time. The idea of modeling LOBs based on the structure of a Neural HP is relatively new, and it aims to extend previous work on applying HPs to LOB models by allowing the process to capture more real-world phenomena. In particular, inhibition and inertia effects in LOB data, as shown in Lu and Abergel (2018), can now be modeled. More specifically, the $\alpha_{i,j}$ and λ_i values in Equation (2) can now be negative, which is not possible in the more restrictive nonlinear MVHP framework. This could result in a nonnegative activation function, i.e., negative intensities, which subsequently gets passed through a nonlinear transfer function to ensure positivity. Additionally, the combined effect of past events is not required to be additive, and it can also account for delayed effects that are often not apparent through the simple decay constant used under regular HPs like in Equation (2). Thus, the Neural Hawkes process has the ability to overcome some of the shortcomings in traditional HP model, where now the influence of past events on future events can be greater than, less than, or even reduce their impact, and it may also depend on the order in which past events occurred.

In this section, we will proceed by introducing our Neural HP model in Section 3.1, where we will outline the network architecture and how we conducted our training and testing, where a summary of these results will also be given. We will also show how the 12 LOB events in our model can be simulated based on these results. Then in Section 3.2, we will discuss how we use these results to simulate our LOB event-based midprice process, which extends much of the previous literature where an independence between events and price processes is often assumed. We will also provide sample midprice simulations for the assets in the LOBSTER data, where we provide an analysis of how many parts of these results are in line with the real LOB data discussed in Section 2.1.

3.1 *Neural Hawkes Process Framework*

Here we explain the main components of the Neural HP model we developed, where many parts are heavily in line with the standard Neural Hawkes model in Mei and Eisner (2017) and in Shi and Cartledge (2022), where the latter is an extension of the original Neural Hawkes model applied to a state-dependent LOB model. One can also visualize the structure of the network simulation process we created in Figure 1. To give a brief overview before going into more

depth, this network receives the LOB event-type and a market state variable as an input, the LSTM cell then approximates the structure of the intensity function which evolves via the exponential kernel, and as an output, the network receives a value for each LOB event’s new intensity. Our model is trained using the LOBSTER data described earlier, where we split the data into 60% training, 20% validation, and 20% testing. In the following parts of this section, we will describe the working parts of this framework in more detail.

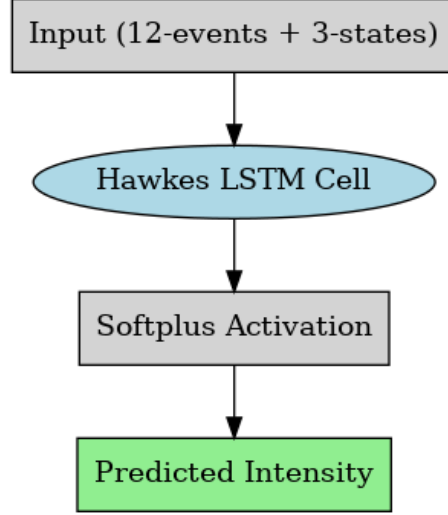


Figure 1: A visualization of the structure of our LSTM Neural Hawkes LOB model.

The main working parts of this framework can be summarized as follows:

- **Input:** The model takes as input the LOB event type through one-hot encoding of event-type and a market state, which is an additional volume imbalance indicator feature representing the current state of the market, as first described in Cartea et al. (2018a) and also used in Shi and Cartlidge (2022). This market state feature uses the volume posted on the best bid or best ask as a proxy for predicting short-term order flow. More specifically, denote this predictor as I where:

$$I = \frac{v^{b(1)} - v^{a(1)}}{v^{b(1)} + v^{a(1)}}, \quad (5)$$

where $v^{b(1)}$ and $v^{a(2)}$ is the volume posted on the best bid and ask, respectively. Then, the state of the market is defined categorically as follows:

$$x = \begin{cases} 0, & \text{if } I \in [-1, \theta] \\ 1, & \text{if } I \in [-\theta, \theta], \\ 2, & \text{if } I \in [\theta, 1] \end{cases} \quad (6)$$

where here x denotes the market state and θ is a fixed parameter setting the boundary values. In our model, we set $\theta = 0.4$ as in Shi and Cartlidge (2022). θ essentially determines the boundary for specifying the difference between a balanced (1) and unbalanced market (0 or 2). This predictor essentially states that if the volume imbalance indicator predicts an up movement, events which cause the midprice to go up are more likely to occur, with a similar logic for the down and no move categories.

- **LSTM Memory Update:** The famous LSTM structure, as first introduced in the seminal paper by Hochreiter (1997), is used to maintain and update a hidden state h_t and a cell state c_t . Here, we follow the method in Shi and Cartlidge (2022) under our more granular LOB event space, where they stack m Continuous-Time LSTM units for encoding the input information, where recall m is the number of LOB event types. This extension to the traditional LSTM structure in Mei and Eisner (2017) allows the hidden state parameters to now be specifically computed for each of our LOB event types, where the standard approach previously shared these parameters. The general update mechanism for this LSTM system, as used in our analysis, can be described as follows:

$$i_t = \sigma(W_i[x_t, h_{t-1}] + b_i) \quad (\text{Input Gate}) \quad (7a)$$

$$f_t = \sigma(W_f[x_t, h_{t-1}] + b_f) \quad (\text{Forget Gate}) \quad (7b)$$

$$g_t = f_t \odot c_{t-1} + i_t \odot \tanh(W_g[x_t, h_{t-1}] + b_g) \quad (\text{Target Cell State}) \quad (7c)$$

$$\delta_t = \exp\{W_\delta[x_t, h_{t-1}] + b_\delta\} \quad (\text{Decay Rate}) \quad (7d)$$

$$c_t = g_t + (c_{t-1} - g_t) \odot \exp\{-\delta_t \Delta t\} \quad (\text{Cell State}) \quad (7e)$$

$$o_t = \sigma(W_o[x_t, h_{t-1}] + b_o) \quad (\text{Output Gate}) \quad (7f)$$

$$h_t = o_t \odot \tanh(c_t) \quad (\text{Hidden State}) \quad (7g)$$

where:

