Limit Order Book Simulation with Generative Adversarial Networks

Rama Cont[1], Mihai Cucuringu[1,2,3], Jonathan Kochems[4], and Felix Prenzel*[1,4]

[1]Mathematical Institute, University of Oxford
[2]Department of Statistics, University of Oxford
[3]The Alan Turing Institute
[4]JP Morgan†

July 16, 2023

Abstract

We propose a nonparametric method for simulating the dynamics of a limit order book using Generative Adversarial Networks (GAN) to learn the conditional distribution of the future state of the order book given its current state from time series of the limit order book. Our method yields a scenario generator for limit order books which captures a range of stylized facts and salient properties of limit order book transitions. We show that the trained generator is also able to correctly reproduce some key properties observed in empirical studies on market impact. In particular, the model exhibits a decaying marginal impact of trade size, higher impact of aggressive orders, as well as a decreasing relation between impact and order book depth.

Contents

*Corresponding author: prenzel@maths.ox.ac.uk

†Opinions expressed in this paper are those of the authors, and do not necessarily reflect the view of JP Morgan.

1 Introduction

Most exchanges nowadays organize trading activities via a centralized limit order book (LOB). These LOBs are records of outstanding limit orders (LOs) indicating the willingness of an agent to either buy or sell a certain quantity of an asset at a certain price. Many different agents with different objectives and trading styles interact in the LOB, thus leading to complex dynamics. Such dynamics and their statistical properties have been extensively studied in the literature. Models have been built to capture and reproduce the observed dynamics. In particular, synthetic data is of high value to practitioners, allowing them to build more robust execution algorithms, and also more rigorously test the performance of various models and trading strategies.

In most settings, either event-based or queue-based models have been developed in the literature. In the setting of ultra-high frequency applications, it is indispensable to model the entire event stream, since algorithms act on these single events. However, for many practical applications in mid-to-high frequency settings, it suffices to model the order book snapshots at certain regular time intervals.

In this study, we address the modeling of time series of LOB snapshots which has received less attention in the literature. More precisely, we leverage Generative Adversarial Networks (GANs) in order to learn the conditional probability distribution of the transition between two consecutive LOB snapshots, and thereby implicitly the price changes in the LOB. The aim is to match both unconditional as well as conditional metrics.

1. Unconditional metrics: Using the trained GAN to generate synthetic data, are unconditional stylized facts recovered?
2. Conditional metrics: The primary aim of a generative LOB model is to interact with it in order to model the effect of the interaction on the system. Hence, interacting with the LOB state should have similar effects as studied in the literature. This particularly concerns the market impact of executions, including the well known square root law (Gatheral, 2010). This is studied via both limit order submission (liquidity provision) as well as the submission of market orders (liquidity extraction).

Our proposed approach to model LOB snapshots is able to reproduce a number of statistical properties of typical LOB snapshot time series, such as distributions, correlation structures, and price dynamics. Additionally, the price process is implicitly in line with the quantity process due to the design of the network and its input structure. Finally, the model – once trained to a sufficient accuracy – is able to reproduce commonly known patterns from the market impact literature. Interacting with the LOB state shows effects such as the expected market impact and the square root law (Gatheral, 2010) when changing the quantity of injected orders. We find this particularly interesting as the model is not explicitly being trained to do so, and furthermore, it only learns the dynamics of the LOB snapshots. This work opens many future research directions for improvement of LOB snapshot simulation, mostly regarding the inclusion of more history, stabilization of the model convergence, support of sparse LOB snapshots for empty levels and intraday effects.

The remainder of the paper is structured as follows. Section 2 presents related work from the LOB literature. Limit order books and the object modeled is presented in Section 3. Section 4 reviews Wasserstein Generative Adversarial Networks and outlines the design for the model in the present study. Unconditional metrics of stylized facts are presented in Section 5 and results of interaction with the simulator are shown in Section 6. Section 7 concludes and describes further extensions of this research direction.

2 Related work

Every action in a LOB on an exchange is recorded and stored. The availability of such LOB data for a large number of years has led to a vast amount of research articles. Broadly speaking, there are three main goals in this literature, namely (1) empirical studies explaining the dynamics, (2) forecasting certain future quantities of the LOB, and (3) modeling the dynamics of LOBs. One particular point of interest in these studies is also the price impact of trades in LOBs.

An overview of limit order book modeling is given in (Cont, 2011; Gould et al., 2013). Empirical studies on properties of LOB include (Cont, 2001; Bouchaud, Mézard, Potters, et al., 2002; Potters & Bouchaud, 2003; Cont, 2011). Due to its large relevance for industrial applications, prediction of future price changes in LOBs and trading volumes have been conducted. More recently, (J. A. Sirignano, 2019; J. Sirignano & Cont,

2019; Zhang, Zohren, & Roberts, 2019) have used large amounts of LOB data in order to predict future price movements using different neural network architectures.

The main focus of many studies in the literature is the development of methods for modeling the dynamics in limit order books on an *event* base. To this end, researchers predominantly used point processes, namely Poisson processes for example in (Cont, Stoikov, & Talreja, 2010; E. Smith, Farmer, Gillemot, Krishnamurthy, et al., 2003; Huang, Lehalle, & Rosenbaum, 2015). While order flow modeled via Poisson processes does not perfectly reproduce the market dynamics, in particular with respect to inter-arrival times and cross-excitation behaviour, its simplicity allows one to derive interesting quantities. For instance, (Cont et al., 2010) derive analytical solutions for the probability of a price increase, the execution of a limit order before a mid price move, and the execution of both a buy and sell limit order before a mid price move. To account for dynamics dependent of the queue size, (Huang et al., 2015) introduce a queue-reactive model with arrival intensities dependent on the queue state. Hawkes processes (Hawkes, 1971) have been used in (Abergel & Jedidi, 2013, 2015; Morariu-Patrichi & Pakkanen, 2022) to model the order flow in order to account for the clustering and excitation behavior in LOBs. (Morariu-Patrichi & Pakkanen, 2022) introduce state dependency to better model dependence properties of the order flow conditioned on either spread or order imbalance. On the other end, the computer science community has developed agent-based model which assume different types of agents that act together in the LOB (Paddrik et al., 2012; Byrd, Hybinette, & Balch, 2019). The main difficulties with agent-based models consist of defining the agents and calibrating them such that the joint simulation leads to realistic markets. That is because each single agent type is generally interpretable and tractable. However, the joint dynamics are usually not which makes calibration a challenging task. (Paddrik et al., 2012) is particularly interesting due to the fact they use agent types and calibrations backed by an empirical study of the flash crash (Kirilenko, Kyle, Samadi, & Tuzun, 2017). (Cont, Cucuringu, Glukhov, & Prenzel, 2023) attempt to shed light into different types of agents analyzing and modeling the order flow of a major broker.

All the previously mentioned studies model the queue sizes on an order-by-order basis. Alternatively, (Cont & Müller, 2021) model the LOB state as a density which evolves over time via a stochastic partial differential equation (SPDE). The SPDE is meant to explain the net changes of the density over time instead of creating a discrete order stream which leads to changes in the LOB. In particular, they assume that events in LOBs may be summarized into a net change in the order book. Each factor in the SPDE represents certain properties, such as high-frequency traders which lead to fluctuations similar to Gaussian noise or traders replacing their orders closer to the mid price.

With recent advances in machine learning and increased availability of computational resources, the focus has recently shifted to more data-driven methods. In particular, generative adversarial networks (GANs), introduced in (Goodfellow et al., 2014) have gained a lot of attention for synthetic data generation, mostly in the context of speech and image data. The GAN framework consists of two neural networks competing against each other, thereby improving the quality of the generated data until a Nash equilibrium is reached. Properly trained, GANs have shown the ability to learn complex structures in the data. Since their inception, a large body of work has been conducted to improve well known issues regarding their training instability (Arjovsky, Chintala, & Bottou, 2017; Gulrajani, Ahmed, Arjovsky, Dumoulin, & Courville, 2017; Mescheder, Nowozin, & Geiger, 2017; Mescheder, Geiger, & Nowozin, 2018; C.-L. Li, Chang, Cheng, Yang, & Póczos, 2017). Most commonly known are variations of the Wasserstein GAN (WGAN) (Arjovsky et al., 2017), which uses the Wasserstein distance approximation via the Kantorovich-Rubinstein duality to measure the distance between the data and the generated distribution. (Mirza & Osindero, 2014) have first introduced conditional GANs in order to condition generated samples on variables (e.g. one-hot encoding of digits).

The ability of generative networks to learn complex distributions and generate samples from them, has drawn significant attention from academics and practitioners in the financial area. GANs have been applied in 1-dimensional settings, for instance in (Wiese, Knobloch, Korn, & Kretschmer, 2020; Da Silva & Shi, 2019; Takahashi, Chen, & Tanaka-Ishii, 2019; K. E. Smith & Smith, 2020; Ni, Szpruch, Wiese, Liao, & Xiao, 2020). In particular, (Wiese et al., 2020) design a generator architecture including temporal convolutional neural networks outperforming conventional GARCH models. In the multidimensional setting, (Wiese, Bai, Wood, & Buehler, 2019) deploy GANs to generate time series discrete local volatility surfaces. Hybrid approaches have been presented in (Marti, 2020; Prenzel, Cont, Cucuringu, & Kochems, 2022). (Marti, 2020) learn to generate realistic correlation matrices with GANs. (Prenzel et al., 2022) generate synthetic calibrations of order flow models introducing new dynamics into conventional, analytically tractable models. With respect to direct modeling of LOBs, (J. Li, Wang, Lin, Sinha, & Wellman, 2020; Coletta, Moulin, Vyetrenko, & Balch, 2022) deploy a conditional WGAN to generate the event stream in an order book, conditioned on the previous order stream. Each order is a tuple of price, quantity, type of the order, time since past order, ± 1 side of the book. However, in these studies GANs are used to output discrete values (e.g. the type of the order) while

they are generally designed to learn continuous probability distributions.

This work aims to model the net transition of the order book state using GANs. More precisely, we design and train a probabilistic model to learn the distribution of future LOB states given some state as condition. To the best of our knowledge, this is the first study to achieve this, and to design a model with realistic market impact patterns.

3 Problem setting

This section formulates the problem we are dealing with. It introduces limit order books and outlines which particular representation of it this study aims too model.

3.1 Limit Order Books

Limit order books are the mechanism in which electronic stock trading is primarily organized. A limit order describes the willingness to buy a certain quantity of an asset at a pre-specified price.

Definition 3.1 (Limit Order)**.** A limit order $x = (t, p, q)$ is characterized by

- an arrival time $t \in \mathbb{R}^+$,
- a price $p \in \delta\mathbb{N}$ which is a multiple of the price tick $\delta > 0$,
- a quantity $q \in \mathbb{Z} \setminus \{0\}$ with $q < 0$ denoting buy orders and $q > 0$ denoting sell orders.

A limit order is considered "outstanding" as long as it has neither been cancelled nor fully executed.

A limit order sits in the book until it is either canceled by the agent that has submitted the order or executed against a market order.

Definition 3.2 (Limit Order Book)**.** At any time $t \in \mathbb{R}^+$ the set of all outstanding orders $\mathcal{L}(t)$ is the limit order book. E.g. for some $\mathcal{L}(t^*)$, $t_x \leq t^* \; \forall \; x \in \mathcal{L}(t^*)$.

Many market participants without access to more detailed data sets see the LOB in forms of queue sizes which build the cumulative demand supply for a certain price at a certain time (see (Cont et al., 2023) for more information on different views of the LOB).

Definition 3.3 (Queue Size)**.** For a given limit order book $\mathcal{L}(t)$ at time $t \in \mathbb{R}^+$ for a certain price $p \in \delta\mathbb{N}$ is denoted as

$$Q_p(t) = \sum_{\mathcal{L}_p(t)} q_x, \tag{1}$$

where $\mathcal{L}_p(t) = \{x \mid x \in \mathcal{L}(t), p_x = p\}$. The queue size is the cumulative amounts of shares the market is willing to buy (sell). The LOB viewed as queue sizes, $Q(t)$, takes values in $\mathbb{Z}^{\delta\mathbb{N}}$. The collection of the queue sizes is also known as a LOB snapshot.

Similar to the signs of single orders (Definition 3.1), a positive queue size indicates a sell (ask) queue and a negative queue size indicates a buy (bid) queue. The best ask price denotes the lowest price a sell order exists in the LOB, i.e.

$$p_a(t) = \inf_p \{p > 0, Q_p(t) > 0\}. \tag{2}$$

Vice versa, the best bid price denotes the highest price a buy order in the LOB, i.e.

$$p_b(t) = \sup_p \{p > 0, Q_p(t) < 0\}. \tag{3}$$

The so-called mid-price is then just the average between bid and ask price, $p_{mid}(t) = \frac{p_a(t)+p_b(t)}{2}$. For no arbitrage reasons, $p_a(t) > p_b(t)$ holds at any time, as well as $Q_p(t) \geq 0 \; \forall \; p > p_a(t)$, and $Q_p(t) \leq 0, \; \forall \; p < p_b(t)$.

A limit order book snapshot is then the collection of queue sizes, usually for $k \in \mathbb{N}$ levels around the mid price. An exemplary centered LOB snapshot is depicted in Figure 1.

