Quantitative Finance

ISSN: 1469-7688 (Print) 1469-7696 (Online) Journal homepage: www.tandfonline.com/journals/rquf20

A generative model of a limit order book using recurrent neural networks

Hanna Hultin, Henrik Hult, Alexandre Proutiere, Samuel Samama & Ala Tarighati

To cite this article: Hanna Hultin, Henrik Hult, Alexandre Proutiere, Samuel Samama & Ala Tarighati (2023) A generative model of a limit order book using recurrent neural networks, Quantitative Finance, 23:6, 931-958, DOI: 10.1080/14697688.2023.2205583

To link to this article: https://doi.org/10.1080/14697688.2023.2205583

Published online: 24 May 2023.

Quantitative Finance, 2023
Vol. 23, No. 6, 931–958, https://doi.org/10.1080/14697688.2023.2205583

A generative model of a limit order book using recurrent neural networks

HANNA HULTIN †‡*, HENRIK HULT †, ALEXANDRE PROUTIERE§, SAMUEL SAMAMA‡ and ALA TARIGHATI‡

†Department of Mathematics, KTH Royal Institute of Technology, Stockholm, Sweden
‡SEB Group, Stockholm, Sweden
§Department of Intelligent Systems, KTH Royal Institute of Technology, Stockholm, Sweden

(*Received 7 November 2022; accepted 24 March 2023; published online 25 May 2023*)

In this work, a generative model based on recurrent neural networks for the complete dynamics of a limit order book is developed. The model captures the dynamics of the limit order book by decomposing the probability of each transition into a product of conditional probabilities of order type, price level, order size and time delay. Each such conditional probability is modelled by a recurrent neural network. Several evaluation metrics for generative models related to trading execution are introduced. Using these metrics, it is demonstrated that the generative model can be successfully trained to fit both synthetic and real data from the Nasdaq Stockholm exchange.

Keywords: Limit order book; Machine learning; Recurrent neural networks; Generative modelling; High-frequency trading

JEL Classifications: C45, C51, C52

1. Introduction

In high-frequency trading, assets are usually traded in a limit order book on an electronic exchange. The limit order book contains buy and sell orders at different price levels. Agents execute trades by placing market orders that match existing limit orders in the order book, or by posting buy or sell limit orders at the desired price level. Agents may also cancel outstanding orders. As a result of a large number of market agents placing orders, often using automated trading strategies, the dynamic evolution of the limit order book is complex and nonlinear. To understand the complex dynamics of the limit order book, the aim of this paper is to construct a generative model that accurately captures its main features.

There are several reasons for constructing a model for realistic generation of a limit order book, for instance when assessing the performance of automated trading strategies or when designing strategies for risk management or trading execution. Even though simulation models, such as the one proposed in this paper, are generally too slow to use for real-time decisions on risk management or execution strategies, they are useful in the development of new strategies. They allow for controlled experiments in an artificial environment that resembles the real world, where the agent can interact with the model.

When developing a new execution strategy, it is necessary to evaluate the performance of the strategy properly before putting it into production. In high-frequency trading, every transition of the order book may provide information on its future evolution. It is insufficient to rely on historical data for backtesting, since then the market impact of trades cannot be accurately incorporated. In contrast, a realistic simulation model of the limit order book can be used to evaluate the strategy, but modelling the limit order book at a detailed level is a challenging task, addressed in this paper.

The problem of optimal execution, or optimal control of execution costs, has been studied for decades. One way of modelling the execution costs is by defining the dynamics of a market price, and a price impact function specifying the impact a trade has on the market price. This kind of modelling is for example used in Bertsimas and Lo (1998), Almgren and Chriss (2001), Alfonsi *et al.* (2010), Obizhaeva and Wang (2013) and Cartea *et al.* (2015).

The dynamics of limit order books have also been studied using theory of stochastic processes in queueing theory, see, for example Cont *et al.* (2010), Hult and Kiessling (2010), Cont and De Larrard (2013), Huang *et al.* (2015), Huang and Rosenbaum (2017) and Muni Toke

*Corresponding author. Email: hhultin@kth.se

and Yoshida (2017). Typically the dynamics are simplified considerably to construct a tractable model. Common simplifying assumptions include that the limit order book is Markovian, that the order flows at different levels are independent processes and that the order size is constant. These simplified models can still capture important features of the dynamics of the limit order book and improve the understanding of the limit order book, but are often not sufficiently realistic to use in practice. Such approaches are useful for understanding the mechanisms of the markets, but are typically not designed to be realistic enough to use for backtesting and development of strategies for trading and risk management. Modelling the limit order book by standard stochastic processes typically leads to models that are very simplistic compared to the real world, to ensure tractability.