- i_t , f_t , and o_t are the input, forget, and output gates. Intuitively, the input gate determines how much of the new target cell state should be added to the memory cell, the forget gate decides how much of the previous cell state should be retained or forgotten, and the output gate is then a filtered version of the memory cell state.
 - g_t is the target cell state, which is the update proposed to the memory cell based on the current input and the previous hidden state.
 - c_t is the memory cell, which stores long-term information by combining the previous memory cell and the new target cell state.
 - h_t is the hidden state, which is the output of the LSTM cell, which is passed to the next time step as in a regular RNN.
 - x_t is the input at time t , which includes information about the type of event, the market state, and the time step.
 - The activation functions, σ and \tanh , represent the sigmoid and hyperbolic tangent functions, respectively.
 - \odot represents the element-wise Hadamard product of two matrices.
- **Output (LOB Event Intensity Function):** Based on the LSTM RNN structure described, the intensity function in Equation (2) can now be reformulated as follows:

$$\lambda_i(t) = \phi_i(h_i(t)) = \ln \left(1 + e^{h_i(t)} \right). \quad (7)$$

Similarly to Equation (2), each $\lambda_i(t)$ jumps discontinuously and drifts towards a stable baseline value λ_i , but in the Neural Hawkes process setup, however, these dynamics are governed by a hidden state vector $h(t) \in (-1, 1)^D$ as defined in Equation (7g), where each events intensity $\lambda_i(t)$ is determined by the corresponding component $h_i(t)$. The influence of each LOB event on the intensity function is captured by the memory cell vector $c(t)$ in the LSTM RNN. More specifically, the LSTM input and forget gates

defined in Equations (7a)-(7b) determine how past event are remembered or forgotten. The input gate determines how much influence new events have on the cell state, similar to how α_{ij} scales the contribution of past events in a Hawkes process but in a more dynamic way. The decay function in Equation (7d) is learned dynamically, allowing it to adapt to different contexts, unlike the fixed decay β_{ij} in a standard HP. Meanwhile, the forget gate determines how much of the past memory is preserved before decay is applied. Lastly, it is clear that ϕ is represented by the Softplus function, which is used to perform the nonlinear transformation, and this ensures that each LOB event’s predicted intensity function is nonnegative. Due to the m -stacked LSTM structure, each LOB event now also has a unique intensity based on its specific latent dynamics.

- **Training Phase:** Here, the model is trained over 20 epochs, where we use batch sizes of 256 and a rolling window approach over each time step and sequence lengths of 100. As in Shi and Cartlidge (2022), our loss function will primarily focus on the loss arising from the Hawkes dynamics, which is defined by the following negative log-likelihood function,

$$\mathcal{L} = \sum_{j=1}^{J-1} \sum_{i=0}^m \log(\lambda_i(t_{j+1})) - \int_{t_j}^{t_{j+1}} \lambda_i(s) ds, \quad (8)$$

over the likelihood of event times t_1, t_2, \dots, t_J , where J represents the total number of event times. The parameters are then updated using the standard RMSprop (Root Mean Square Propagation) adaptive gradient-based optimization algorithm, where a learning rate of 0.002 is used. The accuracy rate over each batch is also tracked for evaluation.

	AAPL	AMZN	GOOG	INTC	MSFT
Training					
Loss	1.7739	1.6952	2.6181	-0.4972	-0.6151
Accuracy	0.4588	0.4401	0.3778	0.5675	0.5489
Testing					
Loss	1.5426	1.5662	2.76	-0.2038	-0.5639
Accuracy	0.4054	0.3904	0.3153	0.4969	0.5124

Table 2: Training and Testing Results in AAPL, AMZN, GOOG, INTC and MSFT from the LOBSTER sample datasets.

In Table 2, we show the results for the training and test sets in the 5 assets studied (AAPL, AMZN, INTC, GOOG, MSFT). Here, we show the value of the loss function, as computed using Equation (8), and the accuracy rate, which measures the probability of correctly predicting the next event in the batch. Table 2 shows us that the training results generalize to the testing results quite well in this setting. In Figure 2, we also provide a visualization of the training results in all five assets, where we show how the loss function and accuracy evolved over the first 20 epochs. It is clear that training improved these measures as the loss decreased and the accuracy increased, and that the generalization error from training to testing is relatively low in this setting. As can be seen by the shape of loss function, it is clear that after 20 epochs more training would likely not improve the results significantly and would more likely just lead to an

overfit model. It is clear that the best results were in INTC and MSFT, which coincidentally is also the case in the results in Shi and Cartlidge (2022) for their model with 4 LOB events.

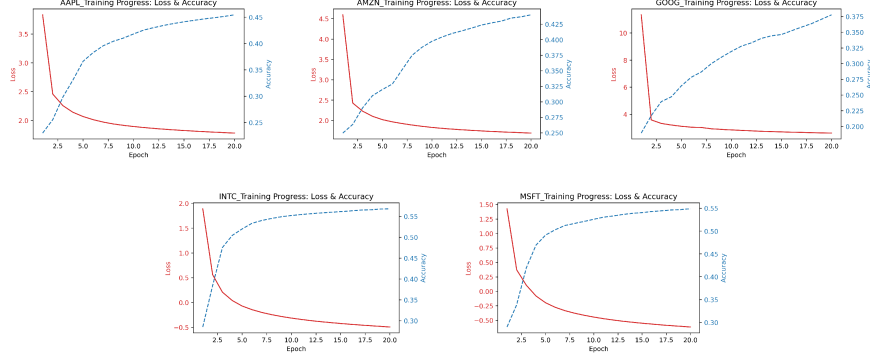


Figure 2: A visualization of the loss function and accuracy during the training phase in AAPL, AMZN, GOOG, INTC and MSFT, where the loss (red) and the accuracy (blue) is shown over 20 epochs.

Now, based on the results of the training phase, the 12 LOB events in our study can be simulated. Here, we adopt the widely used approach for simulating MVHPs: Ogata’s thinning algorithm, as first derived in Lewis and Shedler (1979), is a rejection sampling method used to simulate non-homogeneous Poisson processes. Ogata’s thinning algorithm was first applied to simulating MVHPs in Liniger (2009) and was also used for simulation purposes in the original Neural Hawkes paper by Mei and Eisner (2017), as well as in Shi and Cartlidge (2022) for simulating their set of four LOB events. The algorithm first simulates an inhomogeneous Poisson process with an upper-bound intensity and then selectively accepts events based on its conditional intensity function. The waiting time, Δt , follows an exponential distribution, that is, $\Delta t \sim \text{Exp}(\Lambda_t)$, where here $\Lambda_t = \sum_{i=1}^m \lambda_i(t)$, and $m = 12$ represents the number of LOB events. This has the probability density function $P(\Delta t) = \Lambda(t)e^{-\Lambda(t)\Delta t}$, where the expected time until the next event is $\mathbb{E}[\Delta t] = \frac{1}{\Lambda(t)}$. This is then sampled using the standard method $\Delta t = -\frac{\log(U)}{\Lambda(t)}$, where $U \sim \text{Uniform}(0, 1)$. The probability of each type of LOB event i occurring is then modeled as,

$$P(i|\text{LOB event at } t) = \frac{\lambda_i(t)}{\Lambda(t)}. \quad (9)$$

Intuitively, it is now quite obvious that LOB events with a higher intensity are more likely to be chosen.