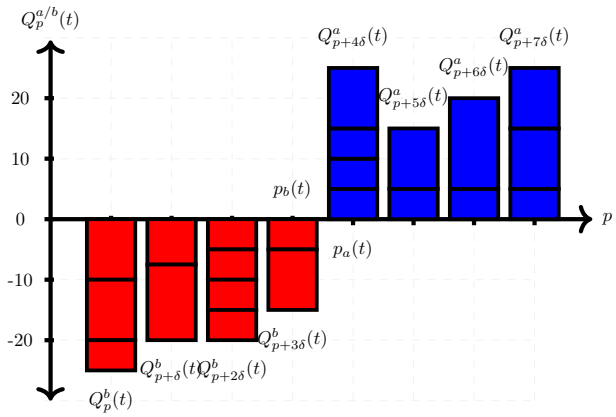


Figure 1: Arbitrary LOB snapshot indicating the queues' resulting from several orders. Buy queues (red) are negative, sell queues (blue) are positive.

Definition 3.4 (Centered LOB snapshot)**.** For some price grid of length $2k$, i.e. $(p, p+\delta, \dots, p+(2k-1)\cdot\delta)$, the centered order book at time t is defined as

$$X_t = \left(\underbrace{Q_p(t), \dots, Q_{p+(k-1)\cdot\delta}(t)}_{\leq 0 \rightarrow \text{ bid}}, \underbrace{Q_{p+k\cdot\delta}(t), \dots, Q_{p+(k-1)\cdot\delta}(t)}_{\geq 0 \rightarrow \text{ ask}} \right) \tag{4}$$

$X_t \in \mathbb{Z}^{2k}$. At every t, the price grid is chosen such that $Q_{p+i\delta}(t) \leq 0 \; \forall \; i \in \{0, \dots, k-1\}$ and $Q_{p+i\delta}(t) \geq 0 \; \forall \; i \in \{k, \dots, 2k-1\}$, hence "centered" LOB snapshot.

In this representation, $Q_{p+(k-1)\delta}(t)$ and $Q_{p+(k)\delta}(t)$ correspond to the best bid (ask) queue at the corresponding best bid (ask) price

$$p_b(t) = p + (k-1)\delta(t), \quad \text{and} \quad p_a(t) = p + k\delta(t).$$

The above holds under the assumption that queues are not empty, i.e. $Q_p(t) \neq 0$.

3.2 Conditional Order Book Density

Market participants' actions lead to (1) submission, (2) cancellation and (3) execution of limit orders in the LOB, which are discrete events in continuous time. Each of these actions leads to one change in the book. Its cumulative change over some appropriate time span Δt may be seen as a continuous transition of the different queue sizes from state X_t to $X_{t+\Delta t}$. In the present work, we aim to model the distribution of the centered order book state at time $t + \Delta t$. The future LOB snapshot for the same fixed price grid $(p, p+\delta, \dots, p+(2k-1)\cdot\delta)$ takes one of the following three forms.

1. No price change: The sign of the queue at price p, $Q_p(t)$, remains the same if $|Q_p(t+\Delta t) - Q_p(t)| < |Q_p(t)|$, i.e. the change is smaller than the queue size. If there is no price change, $X_{t+\Delta t}$ has the same expression as in Equation (4), but at time $t + \Delta t$. The future state has the following form

$$X_{t+\Delta t} = \left(\underbrace{Q_p(t+\Delta t), \dots, Q_{p+(k-1)\cdot\delta}(t+\Delta t)}_{\leq 0 \rightarrow \text{ bid}}, \underbrace{Q_{p+k\cdot\delta}(t+\Delta t), \dots, Q_{p+(k-1)\cdot\delta}(t+\Delta t)}_{\geq 0 \rightarrow \text{ ask}} \right). \tag{5}$$

2. Price decrease: For a price decrease, one or more bid queues have to turn positive, thereby turning into ask queues. In case of Equation (4), at least $Q_{p+(k-1)\cdot\delta}(t+\Delta t)$ would be positive, turning it into an ask quantity. A price decrease of one tick (δ) leads to the following shape of the LOB snapshot

$$X_{t+\Delta t} = \left(\underbrace{Q_p(t+\Delta t), Q_{p+\delta}(t+\Delta t)}_{\leq 0 \rightarrow \text{ bid}}, \underbrace{Q_{p+2\cdot\delta}(t+\Delta t), Q_{p+3\cdot\delta}(t+\Delta t), Q_{p+4\cdot\delta}(t+\Delta t), Q_{p+5\cdot\delta}(t+\Delta t)}_{\geq 0 \rightarrow \text{ ask}} \right). \tag{6}$$

Similarly, a price change of k ticks would be induced by a sign change of k - former - bid quantities.

3. Price increase: Vice versa, for a price increase, one or more bid quantities have to turn negative, thereby turning into bid queues. E.g., for a one tick price increase, $Q_{p+k\cdot\delta}(t+\Delta t)$ has to turn negative and for $k=3$ the order book state would read

$$X_{t+\Delta t} = \left(\underbrace{Q_p(t+\Delta t), Q_{p+\delta}(t+\Delta t), Q_{p+2\cdot\delta}(t+\Delta t), Q_{p+3\cdot\delta}(t+\Delta t)}_{\leq 0 \rightarrow \text{ bid}}, \underbrace{Q_{p+4\cdot\delta}(t+\Delta t), Q_{p+5\cdot\delta}(t+\Delta t)}_{\geq 0 \rightarrow \text{ ask}} \right). \quad (7)$$

The advantage of the LOB representation in Equation (4) is that price changes are inherently modeled via the process of the LOB snapshots. In particular for large tick stocks, price changes are discrete and generally a few ticks, for an appropriate Δt. This is because typically only few queues get depleted and filled by either the same side (no price change) or the other side (price change). Hence, Equation (4) is much more natural than, for example, adding the price change as an additional, discrete variable. Furthermore, price changes are by design consistent with the queue size process. At the same time, it is important to consider that at most k price changes during Δt can be modeled. This is important to keep in mind when choosing k and Δt and should be chosen appropriately depending on the particularities of each individual stock. Especially for small tick stocks, one may want to consider binning several levels into a single one.

As mentioned above, we are interested in the distribution of future centered LOB snapshots $X_{t+\Delta t}$, and aim to draw samples from the distribution

$$X_{t+\Delta t} \sim \mathbb{P}_r(x_{t+\Delta t}), \quad (8)$$

where subscript r refers to the probability distribution of the data assumed to be the "real" distribution. Clearly, LOB snapshots depend on each other, as certain realizations of $X_{t+\Delta t}$ are more likely than others depending on the current state X_t (and possibly depending on much larger history, as for instance, (J. Sirignano & Cont, 2019) find longer path dependency in LOBs). Modeling the unconditional distribution $\mathbb{P}_r(x_{t+\Delta t})$ is thus not sufficient.

Instead, a more realistic target is to learn the conditional distribution of centered LOB snapshots at $t+\Delta t$,

$$X_{t+\Delta t}|S_t \sim \mathbb{P}_r(x_{t+\Delta t}|s_t) \quad (9)$$

where $S_t \in \mathbb{R}^s$ is some state which serves as a condition for the order book sample.

Such a condition S_t can take many forms, but primarily should contain

1. Recent history in the form of previous lagged order book states

   $$X_t, X_{t-\Delta t}, \ldots, X_{t-h\cdot\Delta t};$$

   such states may be compressed with a suitable encoding function

   $$f(X_t, X_{t-\Delta t}, \ldots, X_{t-h\cdot\Delta t}),$$

   or form a raw condition.
2. "Global" conditions such as time of day, value of the volatility index VIX, etc.

Note that in a Markovian setting, S_t may just be X_t or a compression of X_t.

4 Generative Adversarial Networks

Sampling from the distribution in Equation (9) is not straightforward because the distribution is most likely not tractable and entails complex dependencies. Generative adversarial networks may be used as a data-driven method to learn to draw samples from a generated distribution $\mathbb{P}_g(x_{t+\Delta t}|s_t)$ which is similar to the data distribution $\mathbb{P}_r(x_{t+\Delta t}|s_t)$.

In generative modeling, one is confronted with "real" (denoted with subscript r) data X following a probability measure $\mathbb{P}_r$, i.e., $X \sim \mathbb{P}_r$ and $X \in \mathbb{R}^d$, where d denotes the dimension of the input data. The aim is to generate samples $\tilde{X} \in \mathbb{P}_g$, where $\mathbb{P}_g$ is the induced distribution of some generative model such that $D(\mathbb{P}_r, \mathbb{P}_g) \to 0$ for some distance or divergence measure D. In many applications, such as image or speech generation, this task is fairly difficult due to the high dimensionality and complexity of $\mathbb{P}_r$. In the setting of time series, this particularly applies to the dependencies on the past and other conditions.

4.1 Wasserstein GAN

GANs in their original representation from (Goodfellow et al., 2014) tackle this task via a 2-player mini-max game in which two agents/models compete against each other:

1. **Generator:** The generator is given a random noise seed and maps it into the sample space. The generated samples should be real enough so that the discriminator cannot distinguish between real and fake samples. More precisely:

   Definition 4.1 (Generator (g))**.** Let $Z \sim \mathbb{P}_z, Z \in \mathbb{R}^z$ be a sample drawn from some prior/seed and $\Theta^{(g)}$ the parameter space of g (usually a set of trainable weights of a neural network). The generator g is then a function parameterized by $\theta^{(g)} \in \Theta^{(g)}$, transforming the noise seed Z from the latent space $\mathbb{R}^z$ into the sample space $\mathbb{R}^d$

   $$\begin{aligned} g : \Theta^{(g)} \times \mathbb{R}^z &\to \mathbb{R}^d, \\ (\theta^{(g)}, Z) &\mapsto g_{\theta^{(g)}}(Z). \end{aligned} \tag{10}$$

2. **Discriminator:** The discriminator/critic is given real and generated/fake data samples and aims to efficiently map real and fake data into a decision space.

   Definition 4.2 (Critic (f))**.** Let $X \in \mathbb{R}^d$ be either $\sim \mathbb{P}_r$ or $\sim \mathbb{P}_g$, and $\Theta^{(f)}$ the parameter space of f (e.g. trainable weights of a neural network). The critic f is a function parameterized by $\theta^{(f)} \in \Theta^{(f)}$, such that

   $$\begin{aligned} f : \Theta^{(f)} \times \mathbb{R}^d &\to \mathbb{R}, \\ (\theta^{(f)}, X) &\mapsto f_{\theta^{(f)}}(X). \end{aligned} \tag{11}$$

In its original implementation, the discriminator performs a binary classification on real and fake samples, and the min max training corresponds to the minimization of the cross entropy of $\mathbb{P}_r$ and $\mathbb{P}_g$. In this paper, we mostly the Wasserstein GAN (Arjovsky et al., 2017; Gulrajani et al., 2017).

(Arjovsky et al., 2017) first make use of the Wasserstein-1 distance which is defined as

$$W(\mathbb{P}_r, \mathbb{P}_g) = \inf_{\gamma \in \Pi(\mathbb{P}_r, \mathbb{P}_g)} \mathbb{E}_{(x,y)\sim\gamma} \left[\|x - y\| \right]. \tag{12}$$

The reason for this is to provide a continuous and differentiable distance between two probability measures – the real and the generated probability distribution. To deal with the intractability of the infimum in (12), the Kantorovich-Rubinstein Duality is used as the objective function to approximate (12). This dual problem of the W1 distance reads

$$\max_{\|f\|_L \leq 1} \mathbb{E}_{\mathbb{P}_r}[f(x)] - \mathbb{E}_{\mathbb{P}_g}[f(x)] \tag{13}$$

where x in the first part is the real data and and in the second part is the generated data. f is some 1-Lipschitz function maximizing the expression in Equation (13). In the Wasserstein GAN setting this is done with the critic Definition 4.2. The model, generally dubbed as the *critic* as defined in Definition 4.2, is trained to approximate the dual of the Wasserstein-1 distance. Note, it must be ensured that f is 1-Lipschitz in order for the dual problem to hold.

Finally, both d and g, precisely their parameters $\theta^{(d)}$ and $\theta^{(g)}$, are trained via simultaneous optimization (also called alternating gradient descent) in a joint min-max game:

$$\min_{\theta_{(g)}} \max_{\|f\|_L \leq 1} \mathbb{E}_{\mathbb{P}_r}[f_{\theta^{(f)}}(x)] - \mathbb{E}_{\mathbb{P}_z}[f_{\theta^{(f)}}(g_{\theta^{(g)}}(z))] \tag{14}$$

where $g_{\theta^{(g)}}(z)$ is the generated data. Intuitively, the critic is trained to best approximate the W1 distance during the training process while generator uses the gradient of the approximated W1-distance in order to improve its sample quality and thereby minimize that distance.

One challenge that arises is the 1-Lipschitz condition of the dual problem from Equation (14), which must hold for f for the correct computation of the Wasserstein-1 distance. To this end, (Arjovsky et al., 2017) suggest the rather rigorous approach to clip the weights of the critic, i.e. restrict $\Theta^{(f)} = [-c, c]^l$, with l denoting the number of trainable weights of the critic, and $c \in \mathbb{R}$ denoting a clipping threshold often chosen relatively small (0.01 in (Arjovsky et al., 2017)). This approach, however, runs into the danger of being too restrictive, which may lead to a too restricted critic. This renders the WGAN approach to fail in many simple examples

according to (Arjovsky et al., 2017). To mitigate this issue, (Gulrajani et al., 2017) suggest a gradient penalty which penalizes gradients above 1 for linear combinations between arbitrary points drawn from $\mathbb{P}_r$ and $\mathbb{P}_g$. The updated critic's objective is adjusted with a gradient penalty and reads

$$\max_{\theta^{(f)}} \; \mathbb{E}_{\mathbb{P}_r}[f_{\theta^{(f)}}(x)] - \mathbb{E}_{\mathbb{P}_z}[f_{\theta^{(f)}}(g_{\theta^{(g)}}(z))] - \lambda \mathbb{E}_{\mathbb{P}_{\hat{x}}}\left[(\|\nabla_{\hat{x}} f_{\theta^{(f)}}(\hat{x})\|_2 - 1)^2\right], \tag{15}$$

with the penalty coefficient $\lambda \in \mathbb{R}$ leading to a more efficient enforcement of the Lipschitz condition (Arjovsky et al., 2017). $\mathbb{P}_{\tilde{x}}$ is the distribution of $\tilde{x} = ux + (1-u)g(z)$, where $U \sim U(0,1)$ is a uniformly distributed random variable in the interval $[0,1]$. Note, the penalty is subtracted as the critic is trained to maximize Equation (15).

4.2 GAN Formulation for LOB Simulation

The aim is to learn the probability distribution introduced in (9). As outlined above, unconditional sampling will likely not suffice because the future LOB snapshot will depend on other variables, such as previous snapshots of the LOB but also volatility and time-of-day effects. We therefore use the setting of conditional GANs first introduced in (Mirza & Osindero, 2014) to condition the samples to be drawn on some state $S_t \in \mathbb{R}^s$. Hence, the generator used including the state S_t has the following form

$$\begin{aligned} g : \Theta^{(g)} \times \mathbb{R}^z \times \mathbb{R}^s &\to \mathbb{R}^d \\ (\theta^{(g)}, Z_t, S_t) &\mapsto g_{\theta^{(g)}}(Z_t, S_t) = X_{t+\Delta t}. \end{aligned} \tag{16}$$

The simplest setting for a model is a Markovian system for which will we will present the results in the sequel, hence $S_t = X_t$. Similar to the generator, also the discriminator (critic) is given S_t as input to "judge" whether, given a certain condition, a future state is realistic or not. In case of the Wasserstein GAN this amounts to

$$\begin{aligned} f : \Theta^{(f)} \times \mathbb{R}^d \times \mathbb{R}^s &\to \mathbb{R} \\ (\theta^{(f)}, X_{t+\Delta t}, S_t) &\mapsto f_{\theta^{(f)}}(X_{t+\Delta t}, S_t). \end{aligned} \tag{17}$$

In total, six fully connected layers are used for both g and d (respectively f, in the Wasserstein GAN case). The architecture of the generator is visualized in Figure 2, and entails two fully connected layers, each of which separately transforms Z_t and S_t. The transformations ($h_{2,1}$ and $h_{2,2}$) are then concatenated and fed into two final layers. Finally, the output of the last layer is then the generated order book state $X_{t+\Delta t}$.

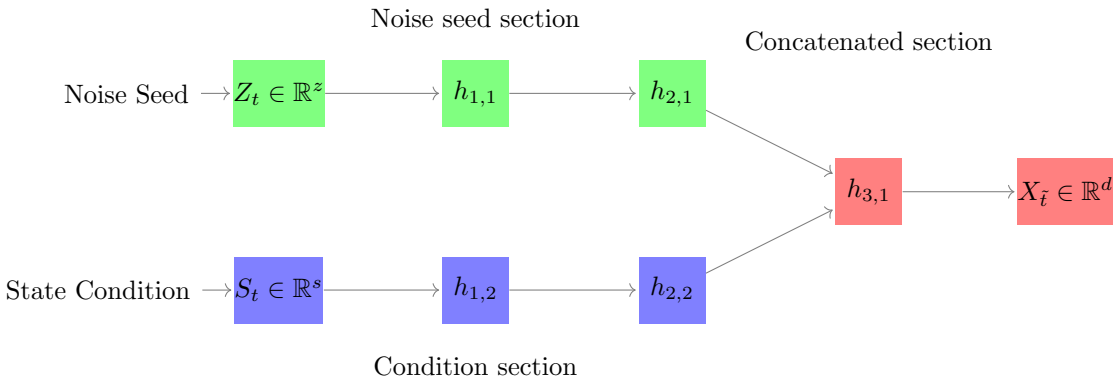


Figure 2: Conditional GAN scheme with three hidden layers of which the first and the second ($h_{1,i}$ and $h_{2,i}$) are separated. $h_{2,1}$ and $h_{2,2}$ are then concatenated to combine the network's noise seed and condition section to generate $X_{\tilde{t}}$.

For g, we use ReLU for hidden layers and linear activation for the output layer because any queue size may be positive (for sell queue) or negative (for a buy queue). Training is performed using the RMSprop and ADAM optimizer. Further hyperparameter specifications can be found in Table 1. In order so stabilize training, we further apply gradient clipping during the training process to the discriminator to avoid fluctuations particularly due to gradient noise in later stages of the training.

Activation Hidden Layers	ReLU
Activation Final Layer	Linear
# of hidden layers	3
Hidden Layer Size	32-64
Δt	10 seconds
Days of data	20 days
# Samples	$\sim 40k$
Optimizer	Adam
Learning Rate	$1e-5$
Iterations	$\sim$ 300k

Table 1: Table indicating attributes of network and data used for training.

5 Empirical results

This section introduces the data used throughout this work, reviews the training process and presents numerical results of (recurrent) simulation without interaction.

5.1 Data set description

The data we used in this research is obtained from lobsterdata.com. The platform provides L3 data with a message file containing the orders sent to an exchange (primarily market, limit and (partial) cancellation orders), as well as a file containing the corresponding order book states.

To train the neural network, 20 days of MU tick data from 2016 are used. MU is a large tick stock which rarely contains empty levels and mostly has a spread equal to the tick size. This makes it rather easy to learn for our approach in comparison to other small tick names, as the GAN does not have to produce zero-inflated distributions with many ticks mostly close to zero.

Limit order books exhibit non-stationary patterns, particularly at the beginning and at the end of the trading day. We therefore discard the first and last 30 minutes of the continuous trading session, in order to remove market open and closing effects. Setting Δt to 10 seconds results in ∼40k order book transitions for 5.5 hours of trading during 20 days. We choose $k = 3$, in other words, we focus on the first three bid and ask levels. In this case, more than 99.9% of the price changes fall within this range. Furthermore, $4 \cdot \hat{\sigma}_{\Delta t} < 3$ where $\hat{\sigma}_{\Delta t}$ is the volatility of the price changes measured in ticks over 10 seconds (assuming the price behaves like a Brownian motion).

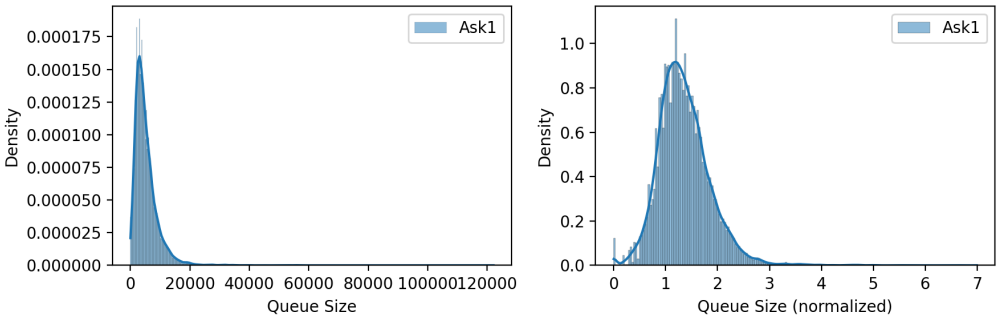


Figure 3: Distribution of queue sizes (left) and normalized queue sizes (right).

As for most data-driven tasks, normalization is crucial. While a log-transformation seems sensible, problems may occur for queue sizes around or equal to zero. Furthermore, due to the input structure of the state Equation (4), the sign of the queue must be preserved. We hence normalize via square root and a normalization constant

$$\tilde{Q}_p(t) = \frac{sign(Q_p(t))\sqrt{|Q_p(t)|}}{c} \tag{18}$$

where

$$sign(Q_p(t)) = \begin{cases} 1, & \text{if } Q_p(t) \geq 0 \\ -1, & \text{if } Q_p(t) < 0 \end{cases} \tag{19}$$

preserves the correct sign of the queue. Furthermore, c is a constant to scale the data into a certain range, which is particularly important for numerical aspects and if one wants to use data from different names. In Figure 3, the normalized distribution is shown on the right hand side.

5.2 Training process

As in every unsupervised learning task, there is no direct metric to observe the quality of the generated samples. In particular, the loss of the entropy GAN does not indicate sample quality, and while the Wasserstein distance in the setting of (Arjovsky et al., 2017; Gulrajani et al., 2017) can indicate the quality over the course of the training, the distance cannot be used to compare different models. Instead, metrics generally found to be important in real data are observed and tracked during the training process to obtain insights into what the model learns as well as how fast and accurately it does so.

For most GAN specifications used in this study, the model learns the metrics quickly and shows some convergence without much tuning. However, some metrics start to fluctuate once they have reached a certain level – especially in the setting of the entropy GAN as introduced first by (Goodfellow et al., 2014). This is particularly critical for the price paths. The shape of the marginal distributions is shown in Figure 5a. Small differences in the probability mass can imply different probability for price increases or decreases can lead to drifts in the generated price paths. Hence, in particular when using a balanced training set, the generated distributions should be as symmetric as possible in order to ensure the stability of the synthetic data. Due to the aforementioned fluctuations, this is not a trivial task to handle. The reason for these fluctuations is potentially based on the GAN structure which, in theory, converges to a saddle point (Nash equilibrium). The iterative gradient descent has as consequence that this saddle point slightly moves around with every step, and hence the GAN never really is quite optimal on a small enough scale, which is enough for these imbalances to manifest. In particular (Mescheder et al., 2017, 2018) analyze these dynamics in more detail. The issue of this symmetry and how to enforce it are addressed in Section 5.4.

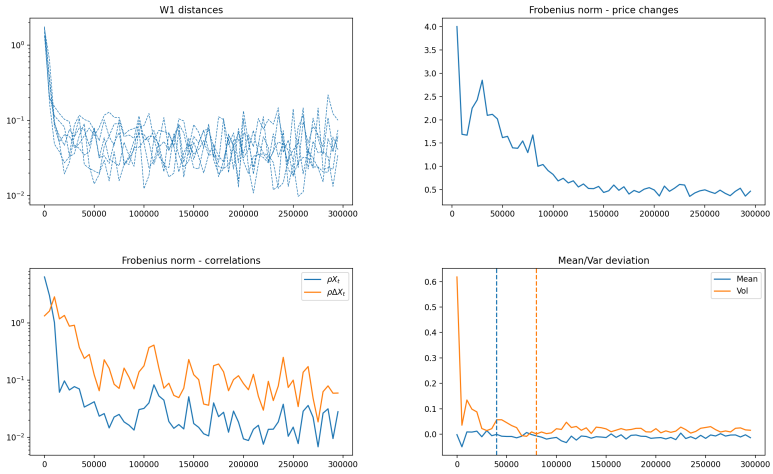


Figure 4: Convergence WGAN-GP with Gradient Clipping.

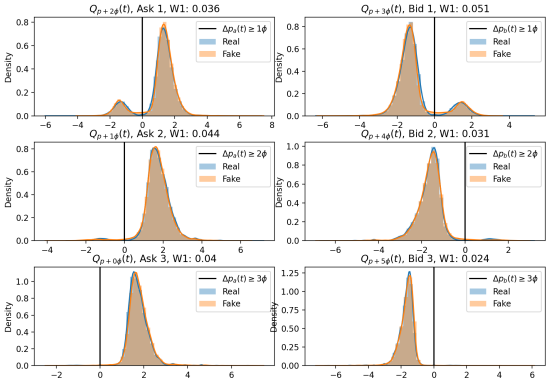
We find WGAN-GP as presented in (Gulrajani et al., 2017) to work best and to lead to the smoothest training process. In particular, the use of gradient clipping of the critic leads to less "extreme" changes at each gradient step. This builds a more consistent base to learn from for the generator. E.g., if a certain gradient becomes too large during optimization, clipping avoids the discriminator/critic from moving too far away. Note, we clip by value not by norm.

Figure 4 shows several tracked metrics during the training process. The upper left plot indicates the W1-distances of the marginal distributions which are learned fairly quickly. This also holds for the deviation of the mean and the volatility from the generated samples. In comparison, these lines were much more volatile for the entropy GAN version and without the use of gradient clipping. Perhaps most interestingly to observe

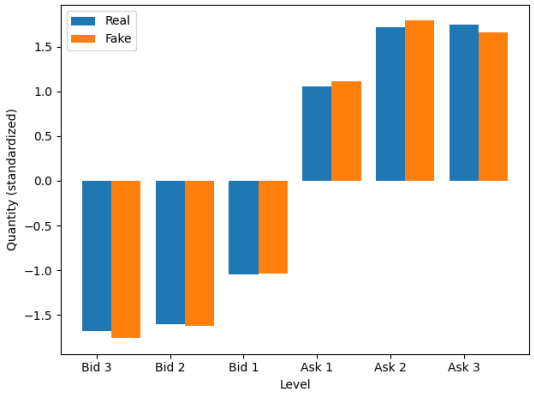
are the conditional price changes, which are only really learned by the GAN over time. This indicates that the GAN first matches the marginal distributions without much effort, and then needs more time to also capture the conditional properties of the LOB snapshots. The next section analyses each of the metrics in more detail.