In our setting, we ran 200 simulations over each batch. Table 3, above, presents the cumulative frequency of each LOB event in the simulation, while Figure 3, below, provides a visualization of their occurrence over time in the first simulation. The results in Table 3 show that non-aggressive events (with exponent 0), occur more often than aggressive events (with exponent $+/-$), which is in line with the real data in Table 1. While the simulation still exhibits a higher proportion of aggressive events relative to non-aggressive events compared to the real data, the overall event distribution remains informative for analysis. Thus, it is clear that the model has learned which events are more likely to occur, but still with some inaccuracies, as also clearly shown from the training and testing results in Table 2. Note, however, that this type of Neural Hawkes model has been proven to beat the standard benchmarks of traditional

Event-Type	Stock Ticker				
	AAPL	AMZN	GOOG	INTC	MSFT
LB^+	2,790	1,984	2,781	1,884	2,290
LS^-	3,092	1,640	2,841	1,836	2,239
MB^+	3,111	1,764	2,625	1,937	2,534
MS^-	2,469	1,572	2,639	1,966	2,211
BC^-	2,265	1,532	2,675	1,852	2,241
SC^+	2,451	1,773	2,430	1,917	2,314
LB^0	4,417	7,953	8,324	7,998	5,917
LS^0	7,653	11,008	7,235	8,788	8,336
MB^0	4,173	1,954	2,866	1,878	5,783
MS^0	2,806	1,488	3,543	4,694	2,359
BC^0	3,663	5,711	5,925	6,474	8,380
SC^0	12,310	12,821	7,316	9,976	6,587

Table 3: The total number of occurrences of each LOB event type over 200 simulations in AAPL, AMZN, GOOG, INTC, MSFT.

Hawkes models in Shi and Cartlidge (2022), thus we certainly still believe this model to be an improvement on many of the previous Hawkes LOB models in the literature, particularly given its event-based structure.

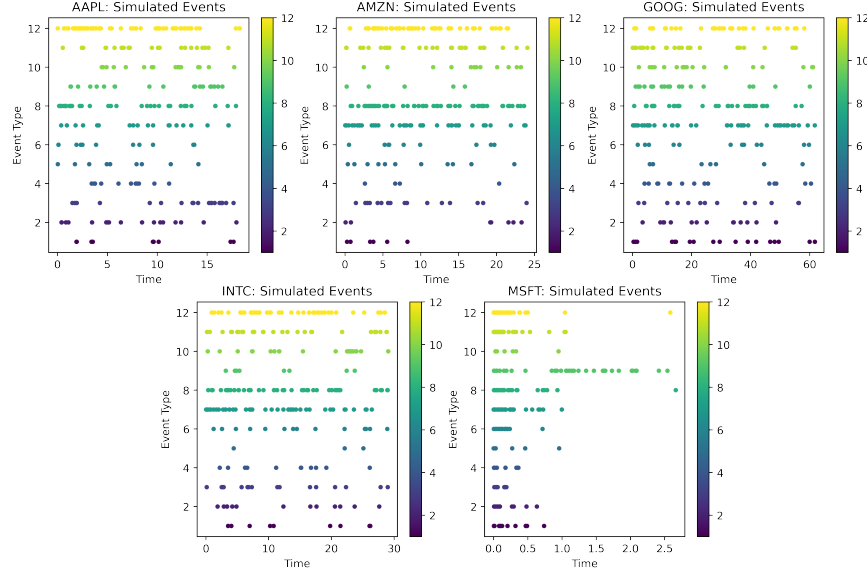


Figure 3: A visualization of how often each LOB event occurred over the first of the two hundred simulations conducted in AAPL, AMZN, GOOG, INTC and MSFT, where each color-coded dot represents the occurrence of a specific event. The y-axis values, ordered from 1 to 12, are in the same order as the events in Table 3.

3.2 Midprice Simulation Process

In mathematical finance, midprice processes (often referred to simply as price processes) are typically simulated by discretizing continuous-time differential equations that incorporate diffusion, pure jump, jump-diffusion, or other pricing model dynamics. Some famous examples of models used to simulate asset prices include using an Euler discretization of the famous Geometric Brownian Motion

model based on Black and Scholes (1973), which may also include jumps as in Merton (1976), then Euler-Maruyama or Milstein methods are also often used (see Kloeden et al. (1992), as well as many other practical formats via Monte Carlo simulation. In much of the Algorithmic and HFT literature, similar simulation techniques are commonly employed. The most widely used approaches, as discussed in the recent survey paper by Jain et al. (2024), include point process-based models where zero-intelligence models like Poisson processes and basic Hawkes process models with dependency structures are the most popular. This survey paper also touches on agent-based models and deep learning-based simulation models. However, most of these simulation techniques overlook the event-based nature of LOB data, which limits their effectiveness in simulating HFT strategies, where granular LOB information is crucial for making informed decisions. We found a previous study, Gašperov and Kostanjčar (2022), that simulates an event-based LOB model within a high-frequency market-making (MM) strategy framework using a Linear MVHP. As noted earlier, this approach has certain limitations that the Neural Hawkes model aims to overcome. In the high-frequency MM domain, an agent-based model has also been developed, as shown in Kumar (2024), where a Neural Hawkes model is used to capture interactions between different trading agents in the market. Our work, however, focuses on the distinct event-based structure of LOB data, which can be difficult for an agent-based model to fully capture without modeling a broad range of agent behaviors at a highly granular level.