5.3 Stylized facts

1. **Marginal distributions:** Figure 5a indicates the marginal distributions of $X_{t+\Delta t}$. The shape of the distributions is learned well. For Ask-1 (Bid-1), the negative (positive) values correspond to the samples with price changes, i.e. the queue is depleted and becomes a queue of the opposite sign. The mass of the generated samples is not as large as for the real samples, which indicates fewer generated price changes. The small bumps in Ask2, Bid2 depicted in Figure 5a correspond to price changes of two ticks, which show less probability mass. This indicates that price changes of two ticker (or more) are much less likely to occur.



(a) Marginal distributions $X_{t+\Delta t}$.



(b) Average LOB shape.

Figure 5: Marginal distributions and average order book shape for both real and fake data.

2. **Average centered order book shape**: Figure 5b shows the average LOB shape for the real and the generated data, which coincide very well. Loooking closer, the average queue sizes tend to be slightly larger for ask quantities.

3. **Correlation structure:** The correlation matrices of both the generated order quantities themselves, as well as its changes (e.g. $Q_p(\tilde{t}) - Q_p(t)$), are shown in Figure 6. In particular, the correlation of the actual order book values are very close to those observed in real data. Also the first difference show a large degree of accordance.

4. **Price paths:** We now turn our attention to properties of the price paths using g in recurrent fashion to generate entire time series. In particular, the generated state $X_{t+\Delta t}$ builds the new condition, $S_{t+\Delta t}$, which is then used to generate $X_{t+2\Delta t}$. After a sign change – i.e. price change, the state is re-centered and the shift is stored as the corresponding price change, as explained in Section 3.

   Figure 7 shows multiple real and generated paths, as well as the mean of the generated paths. The initial order book states X_0 are sampled from the data set. It is also possible to use the same initial order book state X_0. Table 2 shows statistics regarding the price changes. The values for real and fake price paths coincide well. For both real and fake data, the drift is close to zero. Some of the mean's deviations are to sample variance. We also observe a slightly higher volatility for generated price paths. One reason for that is the slightly higher percentage of transitions with more than one tick. While the real distribution 200-step returns show significant excess kurtosis, the generated ones do not, indicating the GAN is not producing price paths with fat tails. This is expected as the model has a Markovian structure when setting $S_t = X_t$

   Figure 7b supports previous statements. The center percentiles coincide very well between real and generated price changes over 200 transitions. However, especially the lowest and highest points in the figure – corresponding to the 0% and 100% quantile and hence indicating the minimum and maximum – show significant deviation further. This indicates that the paths generated by the GAN do not show the same tail properties as the real data. Despite the 0% and 100% quantile being very noisy values, this observation is very consistent.

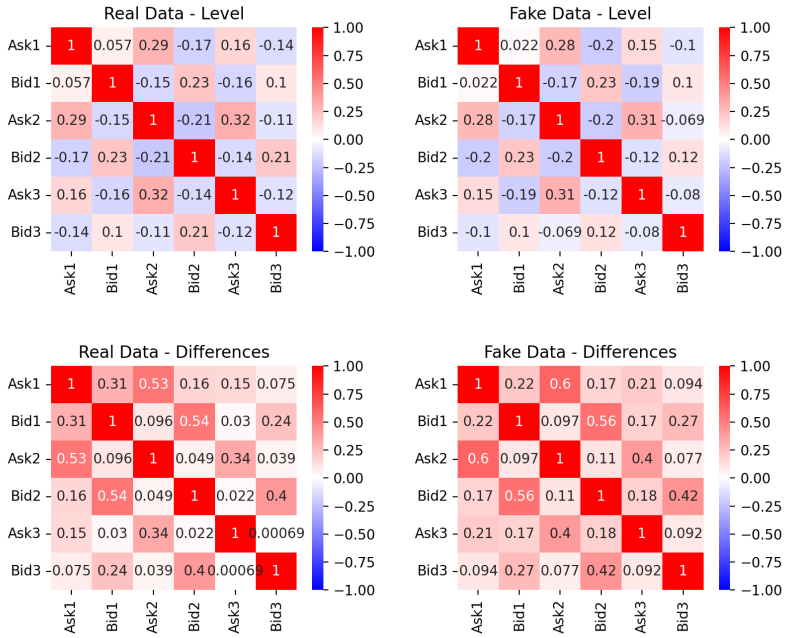
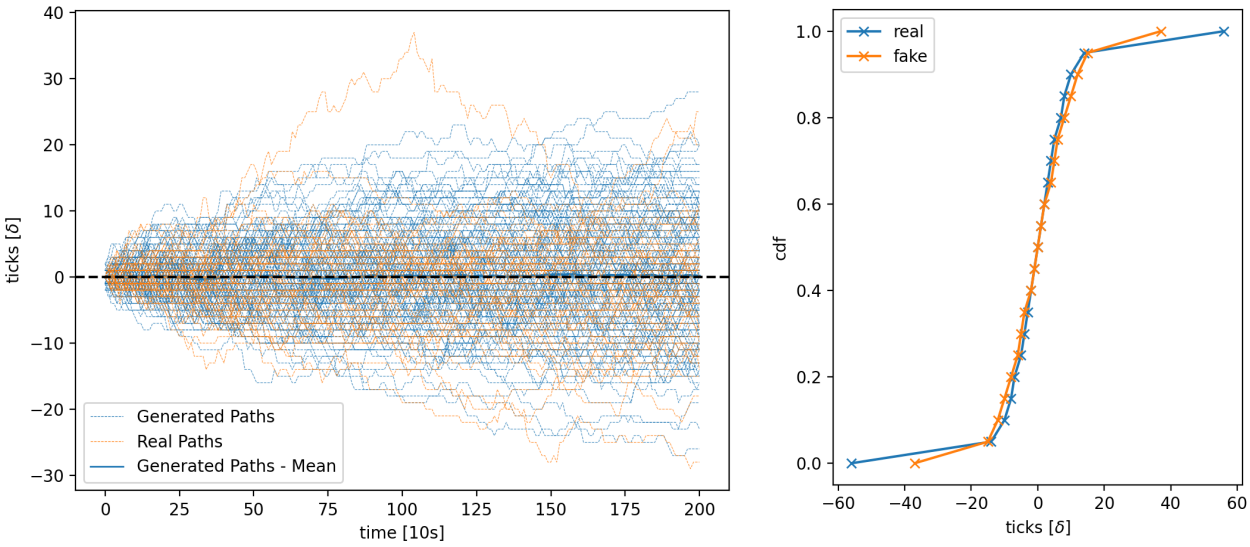


Figure 6: Correlation matrices for real (left) and fake (right) data. The top row pertains to the generated order quantities, while the bottom row to the increments $Q_p(\tilde{t}) - Q_p(t)$.



(a) Real and fake price paths (in ticks).

(b) CDF of $\Delta p_{mid}(200)$ in ticks.

Figure 7: Price paths and corresponding percentiles of terminal prices for real and fake data.

Besides the differences in the tail behavior of the generated paths, the distribution of price changes over 200 transitions is fit quite well. The quantiles in Figure 7b – particularly from 5% to 95% – show very little deviation apart from the mentioned tail behavior. Furthermore, it is to emphasize the price paths show a a surprising fit taking into account of being purely generated by transitions of the order book's queue size process X_t. This indicates proper modeling of the process of queue size transitions can produce realistic price paths.

5. **Conditional price changes:** In the literature, the order imbalance or book imbalance, defined as

$$OI(t) = \frac{Q_{p_b}(t) - Q_{p_a}(t)}{Q_{p_b}(t) + Q_{p_a}(t)}, \tag{20}$$

is well known to have predictive power for future price changes. In particular, the higher the imbalance between bid and ask side of the order book, the more likely is a price change towards the direction of the

	μ	σ	0δ	1δ	-1δ	2δ	-2δ	3δ	-3δ
$\mathbb{P}_r$	-0.24	9.05	0.7373	0.1125	0.1125	0.0156	0.0158	0.0019	0.0020
$\mathbb{P}_g$	-0.55	9.19	0.7214	0.1145	0.1145	0.0194	0.0152	0.0021	0.0027

Table 2: Table summarizing price moves. First two columns indicate 200-step mean and volatility, with the remainder columns showing the frequency of particular price changes.

smaller queue. This can be explained economically by the idea that there is weaker support for the price on the side of the smaller queue. It thus stands to question, whether the model the probability for price changes differs depending on the state and furthermore whether this coincides with what is observed in real data. One quantity to look at is the probability of a positive change of the mid price $p_m(t)$, e.g.

$$\mathbb{P}(p_m(\tilde{t}) > p_m(t) \,|p_m(\tilde{t}) \neq p_m(t), Q_1^a(t) \in [q_k, q_{k+1}], Q_1^b(t) \in [q_l, q_{l+1}]), \tag{21}$$

where $k, l \in \{0, \ldots, 9\}$ indicate the number of the quantile corresponding to 0% to 90% quantile. Figure 8 indicates these probabilities for both real and generated data. For instance, the upper left square indicates the probability for a price change to be positive if both bid and ask queues are in the bucket between the corresponding 0% and 10% quantile.

Figure 8 shows that the generator, solely learning the process of the queues in the order book, accurately reflects the different probabilities of prices changes. In view of the queue sizes distributions, it verifies whether the probability mass on the negative and positive side in Figure 5a – but conditioned on the queue sizes – are correct, as these correspond to the particular price changes. The result indicates that conditioning on history, i.e. at least on the previous state X_t, is necessary and learned well by the generator.

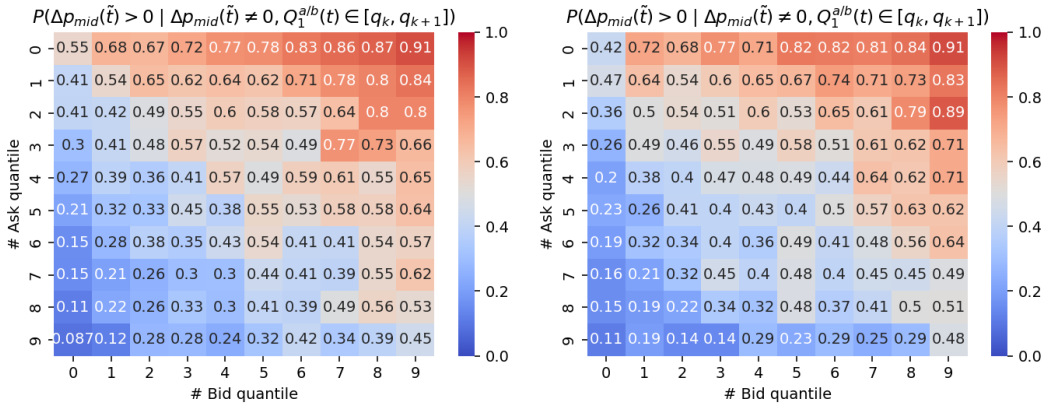


Figure 8: Matrix indicating the probability of a positive price change, given there is a price change, as a function of the quantiles of the best bid/best ask queues. Left: Real, right: Fake.

5.4 Symmetry of simulations

As previously mentioned, training stability is a notorious problem GANs suffer from (Arjovsky et al., 2017; Gulrajani et al., 2017; Mescheder et al., 2017), which has also been documented in Section 5.2. For the present representation, the symmetry of the simulations is important for several reasons. First, the data distributions are close to symmetric, as it can be seen in Section 5.3. Second, on shorter time scales, it is economically sensible the the mechanics in the LOB are generally symmetrical. Lastly, imbalances between opposite generated distributions (e.g. Ask-1 and Bid-1) may introduce drifts.

In particular, it is problematic if

$$\mathbb{P}_g(Q_{p_b}(t + \Delta t) > 0) \neq \mathbb{P}_g(Q_{p_a}(t + \Delta t) < 0),$$

and more generally if

$$\mathbb{P}_g(Q_{p_b - i\delta}(t + \Delta t) > 0) \neq \mathbb{P}_g(Q_{p_a + i\delta}(t + \Delta t) < 0).$$

In other words, price increases and price decreases of certain ticks are not equally likely (in an unconditional sense). The consequences are drifts which potentially offer arbitrage opportunities. To this end, we may make the following assumption.

Assumption 5.1. For any state S_t, the probability distribution should be symmetrical, in the sense that

$$\mathbb{P}_g(Q_{p+i\delta}(t+\Delta t)|S_t) = \mathbb{P}_g(r(Q_{p+(2k-1-i)}(t+\Delta t))|r(S_t)), \tag{22}$$

where $r : \mathbb{R} \to \mathbb{R}$ denotes the point reflection around the mid-price.

This mainly states that state dynamics in the LOB are assumed to be symmetric, e.g. $r(Q_{p_b}(t)) = -Q_{p_a}(t)$. Hence, for each step in the simulation, the state S_t is flipped with probability 0.5.

$$X_{t+\delta t} = \begin{cases} g(Z_t, S_t), & \text{with } p = 0.5 \\ r(g(Z_t, r(S_t))), & \text{with } p = 0.5 \end{cases} \tag{23}$$

By design, the generated probability distribution is then symmetrical around the mid-price, which ensures that there is no structural drift in the simulations that could be exploited by an agent.

5.5 Benchmarks: Poisson and Hawkes order flow

The results in Section 5.3 indicate good fits of distributional quantities of the queue sizes, as well as (conditional) price paths. A natural question to consider is how these results compare to traditional event stream based methods, such as Poisson or Hawkes processes driven order flow.

To this end, we emphasize that our model generates snapshots of limit order books, and not the entire event stream. Consequently, some granularity is lost in the generative process. However, as mentioned in the introduction, this loss is not of critical importance for many market participants that operate outside of the high-frequency setting, but rather focus on mid-frequency time scales.

Both Poisson and Hawkes models are briefly explained in the appendix. We refer the reader to (Abergel, Anane, Chakraborti, Jedidi, & Toke, 2016) for a detailed explanation of the models and their simulation, in particular chapters 6 and 8 therein. Details on the simulation are given in chapter 9.

The figures in Appendix A show marginals, average order book shape, as well as price paths and conditional price changes of the snapshots taken from the Poisson order flow. Figure 13a reveals that the queue size distribution of the generated snapshots differ substantially from that of the real distribution shown in blue. Furthermore, the indicated Wasserstein distances are one order of magnitude higher as they are for the proposed GAN-based model. Most importantly, it can be seen that the probability mass for price changes (i.e. a negative Ask 1 or positive Bid 1 queue respectively) is much smaller in the generated data. This indicates a lower frequency of price changes compared to the real data. Consequently, the average order book shape in Figure 13b also differs substantially in comparison to the GAN-based model. The same figures for the Hawkes-driven order flow are shown in Appendix B.

Figure 14 shows the price paths of the same number of simulations as in Figure 7. A closer look reveals that the generated real paths exhibit a significantly lower volatility, about half of the volatility seen in real price paths and those generated by the suggested GAN model. This is a general issue with Poisson order flow, which has a mean-reverting queue because the cancellation intensity decreases with queue sizes, which is not necessarily realistic.

Finally, Figure 15 shows the equivalent matrix on the right hand side as the right one in Figure 8 but for Poisson order flow. The pattern is somewhat similar to what can be observed in the real data, on the left side. However, the fit is much worse than for the GAN model. Furthermore, the indicated probability is condition that a price change occurs. As mentioned above, the frequency of price changes itself is much lower for the benchmark which is not reflected in this matrix.

6 Interaction with the simulator

Throughout the last sections, we presented an approach to use generative adversarial networks to simulate snapshots of LOB for a specified time interval Δt. The results in Section 5 demonstrate that the proposed

model reproduces unconditional stylized facts very accurately, including price time series.

When it comes to synthetic LOB data/simulators, the ultimate eventual target is to obtain a realistic response of the simulator when the user interacts with it. Most importantly, the so-called *market impact* of the interaction should be realistic. One important study has been conducted by (Gatheral, 2010), which in particular deals the "square-root law". The market impact, as a function of the quantity normalized by trade volume, takes the shape of a square-root function, defined as

$$\text{Cost} = \text{Spread term} + c\sigma\sqrt{\frac{n}{V}}. \tag{24}$$

This interaction generally takes place when either submitting new limit orders, canceling existing limit orders, or executing against existing limit orders via a market order. In this section, we analyze the effect of the

1. execution of a larger parent order of different sizes via market orders
2. liquidity provision via submission of limit orders
3. execution of parent orders via market orders with different slicing.

6.1 Market order execution

In this section, the execution of a parent order via several market orders is considered. In particular, we assume

- a parent order size $Q^{parent} \in \mathbb{Z}$
- an execution horizon of $n \in \mathbb{N}$ time steps LOB transitions (i.e. $n\Delta t$ seconds)
- some frequency f.

Q^{parent} is the total order size to be executed over n time steps and frequency f. If $f = 1$, a market order is sent every time step (Δt), and the amount of liquidity $\frac{Q^{parent}}{n}$ is removed from the book. A market order can be expressed as a 3-tuple (t, p, q), with $q > 0$ and $p = 0$ for sell market orders, and $q < 0$ and $p = \infty$ for buy market orders respectively.

Example 6.1 (Market order execution)**.** Assume some current order book state X_t with ask side $Q_{p+i\cdot\delta}(t) < 0$, $\forall\ i \in k, \dots, 2k-1$, as described in Definition 3.4. A buy market order (t, p, q), with $p = \infty$ and quantity $q < 0$, is then executed instantaneously at time t, which leads to an instantaneous change in the LOB snapshot. For instance, if the market order does not exceed the queue, i.e. $|q| < |Q_{p+(k-1)\cdot\delta}(t)|$, the altered best queue size reads

$$\tilde{Q}_{p+k\cdot\delta}(t) = Q_{p+k\cdot\delta}(t) + q.$$

In case the order is larger than the available liquidity at the best price, the queue gets depleted and the remainder of the order gets executed against available liquidity at the second best level, a phenomenon commonly referred to as multi-level sweep event.

Then, the general altered queue on any bid level reads as

$$\tilde{Q}_{p_b - i\delta}(t) = Q_{p_b - i\delta}(t) + \left(q + \sum_{j=0}^{i-1} Q_{p_b - j\delta}(t) \right)_+, \tag{25}$$

where $p_b = p + (k-1)\delta$ is the best bid price as outlined in Definition 3.4. The second term is the remaining quantity from the parent order which has not been executed by depleting the previous $i-1$ bid levels. The difference is placed into $(\dots)_+$ brackets, as a negative quantity means that the child order was smaller than the cumulative quantity of the previous levels.

Vice versa, the altered queue level on any ask side reads as

$$\tilde{Q}_{p_a + i\delta}(t) = Q_{p_a + i\delta}(t) + \left(q + \sum_{j=0}^{i-1} Q_{p_a + j\delta}(t) \right)_-. \tag{26}$$

Also here, the second term is the remaining quantity of the order after depleting the previous $i-1$ ask levels. This time, the $(\dots)_-$ brackets are in the equation as only a negative remainder indicates a partially executed order.

Figure 9 shows the result of 10000 executions with increasing Q^{parent}. The parent orders are multiples of quantiles of the distribution of queue sizes. The number of time steps n is set to 60, and $f = \frac{1}{3}$. Each child order is hence of size $\frac{Q^{parent}}{n}\frac{1}{f}$. The mean price paths are shown in Figure 9. To improve comparability, the paths are created with the same initial order book states S_0 and also the noise $Z_0, \dots, Z_T$ is identical. This aims to provide a more realistic counterfactual scenario.

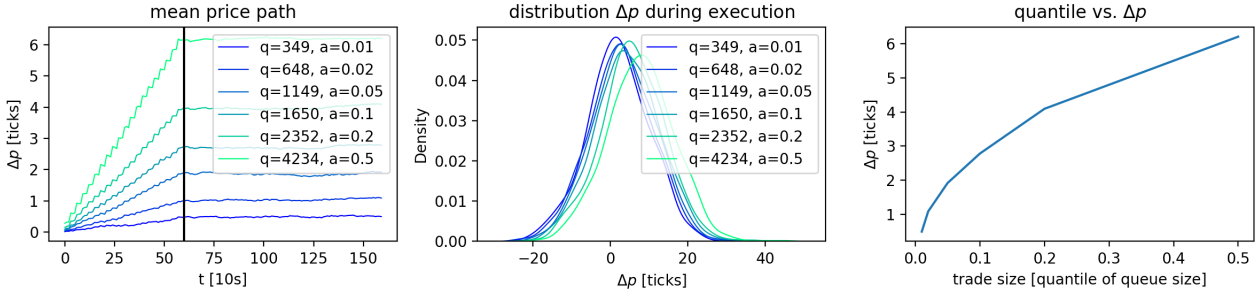


Figure 9: Execution of a parent order with different sizes based on the empirical distribution of the queue size.

- The leftmost plot in Figure 9 shows the average price paths for increasing buy parent orders with increasing sizes. The execution ends after $60\Delta t$ and a total of $n \cdot f$ market orders. It can clearly be seen that the model entails an amount of market impact that aligns well with several expectations.

  1. Buy orders have a positive price impact, i.e. the mean price path of the execution is positive.
  2. The average price impact increases with increasing parent order (child order) size.
  3. The impact stops after the execution, which occurs because of two reasons. First, the extra drift induced by sequentially taking liquidity from one side stops. Second, due to the Markovian structure of the model presented here, a persistent market impact would reveal a lacking stability of the simulator.

- The middle plot in Figure 9 indicates the distributions of the prices at the end of the execution. The shifts of the distribution are clearly visible.

- The right plot in Figure 9 displays the average price impact of the order executions depending on the order size

$$\mathbb{E}_g[\Delta p_{mid} | Q^{target}]. \tag{27}$$

  As already mentioned above, price impact increases with increasing queue size. Furthermore, the curve on the right plot in Figure 9 appears to be concave, which gives the price impact curve a square-root like shape, as often found and argued in the literature, especially in the seminal work of (Gatheral, 2010). The model inherently reproduces this pattern without being specifically trained to do so.

Figure 10 shows 200 of the 10000 execution paths. While the effect of the execution is not strongly visible for the smallest child order size, the effect quickly becomes much stronger. In particular, the two largest order sizes avoid stronger downward moves from the paths. After the end of the execution (indicated by the red vertical line), the trend disappears and the paths continue without any noticeable drift, as one would expect.

For comparison, Figure 16 shows the equivalent experiment with Poisson-based event simulation. Overall, an impact can be seen which tends to be increasing with increasing child order size. However, this impact is very small for the smaller child order sizes and not too conclusive. Only for the largest child order size we observe an impact similar to the one illustrated in Figure 9. This behavior is, however, more of a mechanical nature, because depletion of price level is quite likely in that setting. The price impact curve on the right plot has a more convex instead of a concave pattern, which stands in contrast to observations from the market impact literature (Gatheral, 2010).

Figure 21 displays the impact patterns with Hawkes-based event simulation. The simulation is performed as described in (Abergel et al., 2016). Similar as for our model and the Poisson order flow, an impact is observable

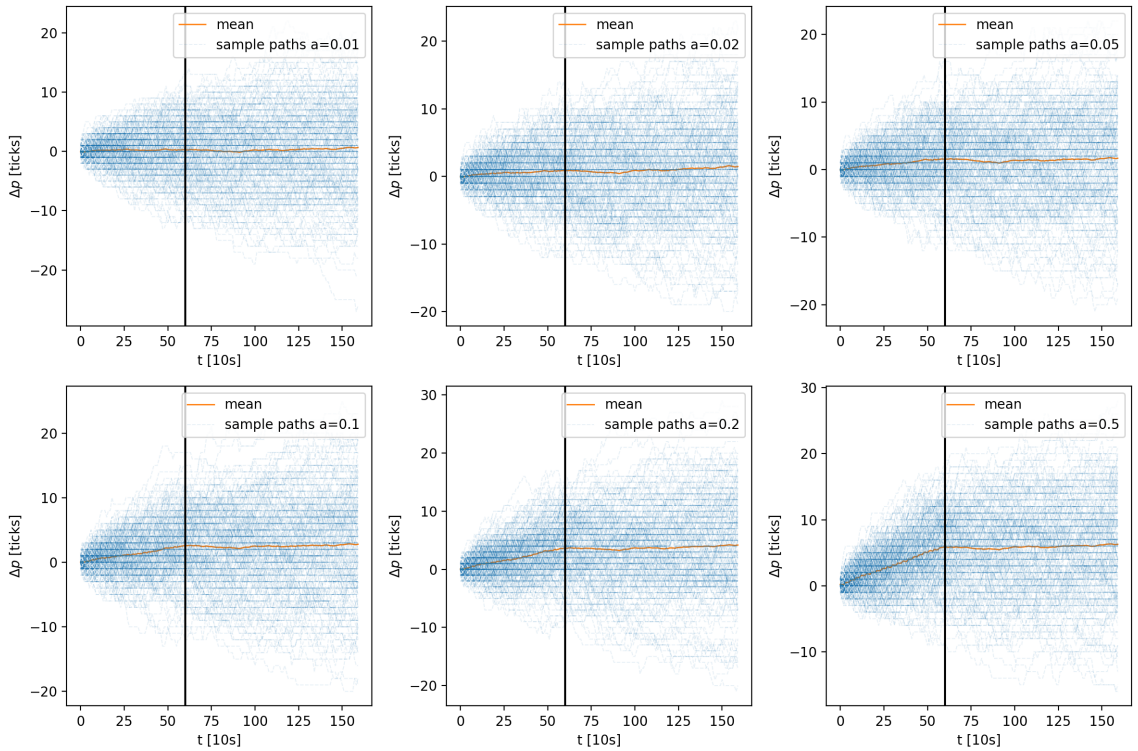


Figure 10: Price paths during buy market order execution. The orange horizontal line indicates the end of execution.

with increasing child order size. Moreover, the impact for smaller child order sizes appears more realistic than using Poisson arrival times. However, no decreasing marginal impact can be observed, and the impact of smaller trade sizes becomes less clear in comparison to the results in Figure 9. Lastly, as for Poisson order flow, it can be seen that the distribution of terminal prices with execution also shows much less volatility than the real data, in comparison to our proposed GAN model.