To simulate the midprice process, we use Equation (3) as defined in Section 2.2. Here, it is clear that the process must start with an initial midprice $V(0)$, where we will use the first midprice value in the data. Each subsequent midprice movement is then determined by our event-based intensity function, computed through our Neural Hawkes simulation process. More simply, we can redefine Equation (3) in a simpler form as needed for a simulation. Here, at each iteration, the price at $V(t + \Delta t)$ is calculated as follows,

$$V(t + \Delta t) = V(t) + \text{sgn}(\Delta V(t))|\Delta V(t)| \quad (10)$$

where recall Δt is a non-uniform time step based on the interarrival times of the LOB events. Here, $\text{sgn}(\Delta V(t))$ represents the sign of the price jump that occurs for each unique LOB event. From the LOB events defined in Section 2.1., we know that if the LOB event simulated by the intensity function is in the set O_u , the jump size is positive i.e., $\text{sgn}(\Delta V(t)) = 1$. Similarly, if the LOB event simulated by the intensity function is in the set O_d or O_n , $\text{sgn}(\Delta V(t)) = -1$ or $\text{sgn}(\Delta V(t)) = 0$, respectively.

Next, we must define the jump size, $\Delta V(t)$, for each LOB event that has just occurred in order to simulate a midprice process. Recall from Equation (3) that the size of the jumps reflects a mapping from the current state to a price change, where the mapping represents the type of price change. For this particular simulation, we sample our jump sizes, $\Delta V(t)$, from a discrete distribution and this can include n countably finite different jump sizes, where each jump size has a unique probability of occurring, where n would be interpreted as the total number of different jump sizes. Thus, we define the probability of a jump size being a particular size as follows,

$$P(\Delta V(t) = x) = \begin{cases} p_1, & x = j_1, \\ p_2, & x = j_2, \\ \vdots & \vdots \\ p_n, & x = j_n, \end{cases} \quad (11)$$

where p_i and j_i represent the probability and size of each jump that can occur,

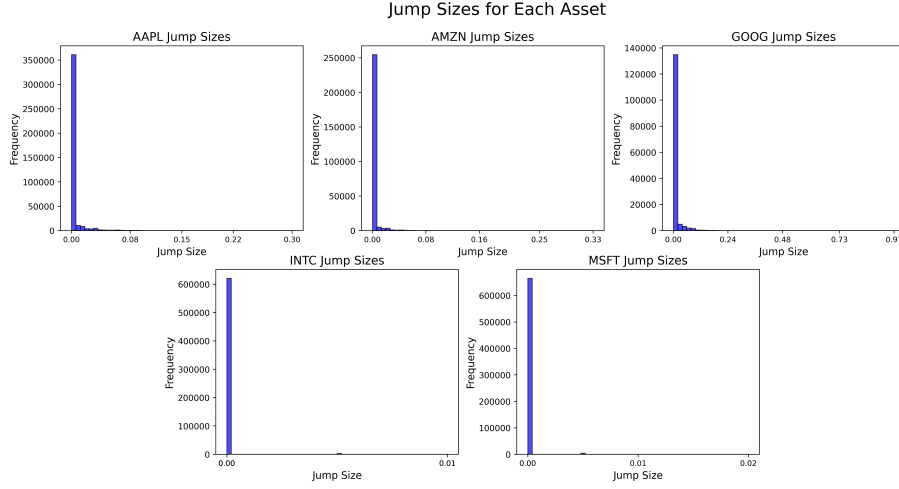


Figure 4: A histogram of jump sizes in each asset studied, showing the extreme heavy right-skewed nature of jump sizes.

respectively. With real LOB data, this can vary significantly depending on the financial asset being modeled. For example, there are many assets where a large portion of the jump moves are 1 tick, and a very small portion of jump sizes have tick size 2, 3, 4, ..., etc. Thus, these jump distributions are often heavily skewed, and each simulation process must accurately reflect this for the particular asset being modeled/simulated. See evidence of this in Figure 4 from the data we studied, where one can also see the total number of different jump sizes in Table 4. It is important to note that no jump size value can be deemed impossible for a particular asset, but above a certain size is extremely unlikely in most market regimes; thus, we ignore this case. An agent or practitioner stress-testing a strategy under this type of model should consider scenarios that allow for more extreme jump events, either by using simulated data or employing datasets with higher volatility than the one used in our study.

	AAPL	AMZN	GOOG	INTC	MSFT
Jump sizes (n)	50	47	91	2	3
Volatility (Real)	0.00005	0.0001	0.00011	0.00018	0.00016
Volatility (Sim)	0.00005	0.0001	0.0002	0.0001	0.0002
Absolute Skewness (Real)	0.0685	0.1276	0.0267	0.1014	0.0908
Absolute Skewness (Sim)	0.1688	0.3134	1.6717	0.0144	0.0555
Excess Kurtosis (Real)	6.6918	17.7054	35.396	-1.9814	-1.9156
Excess Kurtosis (Sim)	6.0606	13.3366	33.8665	-1.9896	-1.9413
Hurst Exponent (Real)	0.3929	0.3428	0.3415	0.1874	0.2846
Hurst Exponent (Sim)	0.5759	0.6521	0.5707	0.3274	0.5583

Table 4: Number of different jump sizes in each asset on June 21st, 2012, along with stylized statistics for the log returns in the real and simulated data.

In Figure 5 we provide a visualization of how each price path evolved in all five assets, and in Table 4 one can find a comparison of some common high-frequency asset price metrics between this simulated data and the real data. It is pretty clear from the results in Table 4 that the simulated data

was able to capture very similar measures in volatility and excess kurtosis, but there were measurable differences in the absolute skewness and Hurst Exponent values. Thus, we can conclude that our simulated data effectively captures broader characteristics such as overall volatility and the fat-tailed nature of the distribution. However, it performs less accurately in finer details, such as matching the observed price range and distinguishing between trending and mean-reverting market behavior.

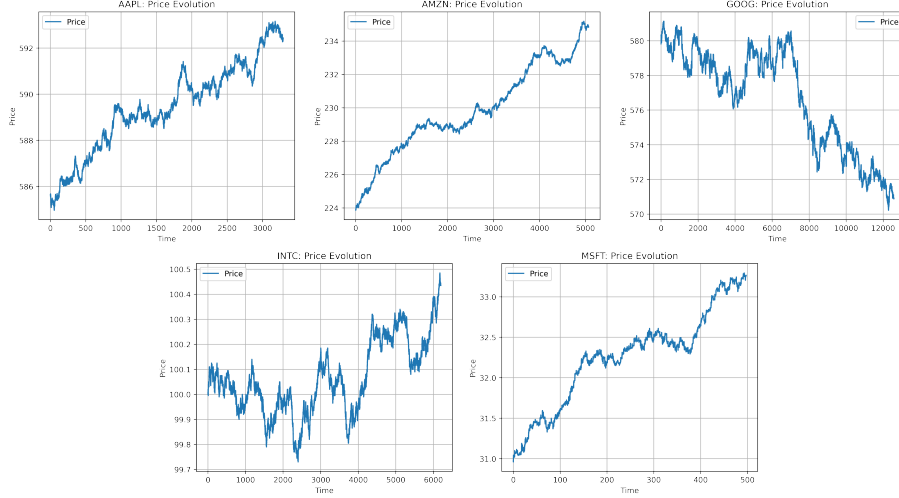


Figure 5: The evolution of the midprice process in each asset based on the previous LOB event simulations.