6.2 Liquidity provision

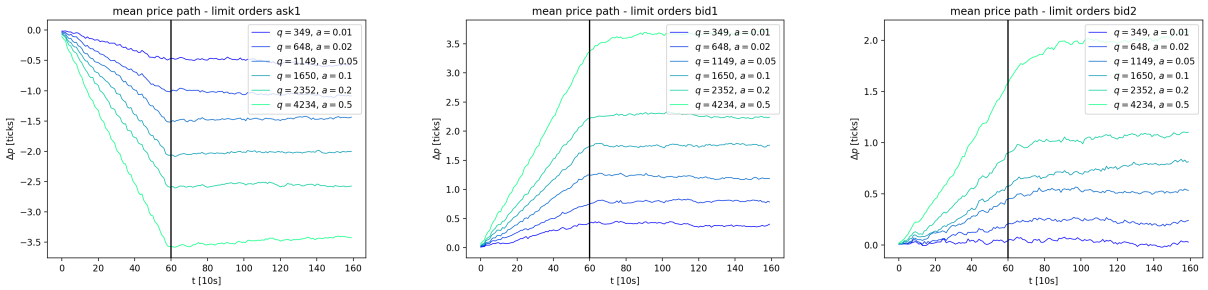
The previous section provided insights into how the model reacts when interacting with it by "taking liquidity" via instantaneously changing the order book state X_t. Market participants also provide liquidity by sending limit orders. Similar to liquidity extraction, the submission of limit orders has an impact. As before, we consider an order submission schedule where a quantity Q^{parent} is submitted over n time steps ($n \cdot \Delta t$ seconds) using different child orders. This results in $n \cdot f$ limit order submissions every $\frac{1}{f}$ time steps. The altered queue size reads

$$\tilde{Q}_p(t) = Q_p(t) + q, \tag{28}$$

and $q = Q^{parent} \cdot \frac{1}{n \cdot f}$. Note that for limit orders, the queue and the order quantity q have the same sign. Only market orders have a different sign. Hence, a queue never gets depleted here and one submission only affects one queue.

Figure 11 shows the mean price paths of 10,000 limit order submissions. For comparability with the experiments from Section 6.1, the order sizes are kept the same with $n = 60$ and $f = \frac{1}{3}$, leading to 20 limit order submissions. Sell order submissions at the best ask price $p_a(t)$, i.e. queue $Q_{p+k\cdot\delta}(t)$ are shown in the leftmost plot, and buy order submission at the best bid $p_b(t)$, i.e. queue $Q_{p+(k-1)\cdot\delta}(t)$ in the middle plot. The rightmost plot shows the effect of providing liquidity at the second best bid price $p_b(t) - \delta$.

Once induced in the order book, we assume the order might be cancelled or executed according to the general dynamics learned by the model. As before, we keep on simulating for 100 transitions after the execution. Furthermore, we use the same state initialization S_0 and noise sequence $Z_0, \dots, Z_T$ as for the market order execution in Section 6.1, in order to preserve the counterfactual quality of the scenarios.



(a) First Ask (b) First Bid (c) Second Bid

Figure 11: Market impact of liquidity provision on both ask and bid side of the LOB. The black vertical line indicates the stop of the trade.

1. Similar to simulated market orders via liquidity extraction, the sequential liquidity provision also exhibits price impact. This price impact has the expected direction; buy limit orders at the bid drive markets up (or avoid price decreases) and vice versa. Limit orders support a particular price level for two main reasons. First, from a technical point of view, more quantity must be executed in order to deplete the particular queue. Second, it indicates demand for a particular price reinforcing the belief that the price of the corresponding asset should be higher. This makes a price move towards the direction of the limit order less likely.

2. The observed price impact for the same quantity placed into the book is lower than when extracting liquidity. E.g. for child order size of $q = 4234, Q^{parent} = 84,680$ shares, the price impact for limit order submission is around 3.5 ticks in comparison to roughly 6 ticks for the market order execution of the same size and execution style.

3. The mean price path seems smoother than the mean price path under liquidity extraction in Figure 9. One reason for this is that no price changes are directly caused by liquidity provision, in contrast to the extraction of liquidity. Liquidity provision instead has more the effect of a price support avoiding the depletion of the queue.

4. The magnitude of the impact when providing liquidity is the same for both the bid and ask side. This hold both for the symmetric sampling as well as normal sampling.

5. Similar as before, the impact stops after execution (at time step 60) and the mean price path continues with zero drift. This is expected, in particular considering the Markovian structure of the generator.

6. Lastly, Figure 11c shows the effect of placing limit orders at the second best bid level $p_b(t)+\delta$ $(p+(k-2)\cdot\delta)$. Interestingly, the model also learns price impact of placing liquidity deeper in the book. The price impact of the same order stream is yet smaller than placing limit orders at the best level – in line with our expectations. Figure 11c shows a slight continuation for a few steps after the execution. This may be interpreted as a "wall of liquidity", built up during the provision for large child order size, which then has more persistent impact.

Figure 17 shows the equivalent results using Poisson order flow. The picture here is very unclear. The addition of limit orders appears to have a rather diffuse impact, which is not quite clear in comparison to the GAN model. Again, the largest order size on the first Ask level tends to have some negative impact. This impact however is very small, and smaller child order sizes do not seem to have any impact at all in comparison. The dynamics when placing at the first bid tend to be the same for all order sizes in this simulation. Limit orders at the second Bid let assume a very slight impact. It stands to question whether this is significant or not. Overall, the picture is very noisy and not as clearly structured as Figure 11.

Similarly, for the Hawkes process driven order flow, the patterns of liquidity provision are fairly unclear, as shown in Figure 22. Similarly to the Poisson order flow, a qualitatively recognizable drift can be seen, in particular in the simulations on the bid side. However, there is no real distinguishable effect between the order sizes in comparison. The overall effect however is much smaller than the one from liquidity provision. Also, the impact does not necessarily appear to be monotone here. We also emphasize that the simulation takes significantly more time than for our proposed model, in line with findings from the Hawkes processes literature.

6.3 POV execution

The previous two subsections provide evidence that the proposed simulator automatically learns to exhibit some form of market impact, which is in line with economic expectations, findings in the literature, and expectations of industry practitioners. This holds for both liquidity extraction via market orders as well as liquidity provision via limit order submissions.

This section analyzes how the model behaves when the same quantity Q^{parent} is executed over different frequencies and lengths. In particular, executing with different frequencies implies that, over the execution horizon, the percentage of volume (POV) is the same despite the simulator not tracking executed volume. It thus stands to question what difference it has on how Q^{parent} is sliced into larger and smaller child orders, which are then sent during n time steps and at different frequencies.

In particular, we assume a quantity Q^{parent} over $n = 100$ time steps and then execute Q^{parent} via market orders, as outlined in Section 6.1 in two ways

1. Execution with even rate, i.e. $f_{even} = 1$: This strategy executes $q = \frac{Q^{parent}}{n}$ at every Δt. The mean price path of the execution is shown in orange in Figure 12. The execution schedule is depicted on the right side in Figure 12.

2. Slow (quantized step) execution with $f_{quantized}$: Executes larger chunks $q = \frac{Q^{parent}}{n \cdot f}$ every $f_{quantized}^{-1}$ time steps. Mean price path and execution schedule are shown in blue, in the same Figure 12.

Figure 12 shows the results for both execution styles and $Q^{parent} = 10,000$. The quantized frequency $f_{quantized}$ is set to 1/10, hence, we execute 10 times as much as during the even execution but only after every 10th transition. Both execution styles lead to roughly the same price impact, suggesting that POV is the main driver in this model – despite not explicitly modeled and incorporated into the learning process. This is in line with studies such as (Almgren & Chriss, 2001), stating that POV is the primary driver of market impact. The price paths of the quantized execution lead to more price changes during the placement of larger market orders, both by inducing price changes directly or causing large imbalances, making price changes very likely to occur.

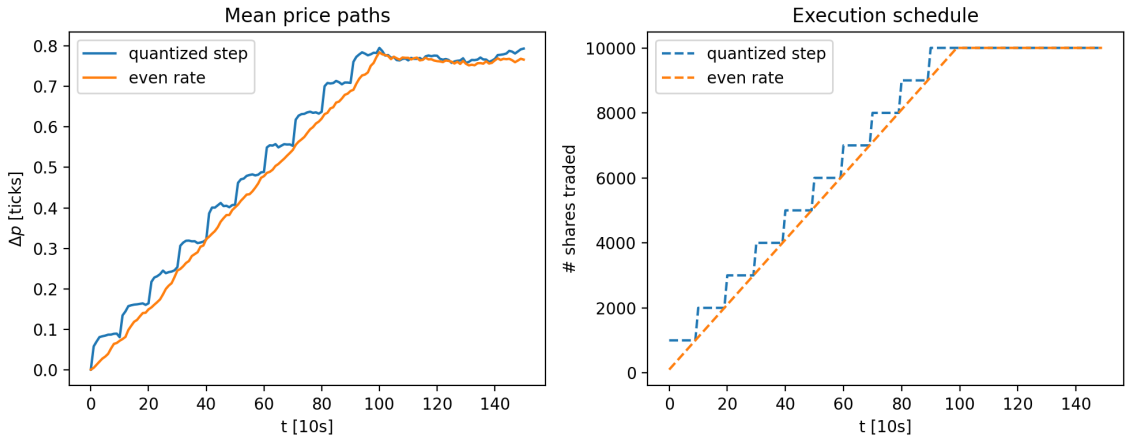


Figure 12: Execution of the same quantity Q^{parent} with a larger and smaller parent orders, leading to the same POV both over the entire time interval.

We remark that by increasing Q^{parent}, decreasing n or decreasing $f_{quantized}$, child orders may become very large – in particular in the quantized execution setting. For these child orders, which consistently deplete more than the queue, the model appears to have difficulties to absorb these larger orders. The consequence is that quantized execution tends to have smaller market impact. Potentially, the model would learn here more when increasing the training data and inducing more samples with several empty levels. However, as the book is mostly populated, frequent acting which leads to level depletion is rather unrealistic.

7 Conclusion

Ever since limit order books have been used first to organize electronic exchanges, a vast amount of studies have been conducted. In particular, the construction of realistic models for generating synthetic data is an ongoing challenge, and has been attempted to solve with a variety of methods, ranging from stochastic point processes to agent-based models. For many applications, it is sufficient to learn the dynamics of the public LOB view – the LOB snapshots for a certain price grid around the best price – rather than the entire order-by-order event stream. This in particular holds true for mid to high-frequency trading because many investors base their decisions on the time series of LOB snapshots.

In this work, we consider the modeling of time series of these LOB snapshots. We seek a generative model that learns the probability distribution of future order book states, conditioned on previous LOB snapshots, aiming to

1. reproduce certain stylized facts, i.e., match distributions, correlations, auto-correlation properties etc.
2. reproduce market impact when interacting with the model.

The proposed model is based on GANs (Goodfellow et al., 2014) that learn the transitions between LOB snapshots for some time step Δt. The current Markovian system only uses the snapshot at time t in order to learn $\mathbb{P}_r(x_{t+\Delta t}|s_t)$. The framework is proposed in a way that enables the GAN to generate discrete price changes via the process of queue sizes, thereby circumventing the weakness of GANs to learn discrete probability distributions. Moreover, generating LOB snapshots by sampling from a conditional probability distribution directly via GANs is much faster and efficient than event-based modeling and simulation.

The obtained results show that GANs are able to learn a wide range of dynamics of LOBs in a simple Markovian setting. As analyzed in Section 5, most statistical properties such as marginal distributions and correlations are learned very fast. Also, the drift and volatility of the price paths quickly match the properties of the real data. By using gradient penalty and clipping of the critic's gradient in the Wasserstein GAN setting, training can be further stabilized, as otherwise the generated distribution tends to fluctuate. This accuracy is indispensable, as slight inaccuracies can lead to drifts in the price paths and other issues. To this end, we adjust the sampling method to make the model's samples symmetric.

Apart from matching statistical properties when using the model to generate synthetic data, the GAN may also be used for actual interaction of the user with the simulator. More importantly, the conditional probability distribution is altered by changing the state which corresponds to simulating trades. Among general patterns, we demonstrate that the model

1. shows market impact which is expected from economic intuition,
2. recovers the square-root law with respect to trade size, as commonly known in the literature,
3. reproduces a smaller impact of liquidity provision compared to liquidity extraction,
4. exhibits even less impact of liquidity provision deeper in the book.

We compare the results to two order flow based models, namely Poisson and Hawkes order flow. The latter is often considered as the benchmark in the field of order book modeling. As mentioned in the introduction, many applications require the modeling of the state of the book, and not of the entire stream. That being enough, our model outperforms in most metrics. Despite losing certain granularity, we show that the quantity process of the queues is modeled much more accurately, which inherently leads to more realistic price paths in comparison to Poisson and Hawkes processes which exhibit much less volatility compared to the real data. While the market impact due to liquidity extraction shows patterns not too different, our model produces a more concave market impact curve. A striking difference is observed in liquidity provision, which does not show a clear picture for the benchmark models, whereas our proposed approach shows a clear impact pattern.

Future research directions. As anticipated in the introduction, this work opens up plenty directions for future research, some of which are indispensable to be researched in order to use the approach as a realistic backtesting environment and other use cases.

GANs are known for their instability during training. In particular, in the later training stages, when the GAN is close to the solution, the GAN tends to fluctuate around the target. These fluctuations can be enough to cause slight imbalances in the generated distributions, in particular the marginals of the best levels in Figure 5a. The end result is that the probability for price increase and decrease are not balanced. This can result in drifts of the generated price paths. It would thus be of great use to further improve the convergence of the model and/or enforce symmetry within the model. Potentially, normalizing flows (Papamakarios, Nalisnick, Rezende, Mohamed, & Lakshminarayanan, 2021) may be used to improve this stability.

The presented results were obtained with a Markovian setting, as outlined in Section 4. However, there is path dependency well documented in previous studies, e.g. (J. Sirignano & Cont, 2019). In particular, the impact of trade sequences may be different, as one would expect a decaying market impact when accounting for history. Whether or not the inclusion of more history leads to improved and/or different results remains to be investigated.

This study models the state X_t and how it evolves over time. It does not contain any information what happens during the transition from t to $t + \Delta t$. In future work, the state may be extended/enlarged such that more information could be modeled, for e.g as the quantity of order submissions, cancellations and executions, in order to obtain insights on whether a submitted order has been executed or not. This could then better inform the fill probability modeling, which is of interest to both practitioners and academics.

We mainly consider dense LOBs where $Q_p(t) \neq 0$. This means that there are no empty levels, and in particular, the spread $s(t) = p_a(t) - p_b(t) = \delta$. Currently, the network is designed with a linear activation function in the final output layer. Assuming the stock modeled contains many snapshots with $s(t) > \delta$, the distribution of the queue sizes will be zero-inflated. The GAN would have to learn to map many values directly (or sufficiently close) to zero which is inconvenient using a linear activation function in the final layer. It would hence be beneficial to adjust the design such that the model supports zero-inflated distribution. This could be achieved by either changing the output structure, or using an activation function which supports and promotes the generation of zeros.

Lastly, we have disregarded stationarity issues in our study. In particular, we uses 20 subsequent days of a stock regardless of its trading volume, time of the day, volatility, etc. To this end, one could use a volume-based or tick-based clock, as an alternative to the calendar clock we used throughout this paper. Furthermore, increasing the size of data and adding further conditions to account for, e.g. intraday patterns, may improve the data generation process.

References

Abergel, F., Anane, M., Chakraborti, A., Jedidi, A., & Toke, I. M. (2016). *Limit order books*. Cambridge University Press.

Abergel, F., & Jedidi, A. (2013). A mathematical approach to order book modeling. *International Journal of Theoretical and Applied Finance*, *16*(05), 1350025.

Abergel, F., & Jedidi, A. (2015). Long-time behavior of a hawkes process–based limit order book. *SIAM Journal on Financial Mathematics*, *6*(1), 1026–1043.

Almgren, R., & Chriss, N. (2001). Optimal execution of portfolio transactions. *Journal of Risk*, *3*, 5–40.

Arjovsky, M., Chintala, S., & Bottou, L. (2017). Wasserstein generative adversarial networks. In *International conference on machine learning* (pp. 214–223).

Bouchaud, J.-P., Mézard, M., Potters, M., et al. (2002). Statistical properties of stock order books: empirical results and models. *Quantitative finance*, *2*(4), 251–256.

Byrd, D., Hybinette, M., & Balch, T. H. (2019). Abides: Towards high-fidelity market simulation for ai research. *arXiv preprint arXiv:1904.12066*.

Coletta, A., Moulin, A., Vyetrenko, S., & Balch, T. (2022). Learning to simulate realistic limit order book markets from data as a world agent. In *Proceedings of the third acm international conference on ai in finance* (pp. 428–436).

Cont, R. (2001). Empirical properties of asset returns: stylized facts and statistical issues. *Quantitative finance*, *1*(2), 223.

Cont, R. (2011). Statistical modeling of high-frequency financial data. *IEEE Signal Processing Magazine*, *28*(5), 16–25.

Cont, R., Cucuringu, M., Glukhov, V., & Prenzel, F. (2023). Analysis and modeling of client order flow in limit order markets. *Quantitative Finance*, 1–19.

Cont, R., & Müller, M. S. (2021). A stochastic partial differential equation model for limit order book dynamics. *SIAM Journal on Financial Mathematics*, *12*(2), 744–787.

Cont, R., Stoikov, S., & Talreja, R. (2010). A stochastic model for order book dynamics. *Operations research*, *58*(3), 549–563.

Da Silva, B., & Shi, S. S. (2019). Style transfer with time series: Generating synthetic financial data. *arXiv preprint arXiv:1906.03232*.

Gatheral, J. (2010). No-dynamic-arbitrage and market impact. *Quantitative finance*, *10*(7), 749–759.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., ... Bengio, Y. (2014). Generative adversarial nets. In *Advances in neural information processing systems* (pp. 2672–2680).

Gould, M. D., Porter, M. A., Williams, S., McDonald, M., Fenn, D. J., & Howison, S. D. (2013). Limit order books. *Quantitative Finance*, *13*(11), 1709–1742.

Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., & Courville, A. C. (2017). Improved training of wasserstein gans. In *Advances in neural information processing systems* (pp. 5767–5777).

Hawkes, A. G. (1971). Spectra of some self-exciting and mutually exciting point processes. *Biometrika*, *58*(1), 83–90.

Huang, W., Lehalle, C.-A., & Rosenbaum, M. (2015). Simulating and analyzing order book data: The queue-reactive model. *Journal of the American Statistical Association*, *110*(509), 107–122.

Kirilenko, A., Kyle, A. S., Samadi, M., & Tuzun, T. (2017). The flash crash: High-frequency trading in an electronic market. *The Journal of Finance*, *72*(3), 967–998.

Li, C.-L., Chang, W.-C., Cheng, Y., Yang, Y., & Póczos, B. (2017). MMD GAN: Towards deeper understanding of moment matching network. In *Advances in neural information processing systems* (pp. 2203–2213).

Li, J., Wang, X., Lin, Y., Sinha, A., & Wellman, M. (2020). Generating realistic stock market order streams. In *Proceedings of the aaai conference on artificial intelligence* (Vol. 34, pp. 727–734).

Marti, G. (2020). Corrgan: Sampling realistic financial correlation matrices using generative adversarial networks. In *Icassp 2020-2020 ieee international conference on acoustics, speech and signal processing (icassp)* (pp. 8459–8463).

Mescheder, L., Geiger, A., & Nowozin, S. (2018). Which training methods for GANs do actually converge? In *International conference on machine learning* (pp. 3481–3490).

Mescheder, L., Nowozin, S., & Geiger, A. (2017). The numerics of GANs. In *Advances in neural information processing systems* (pp. 1825–1835).

Mirza, M., & Osindero, S. (2014). Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*.

Morariu-Patrichi, M., & Pakkanen, M. S. (2022). State-dependent hawkes processes and their application to limit order book modelling. *Quantitative Finance*, *22*(3), 563–583.

Ni, H., Szpruch, L., Wiese, M., Liao, S., & Xiao, B. (2020). Conditional sig-wasserstein GANs for time series generation. *arXiv preprint arXiv:2006.05421*.

Paddrik, M., Hayes, R., Todd, A., Yang, S., Beling, P., & Scherer, W. (2012). An agent based model of the e-mini s&p 500 applied to flash crash analysis. In *2012 ieee conference on computational intelligence for financial engineering & economics (cifer)* (pp. 1–8).

Papamakarios, G., Nalisnick, E. T., Rezende, D. J., Mohamed, S., & Lakshminarayanan, B. (2021). Normalizing flows for probabilistic modeling and inference. *J. Mach. Learn. Res.*, *22*(57), 1–64.

Potters, M., & Bouchaud, J.-P. (2003). More statistical properties of order books and price impact. *Physica A: Statistical Mechanics and its Applications*, *324*(1-2), 133–140.

Prenzel, F., Cont, R., Cucuringu, M., & Kochems, J. (2022). Dynamic calibration of order flow models with generative adversarial networks. In *Proceedings of the third acm international conference on ai in finance* (pp. 446–453).

Sirignano, J., & Cont, R. (2019). Universal features of price formation in financial markets: perspectives from deep learning. *Quantitative Finance*, *19*(9), 1449–1459.

Sirignano, J. A. (2019). Deep learning for limit order books. *Quantitative Finance*, *19*(4), 549–570.

Smith, E., Farmer, J. D., Gillemot, L. s., Krishnamurthy, S., et al. (2003). Statistical theory of the continuous double auction. *Quantitative finance*, *3*(6), 481–514.

Smith, K. E., & Smith, A. O. (2020). Conditional GAN for timeseries generation. *arXiv preprint arXiv:2006.16477*.

Takahashi, S., Chen, Y., & Tanaka-Ishii, K. (2019). Modeling financial time-series with generative adversarial networks. *Physica A: Statistical Mechanics and its Applications*, *527*, 121261.

Wiese, M., Bai, L., Wood, B., & Buehler, H. (2019). Deep hedging: learning to simulate equity option markets. *Available at SSRN 3470756*.

Wiese, M., Knobloch, R., Korn, R., & Kretschmer, P. (2020). Quant GANs: deep generation of financial time series. *Quantitative Finance*, *20*(9), 1419–1440.

Zhang, Z., Zohren, S., & Roberts, S. (2019). Deeplob: Deep convolutional neural networks for limit order books. *IEEE Transactions on Signal Processing*, *67*(11), 3001–3012.

A Poisson Order Flow

The object modeled in Poisson order flow as in our model is the queue size. In difference to our approach, we denote

$$Q_i^a = Q_{p_b+i\cdot\delta} \text{ and } Q_i^b = Q_{p_a-i\cdot\delta}. \tag{29}$$

as the ask or bid queue i ticks away from the best opposite price. Three order types then potentially affect a particular queue, namely limit order submissions and cancellations but also market orders affect the best queues according to price time priority. E.g. the oldest order (with the best position in the queue) at the most favorable price will be executed first with an incoming limit order. The different order types arrive with exponentially distributed inter arrival times in the following way:

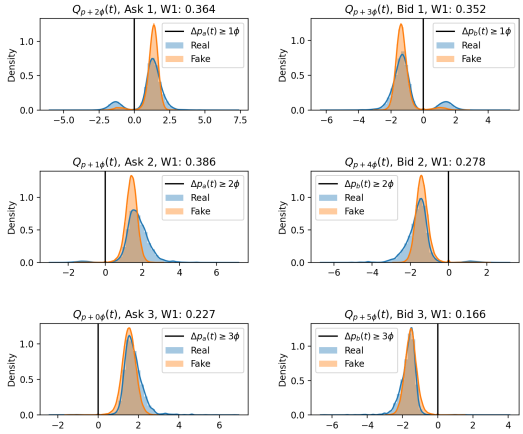
1. Limit order submission at relative price level i arrive with intensity $\lambda_i^{L,a}$ for the i-th level on the ask and $\lambda_i^{L,b}$ on the i-th bid level respectively. Increasing the queue size by $\Delta Q_i^a(t) = q$ where q is the order size.
2. Limit order cancellations at relative price level i arrive with intensity

$$\lambda_i^{C,a/b}(t) = \lambda_i^{C,a/b} \cdot Q_i^{a/b}(t). \tag{30}$$

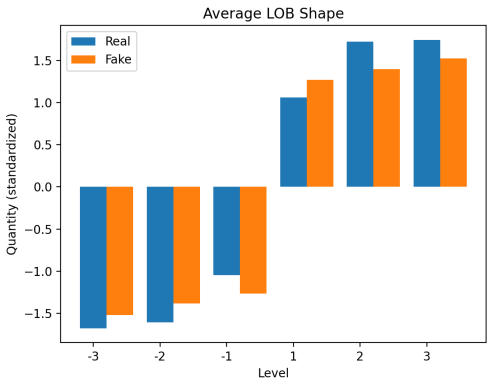
   on ask/bid side respectively. Note the intensity is scaled with current queue size in order to mention ensure non-exploding queues. The change in queue size corresponds to $\Delta Q_p(t) = -q$ where q denotes the size of the cancelled order.
3. Market orders arrive with $\lambda^{M,a/b}$ on the ask/bid side respectively.

Since the arrival times are assumed to be *iid*, the entire process can then be simulated with $Po(\sum_{\lambda\in\Lambda}\lambda)$ where Λ is the set of all intensity rates up to a certain level k. The particular event is then drawn according to the relative size of the intensities.

Order sizes are either set to unit size as in (E. Smith et al., 2003; Cont et al., 2010) or according to some distribution as in (Abergel et al., 2016). The detailed simulation algorithm can be found in (Abergel et al., 2016).



(a) Marginal distributions $X_{t+\Delta t}$.



(b) Average LOB shape.

Figure 13: Marginal distributions and average order book shape for both real and fake data under Poisson order flow.

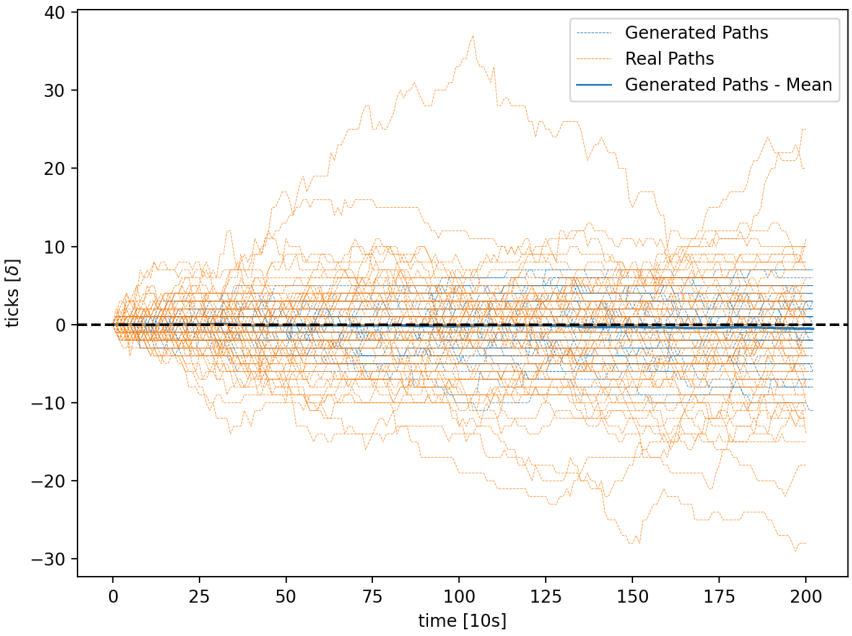


Figure 14: Price paths for real and fake data under Poisson order flow.