4 Application: Deep Reinforcement Learning Market-Making Problem

In this Section, we study how one can back-test a Market-Making style strategy under the previous LOB simulation process, where we will also compare the results with what would have occurred under the real data. One of the major advantages of using an event-based LOB simulation process is that we can now match trade order fills with times they actually occurred, which is not possible under the commonly used diffusion, pure jump, or jump-diffusion models where these types of LOB events are either not tracked or deemed to be independent of the price process. For example, if an MM has a limit order posted at the best bid, we now know a trade can only take place if a Market Order enters the market, which would be recognizable via the two previously defined Sell Market Order types, MS^- and MS^0 . Thus, our MM simulation ensures that a trade order fill only occurs when two parties are present, reflecting the fundamental requirement of real market interactions.

To briefly summarize a MM strategy, consider a market participant who posts limit orders at the bid and ask prices. For simplicity, we assume these orders are placed at the best bid and ask prices. The MM is then seen as providing liquidity to the market as this encourages and enables other market participants to trade at current prices. Large MM players often receive fees from the exchanges for doing this, thus this type of activity can generate positive rewards. In saying that, the MM is still exposed to market risk, thus it is in their best interest to figure out when it is optimal to have limit orders posted in the LOB. In terms of optimization, the goal of the MM is to maximize their

terminal reward subject to certain constraints that they may have. These types of constraints often include meeting certain risk measures, such as a maximum position size. In this section, we will explore a common optimization-type MM problem under a deep RL framework, where we aim to compute the optimal limit order posting strategy. This optimal strategy will be largely determined by the LOB model in Section 3, as price movements and the timing of the MM’s limit order fills are two of the main factors influencing the strategy’s overall performance.

In recent times, deep RL has become one of the more popular optimization approaches to solving these types of Algorithmic and HFT Trading problems. Previously, using Stochastic Optimal Control theory was a common approach in the literature, where popular works include the Bertsimas and Lo (1998) optimal execution framework, the Bouchard et al. (2011) general impulse approach, the Cartea et al. (2015) textbook which devotes a full chapter to many variations of the MM problem, Guéant (2017) by giving a very thorough theoretical overview of the optimal MM problem, and Cartea et al. (2018b) which studies parts of the adverse selection problem in MM. Deep RL gained popularity as advancements in deep learning made it easier to integrate function approximators into the traditional RL framework. This overcame a major limitation of classical RL, which struggled with large state-action spaces commonly found in MM-style problems. See Lalor and Swishchuk (2024c) for a more in-depth discussion on the advantages of deep RL versus stochastic optimal control. Examples in the literature of some recent deep RL applications in MM include Guéant and Manziuk (2019) using deep neural networks under an Actor-Critic method for optimal MM in corporate bonds, Gašperov et al. (2021) provides an overview of popular deep RL approaches in optimal MM, Gašperov and Kostanjčar (2022) solves a Hawkes-based optimal MM problem under the Soft Actor-Critic (SAC) approach, Kumar (2024) formulates an agent-based deep Hawkes model for high-frequency MM and Lalor and Swishchuk (2024c) uses the SAC method under non-Markov based price processes.

The rest of this section will proceed as follows. In Section 4.1, we will begin by discussing the common MM setup that we studied, specifically describing the main components of the deep RL framework. This will involve defining the stochastic processes involved, along with how the trading strategy will be simulated. Then, in Section 4.2, we will provide simulation results for the MM strategy under our simulated LOB setup, as discussed in Section 3.2, as well as showing how the strategy would have performed under the real data. To the best of our knowledge, this will be the first extension of a Neural Hawkes event-based LOB model applied to an MM strategy to date.

4.1 *Market-Making Setup*

The agent in this setting will be able to post limit orders on the best bid and ask, where their goal will be to maximize their terminal reward subject to an inventory constraint. In a deep RL MM framework, the main variables that must be formulated are the state-action space (to be approximated by a Neural Network) and the reward function. Here, we will follow a similar approach to the deep RL setup in Gašperov and Kostanjčar (2022) and Lalor and Swishchuk (2024c). This type of MM strategy was also studied under the more traditional Stochastic Optimal Control approach in Cartea et al. (2015) and Cartea et al. (2018b). First we will define the state space as follows:

$$S_t = (V_t, Q_t), \tag{12}$$

where V_t is the midprice process given in Equation (3) and Q_t will be the inventory process that satisfies the follow equation:

$$Q_t = N_t^- - N_t^+, \quad (13)$$

where N_t is a counting process for the limit order fills on the bid ($-$) and the ask ($+$). In our model, each of MM's limit order fills must now coincide with the 4 market order events in Table 1. Thus, this trade order fill process logic will be extended to include the event-based nature of a trade order fill.

Trade order fills will be divided into separate non-overlapping counting processes for non-adverse fills and adverse fills, as in Lalor and Swishchuk (2024b) and Lalor and Swishchuk (2024c), but now following our event-based logic. The non-adverse trade order fill process will also attach the same type of non-adverse fill probability, as limit order trade fills are not guaranteed to occur when trades are executed at the agent's limit order price by non-aggressive market orders. This is essentially due to the time-priority nature of LOBs, whereby each limit order is generally sent to the back of a queue of previously submitted limit orders. Thus, only market orders that are greater than or equal to the size of this queue are guaranteed to lead to a limit order fill.

Subsequently, we first define the trade order fill logic for non-adverse trade order fills as follows:

$$NFA_t = \sum_{i=1}^N A_{t_i}^+ \mathbb{I}_{\{MB_{t_i}^0 \neq \emptyset\}} * p, \quad (14)$$

and

$$NFB_t = \sum_{i=1}^N A_{t_i}^- \mathbb{I}_{\{MS_{t_i}^0 \neq \emptyset\}} * p, \quad (15)$$

where NFA_t and NFB_t represent the counting processes for all non-adverse fills that occur on the best ask and best bid, respectively. Here, $A_{t_i}^+$ and $A_{t_i}^-$ denote trades that would've occurred on the best ask/bid, given the non-aggressive market order events, MB^0 and MS^0 , occurred at time t_i . However, the MM agent is not guaranteed to receive these fills so we also attached a non-adverse fill probability, p , which can be defined as,

$$p = P(N_{t_i}^{A,\pm} | A_{t_i}^+ \mathbb{I}_{\{MB_{t_i}^0 \neq \emptyset\}} = 0, A_{t_i}^- \mathbb{I}_{\{MS_{t_i}^0 \neq \emptyset\}} = 0). \quad (16)$$

This represents the probability of getting filled given that a non-aggressive market order has entered the market. Next, adverse fills, where the agent is guaranteed to have their limit order filled if an aggressive market order enters the market, can be defined as follows,

$$AFA_t = \sum_{i=1}^N A_{t_i}^+ \mathbb{I}_{\{MB_{t_i}^+ \neq \emptyset\}}, \quad (17)$$

and

$$AFB_t = \sum_{i=1}^N A_{t_i}^- \mathbb{I}_{\{MS_{t_i}^- \neq \emptyset\}}, \quad (18)$$

where AFA_t and AFB_t represent the counting processes for all adverse fills that occur on the best ask and bid, respectively. The main difference in the trade order fill logic in this work compared to Lalor and Swishchuk (2024b) and Lalor

and Swishchuk (2024c) is that now each trade order fill is based on the event-based process of trade orders entering the market, rather than being purely based on simulated price movements from some approximated price process. Intuitively, this new trade order fill logic now states that the agents limit orders will always be filled by aggressive market orders (MB^+ and MS^-), but only sometimes by non-aggressive market orders (MB^0 and MS^0), which depends on the non-adverse fill probability.

Next, define the action space as follows,

$$A_t = \begin{cases} \{0, 1\}, & \text{if } Q_t = -q, \\ \{-1, 0, 1\}, & \text{if } -q < Q_t < q, \\ \{-1, 0\}, & \text{if } Q_t = q. \end{cases} \quad (19)$$

where we can see that the MM agent can choose whether or not to post limit orders at the best bid/ask, subject to an inventory constraint q . Thus, the agent cannot have a long or short position larger than the absolute value of q .

Next, the reward function must be defined, in order to measure the performance of the deep RL MM strategy. Here we will use the commonly defined reward/value function used in the MM literature as follows,

$$\mathbb{E}_\pi \left[W_T - \psi \int_0^T |Q_t| dt \right], \quad (20)$$

where W_t represents the total wealth which can be defined as $W_t = Q_t V_t + C_t$, where C_t represents the cash process. Here, ψ is called the inventory penalty coefficient, which penalizes inventories greater than zero. A MM agent generally prefers to keep their positions in the market low or large only for a very short period, thus, this will encourage the model to close out any large positions as quickly as possible.

Lastly, we will briefly describe the Neural Network architecture used in approximating the State-Action space. As previously stated, we use the Soft Actor-Critic (SAC) approach, which is an off-policy deep RL algorithm that maximizes the expected reward while encouraging exploration through entropy regularization, using two Q-networks, a stochastic policy, and a automatic parameter adjustment for balancing exploitation and exploration. We used the Stable Baselines3 python package to streamline this, which was recently developed by Raffin et al. (2020). See Haarnoja et al. (2018) for the seminal paper on this algorithm and Lalor and Swishchuk (2024c) for a more in-depth introduction on how to apply this in a MM setup as we use the same approach.

4.2 Asset Specific Results - Simulated vs Real Data

In this Section, we will discuss the results that pertain to an asset-specific approach, where each set of results is based on the LOB dynamics learned from the data of each specific asset. We conducted a comparative analysis between the output of the simulated midprice process (based on the Neural Hawkes process framework described in Section 3) and the real LOBSTER (2025) data. In both cases, the deep RL market-making model was trained separately on each of the five assets. From this analysis, we are able to assess the cumulative performance results on unseen data for each particular asset and we can analyze how the major market order event types impacted the MM agent’s performance.

First, see Table 5, where we show the parameter values used in our Deep RL MM model. To start with, we set the maximum inventory to 5 units to prevent the agent’s position from becoming too large. Smaller positions are further

encouraged by the inventory penalty parameter, where we set $\psi = 0.001$. Each trade is of size 1, represented by $dQ = \pm 1$. Then we set the non-adverse fill probability to 0.2, which is in line with the empirical results in Lalor and Swishchuk (2024b), where they computed this probability based on trade order fill data on some of the most liquid futures contracts listed on the Chicago Mercantile Exchange. Lastly, the training and testing sample datasets, denoted X_{train} and X_{test} , were performed on the first 5,000 and the subsequent 2,500 samples of LOB event and price data, respectively, for both the simulated and real datasets.

Parameters			
Parameter	Value	Parameter	Value
q	5	dQ	± 1
p	0.2	ψ	0.001
X_{train}	5,000	X_{test}	2,500

Table 5: Deep RL optimal MM simulation parameters.

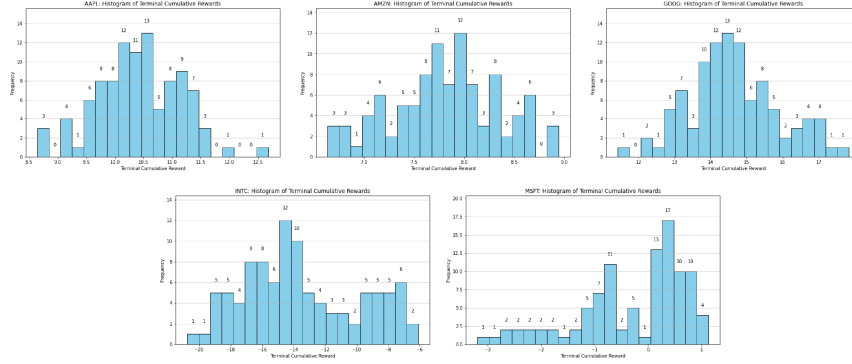


Figure 6: A histogram of the terminal cumulative reward over each testing episode from the **simulated** data.