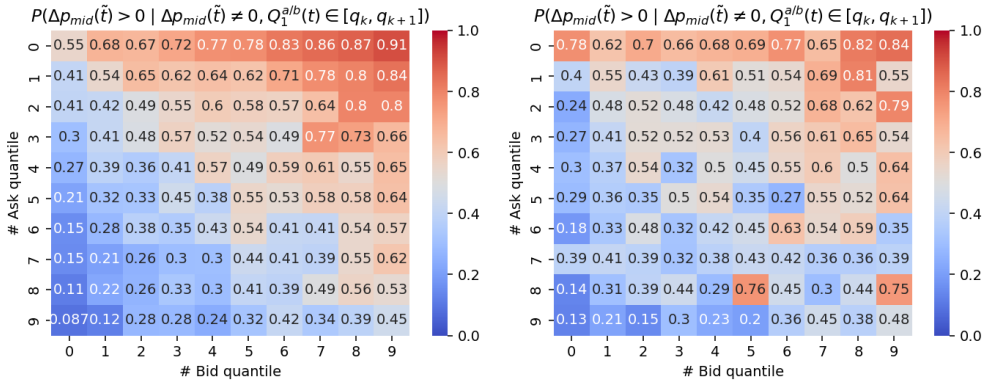


Figure 15: Matrix indicating the probability of a positive price change given there is a price change depending on the quantiles of the best bid/best ask queue. Left: Real, right: Poisson order flow.

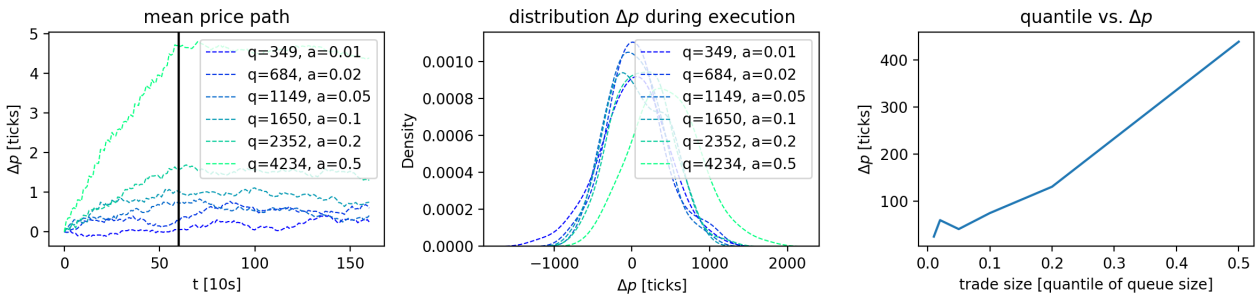
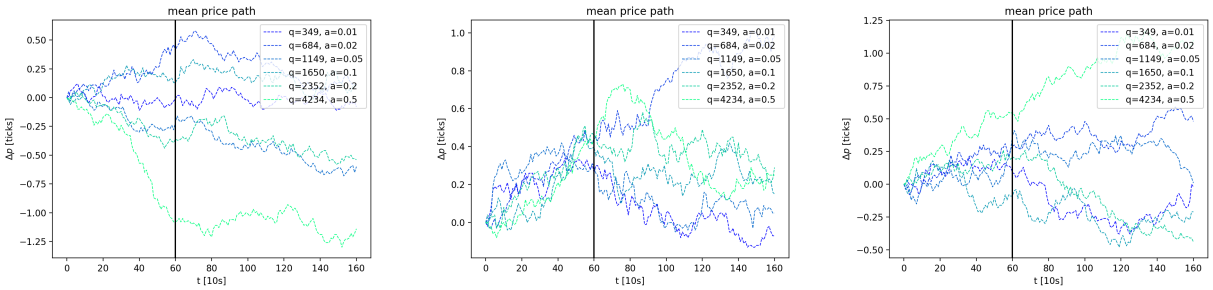


Figure 16: Execution of a parent order with different sizes based on the empirical distribution of the queue size under Poisson order flow. The black vertical line indicates the stop of the trade.

B Hawkes Order Flow

The Hawkes process driven order flow model as in (Abergel et al., 2016) is similar to the Poisson order flow. However, the arrival rates are now given by Hawkes process and are thus time dependent. In general, the arrival



(a) First Ask (b) First Bid (c) Second Bid

Figure 17: Market impact of liquidity provision on both ask and bid side of the LOB under Poisson order flow. The black vertical line indicates the stop of the trade.

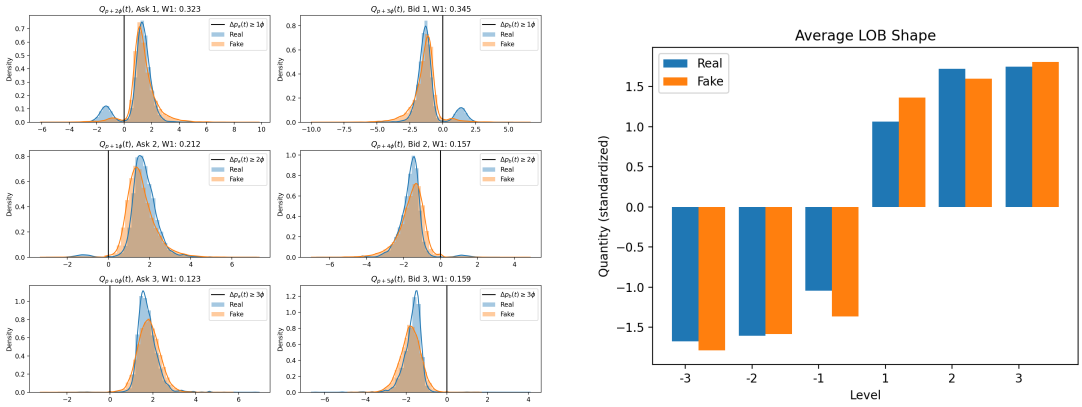
rates of a p-dimensional Hawkes process is given by

$$\lambda^n(t) = \lambda_0^n(t) + \sum_{m=1}^{p} \int_0^t \phi_{nm}(t-s) dN^m(s), n = 1, \dots, p \tag{31}$$

where $\phi_{nm}(t-s)$ is a kernel modeling the effect from a particular event of type m occurred at time s on event type n at time t. For this simulation we use the exponential kernel

$$\alpha_{nm} \exp(-\beta_{nm}(t-s)) \tag{32}$$

which is a common choice together with the power law kernel. The exponential kernel makes the system Markovian and hence faster to simulate. The time dependent intensities are then used via a thinning algorithm so simulate the Hawkes process. Apart from the dynamic intensities and thereby changing inter arrival times, the model is very similar to the Poisson order flow. The detailed simulation algorithm can be found in (Abergel et al., 2016).



(a) Marginal distributions $X_{t+\Delta t}$. (b) Average LOB shape.

Figure 18: Marginal distributions and average order book shape for both real and fake data under Hawkes order flow.

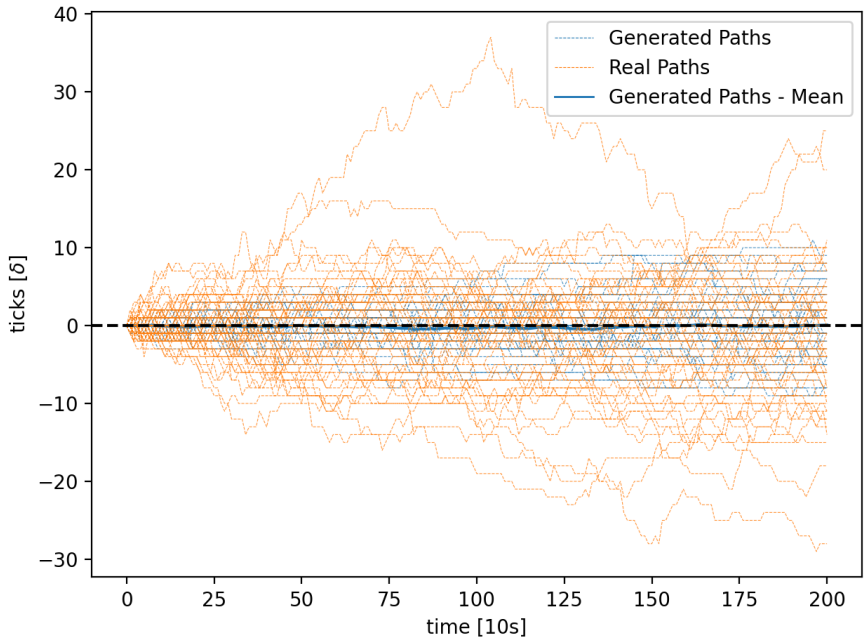


Figure 19: Price paths and corresponding percentiles of terminal prices for real and fake data under Hawkes order flow.

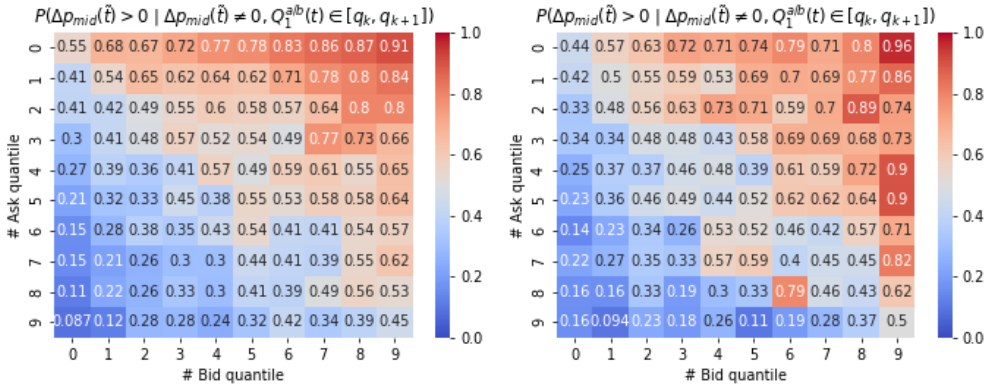


Figure 20: Matrix indicating the probability of a positive price change given there is a price change depending on the quantiles of the best bid/best ask queue. Left: Real, right: Hawkes order flow.

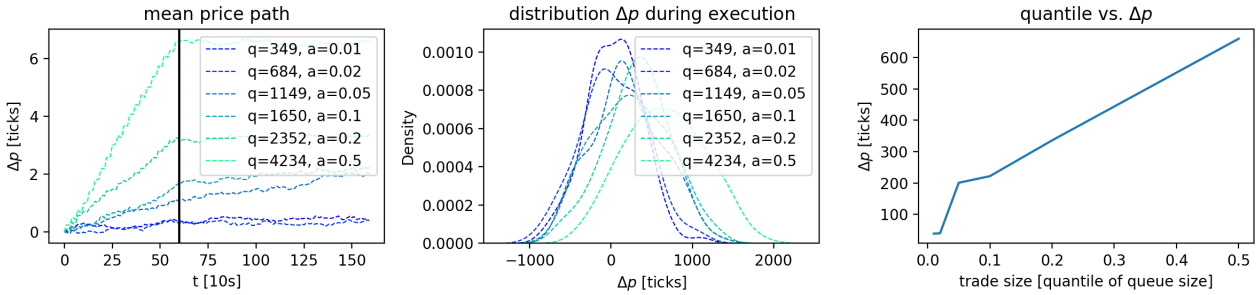
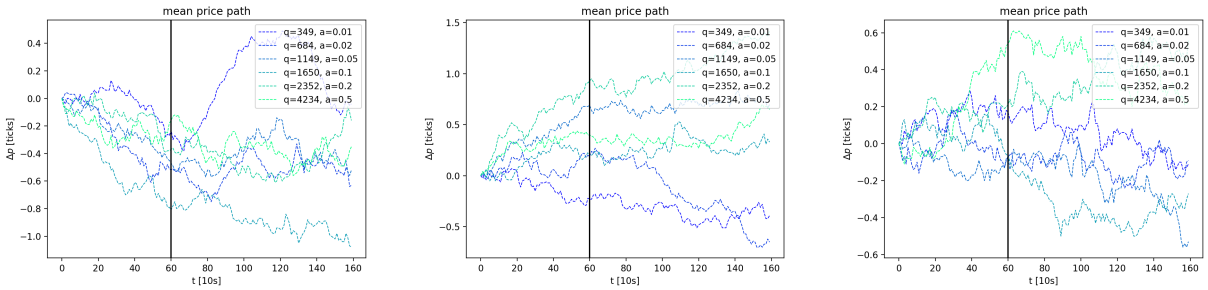


Figure 21: Execution of a parent order with different sizes based on the empirical distribution of the queue size under Hawkes order flow. The black vertical line indicates the stop of the trade.

C Disclaimer

Opinions and estimates constitute our judgment as of the date of this Material, are for informational purposes only and are subject to change without notice. This Material is not the product of J.P. Morgan's Research



(a) First Ask (b) First Bid (c) Second Bid

Figure 22: Market impact of liquidity provision on both ask and bid side of the LOB under Hawkes order flow. The black vertical line indicates the stop of the trade.

Department and therefore, has not been prepared in accordance with legal requirements to promote the independence of research, including but not limited to, the prohibition on the dealing ahead of the dissemination of investment research. This Material is not intended as research, a recommendation, advice, offer or solicitation for the purchase or sale of any financial product or service, or to be used in any way for evaluating the merits of participating in any transaction. It is not a research report and is not intended as such. Past performance is not indicative of future results. Please consult your own advisors regarding legal, tax, accounting or any other aspects including suitability implications for your particular circumstances. J.P. Morgan disclaims any responsibility or liability whatsoever for the quality, accuracy or completeness of the information herein, and for any reliance on, or use of this material in any way.