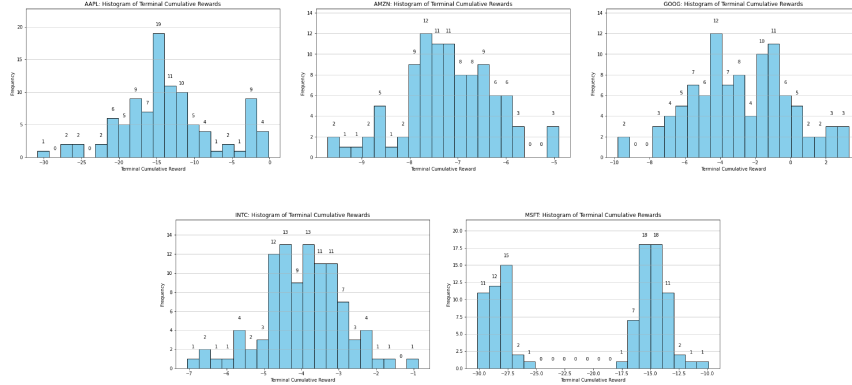


Figure 7: A histogram of the terminal cumulative reward over each testing episode from the **real** data.

We begin by presenting the key cumulative performance results from this analysis, which in RL is often done by analyzing the output of the reward func-

tion at the terminal time T . Figures 6 and 7 illustrate the cumulative terminal rewards in all 100 testing episodes, for both the simulated and real data. Here, it is clear that the cumulative rewards are mostly negative for both sets of data, with some slight differences in how these terminal value outcomes were distributed. This still represents an improvement over the initial results observed during the training period, similar to the findings in Lalor and Swishchuk (2024c), where the MM agent successfully learned to achieve a positive or a less negative reward. However, it is pretty clear that a simple strategy, such as one purely focused on finding optimal times to post limit orders on the best bid/ask, is unlikely to be profitable on its own. The secretive nature of high-performing MM strategies makes it tough to improve on more traditional MM setups, but the main point here is to show how our event-based LOB model can be applied in a real-world setting.

As noted previously, the performance of an MM strategy is often heavily determined by how and when the MM agent receives limit order trade fills and, in particular, how these trade order fills affect the general market exposure of the MM agent. Given our event-based framework, we can now analyze which of the previously defined market order types, as shown in Table 1, matched with the MM’s agent’s posted limit orders. More specifically, Figures 8 and 9 illustrate the frequency with which each type of market order filled the agent’s posted limit orders across all testing episodes, in both simulated and real data, respectively. It is clear that the distribution of the MM agents limit order trade fills is much more likely to come from aggressive market orders rather than non-aggressive market orders, and this is very similar in both the simulated and real data. See also Table 6, where one can also see that the ratio between adverse and non-adverse trade order fills were very similar in each asset, as well as being significantly weighted toward adverse fills. This essentially means that after the MM agent receives a new trade order fill, the resulting position will be out of the money at the next time step more often than not. In reality, it would be in the MM agent’s best interest to avoid these adverse limit order fills. However, as numerous studies have shown, including Lalor and Swishchuk (2024b), effectively mitigating them remains a significant challenge.

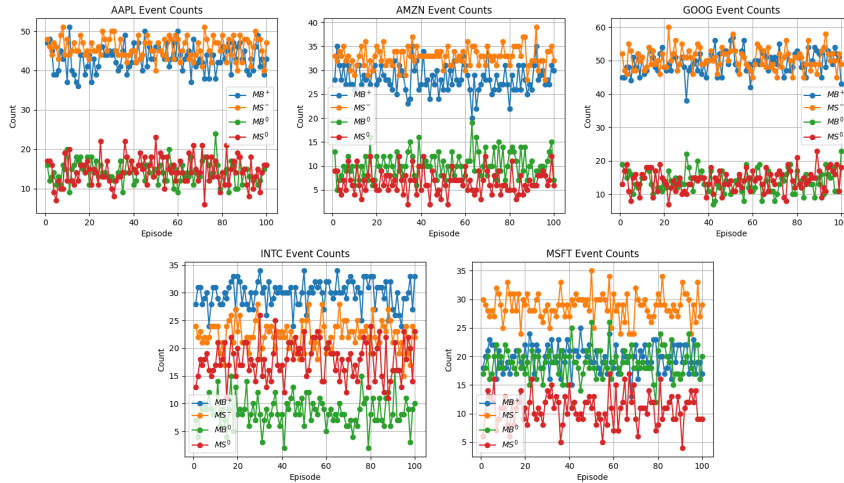


Figure 8: The number of times each Market Order type resulted in a trade order fill for the MM agent in each testing episode in the **simulated** data.

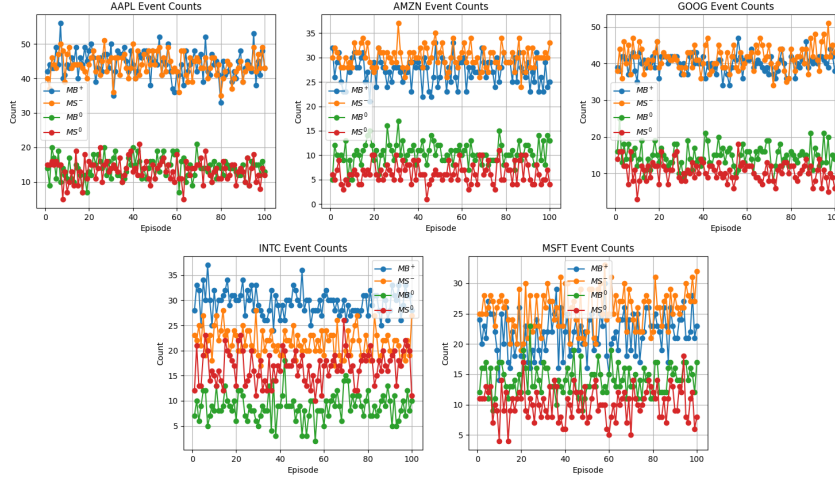


Figure 9: The number of times each Market Order type resulted in a trade order fill for the MM agent in each testing episode in the **real** data.

	AAPL	AMZN	GOOG	INTC	MSFT
Ratio (Real)	3.2093	3.4203	3.2342	2.0004	1.9510
Ratio (Sim)	3.0871	3.4626	3.4151	1.9708	1.6075

Table 6: The ratio of the MMs adverse to non-adverse trade order fills for each asset across all the Neural Hawkes simulation and real data testing episodes.

5 Conclusions and Future Recommendations

In this research, we developed an event-based Neural HP for simulating asset price process at a high-frequency level. Granular LOB information is essential for HFT strategies like for a MM, and we believe our model takes strides in improving analysis in this area. More specifically, we developed an event-based LOB model that takes into account 12 of the main events that appear in the LOB, where each event’s intensity was modeled via a nonlinear MVHP. This nonlinear MVHP model was then approximated via the Neural HP, which helps to overcome many of the limitations present in the traditional HP models. Our empirical results show that many of the broad dynamics seen in the simulated data are in line with the real data, as seen by the volatility and excess kurtosis measures, while the model still struggles with some of the finer details, evidenced by the absolute skewness and Hurst exponent measures. However, the resulting simulated midprice process was then applied with a deep RL MM framework, where very similar results were achieved under the simulated and real data. In this setting, the MMs trade order fills can now be specifically calculated from the LOB event activity observed, which is more in line with how a high-frequency MM assesses market conditions in the real world.

In terms of future research recommendations, we recommend digging deeper to improve the accuracy of the predicted simulated midprice processes, so that it can align even more closely with the real LOB data. Although our Neural Hawkes setup yielded similar results for the simple MM strategy across both simulated and real data, enhancing the model to better distinguish between

aggressive and non-aggressive orders would lead to a more accurate representation of the high-frequency dynamics observed in LOB data. A practitioner may also want to get even more specific with the set of LOB events, by extending the LOB model to include more than the 12 events currently included. This should be achievable if the LOB event’s impact on the midprice process and the dynamics of its intensity function can be mathematically formulated in a reasonably simple way.

Acknowledgments

The authors thank MITACS and NSERC for research funding.

Declarations of Interest

The authors declare that they have no conflicts of interest.

References

- Abergel, F. and Jedidi, A. (2015). Long-time behavior of a hawkes process-based limit order book. *SIAM Journal on Financial Mathematics*, 6(1):1026–1043.
- Bacry, E., Mastromatteo, I., and Muzy, J.-F. (2015). Hawkes processes in finance. *Market Microstructure and Liquidity*, 1(01):1550005.
- Bertsimas, D. and Lo, A. W. (1998). Optimal control of execution costs. *Journal of financial markets*, 1(1):1–50.
- Black, F. and Scholes, M. (1973). The pricing of options and corporate liabilities. *Journal of political economy*, 81(3):637–654.
- Bouchard, B., Dang, N.-M., and Lehalle, C.-A. (2011). Optimal control of trading algorithms: a general impulse control approach. *SIAM Journal on financial mathematics*, 2(1):404–438.
- Cartea, A., Donnelly, R., and Jaimungal, S. (2018a). Enhancing trading strategies with order book signals. *Applied Mathematical Finance*, 25(1):1–35.
- Cartea, Á., Jaimungal, S., and Penalva, J. (2015). *Algorithmic and high-frequency trading*. Cambridge University Press.
- Cartea, A., Jaimungal, S., and Ricci, J. (2018b). Algorithmic trading, stochastic control, and mutually exciting processes. *SIAM review*, 60(3):673–703.
- DeLise, T. (2024). The negative drift of a limit order fill. *arXiv preprint arXiv:2407.16527*.
- Gašperov, B., Begušić, S., Posedel Šimović, P., and Kostanjčar, Z. (2021). Reinforcement learning approaches to optimal market making. *Mathematics*, 9(21):2689.
- Gašperov, B. and Kostanjčar, Z. (2022). Deep reinforcement learning for market making under a Hawkes process-based limit order book model. *IEEE control systems letters*, 6:2485–2490.
- Gould, M. D., Porter, M. A., Williams, S., McDonald, M., Fenn, D. J., and Howison, S. D. (2013). Limit order books. *Quantitative Finance*, 13(11):1709–1742.

- Guéant, O. (2017). Optimal market making. *Applied Mathematical Finance*, 24(2):112–154.
- Guéant, O. and Manziuk, I. (2019). Deep reinforcement learning for market making in corporate bonds: beating the curse of dimensionality. *Applied Mathematical Finance*, 26(5):387–452.
- Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. (2018). Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pages 1861–1870. PMLR.
- Hochreiter, S. (1997). Long short-term memory. *Neural Computation MIT-Press*.
- Jain, K., Firoozye, N., Kochems, J., and Treleaven, P. (2024). Limit order book simulations: A review. *arXiv preprint arXiv:2402.17359*.
- Kloeden, P. E., Platen, E., Kloeden, P. E., and Platen, E. (1992). *Stochastic differential equations*. Springer.
- Kumar, P. (2024). Deep hawkes process for high-frequency market making. *Journal of Banking and Financial Technology*, pages 1–18.
- Lalor, L. and Swishchuk, A. (2024a). Algorithmic and high-frequency trading problems for semi-Markov and Hawkes jump-diffusion models. *arXiv preprint arXiv:2409.12776*.
- Lalor, L. and Swishchuk, A. (2024b). Market simulation under adverse selection. *arXiv preprint arXiv:2409.12721*.
- Lalor, L. and Swishchuk, A. (2024c). Reinforcement learning in non-markov market-making. *arXiv preprint arXiv:2410.14504*.
- Law, B. and Viens, F. (2019). Market making under a weakly consistent limit order book model. *High Frequency*, 2(3-4):215–238.
- Lewis, P. W. and Shedler, G. S. (1979). Simulation of nonhomogeneous poisson processes by thinning. *Naval research logistics quarterly*, 26(3):403–413.
- Liniger, T. (2009). *Multivariate hawkes processes*. PhD thesis, ETH Zurich.
- LOBSTER (2025). LOBSTER: Limit Order Book Reconstruction and Visualization. Accessed: 2025-01-30.
- Lu, X. and Abergel, F. (2018). High-dimensional hawkes processes for limit order books: modelling, empirical analysis and numerical calibration. *Quantitative Finance*, 18(2):249–264.
- Mei, H. and Eisner, J. M. (2017). The neural hawkes process: A neurally self-modulating multivariate point process. *Advances in neural information processing systems*, 30.
- Merton, R. C. (1976). Option pricing when underlying stock returns are discontinuous. *Journal of financial economics*, 3(1-2):125–144.
- Raffin, A., Hill, A., Gleave, A., Kanervisto, A., Ernestus, M., and Dormann, N. (2020). Stable baselines3. GitHub repository.
- Roldan Contreras, A. and Swishchuk, A. (2022). Optimal liquidation, acquisition and market making problems in HFT under Hawkes models for LOB. *Risks*, 10(8):160.

- Shi, Z. and Cartlidge, J. (2022). State dependent parallel neural hawkes process for limit order book event stream prediction and simulation. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 1607–1615.
- Swishchuk, A. and Huffman, A. (2020). General compound Hawkes processes in limit order books. *Risks*, 8(1):28.
- Swishchuk, A., Remillard, B., Elliott, R., and Chavez-Casillas, J. (2019). Compound Hawkes processes in limit order books. In *Financial Mathematics, Volatility and Covariance Modelling*, pages 191–214. Routledge.