Another approach to modelling limit order books is to specify the behaviour of a group of agents, see Parlour and Seppi (2008) and Byrd *et al.* (2020). These models are also important to increase the understanding of the market, but have other shortcomings. For example, it is difficult to calibrate each agent such that the aggregated system resembles the real world.

Recent developments in machine learning have led to the development of flexible models, such as neural networks and extensions thereof, for complex and high-dimensional dynamics. Recurrent neural networks (RNNs) are a special type of artificial neural networks that are primarily used for regression or classification in time series data and other data with temporal-like structure. The temporal dependence is implemented by feedback connections that represent internal states, often referred to as memory. The long short-term memory network (LSTM) is a popular recurrent network architecture designed to deal with the problem of vanishing and exploding gradients that typically occurs when training vanilla RNNs, see Hochreiter and Schmidhuber (1997) and Gers *et al.* (1999). LSTM networks are shown to handle long-range dependence in sequential data well and are successfully applied to many applications, for example speech recognition (Graves *et al.* 2013), handwriting generation (Graves 2013) and machine translation (Sutskever *et al.* 2014).

Models based on neural networks are currently being successfully implemented in a wide range of financial applications, including hedging (Buehler *et al.* 2019), mortgage risk (Sadhwani *et al.* 2021) and computing implied volatility surfaces (Horvath *et al.* 2021). Furthermore, there are several successful examples of using machine learning models for limit order book data, with particular emphasis on predicting mid-price changes. To the best of our knowledge, there are no earlier works on using neural networks for modelling the full dynamics of the limit order book. Kercheval and Zhang (2015) used support vector machines to forecast movements of the mid-price and price spread crossing. Following this work, there have been multiple efforts of tackling the task of predicting mid-price movements with different architectures of neural networks, including RNNs (Dixon 2018) and convolutional neural networks (CNNs) (Doering *et al.* 2017). Sirignano (2019) used a neural network architecture for modelling the joint distribution of ask and bid prices at a future time.

The progress has been facilitated by the publishing of a publicly available benchmark data set for mid-price forecasting of limit order book data (Ntakaris *et al.* 2018). This data set, known as FI-2010, contains data for five stocks traded on the Nasdaq OMX Nordic at the Helsinki exchange from 1 June 2010 to 14 June 2010. The authors of the data set publication have also been involved in several works where the data set was used for mid-price prediction by applying different methods, such as neural bag-of-features, CNNs and RNNs, see Passalis *et al.* (2017), Tsantekidis *et al.* (2017) and Tsantekidis *et al.* (2017). The data set is also used by other authors, for example Zhang *et al.* (2019) introduce their model DeepLOB and apply it to two different data sets with one being the FI-2010 data set.

Recently, there have been some work on applying generative models to financial applications. In particular, restricted Boltzmann machines (Kondratyev and Schwarz 2019, Kondratyev *et al.* 2020), variational autoencoders (Buehler *et al.* 2020), as well as generative adversarial networks (Wiese *et al.* 2019, 2020) have been used to generate time-series of returns.

In this paper, a generative model is constructed to simulate the dynamic evolution of a limit order book at the most granular level, transition by transition. The model is based on RNNs that output conditional probabilities of transitions given the history and state of the order book. The transitions are characterized by event type (market order buy/sell, limit order buy/sell, cancellation buy/sell), price level, order size and point in time. The joint probability of a transition is decomposed as a product of conditional probabilities, where the probability of event type is conditional on the state and history of the order book, the price level is conditional on the event type, state and history, the order size is conditional on the price level, event type, state and history, etc. The model is trained by maximum likelihood estimation using stochastic gradients and backpropagation through time.

To evaluate the quality of the proposed model, it is first trained and evaluated on a synthetic data set generated from a known continuous time Markov chain model, which is a slight extension of the model proposed by Cont *et al.* (2010) by allowing for random order sizes, and then on real equity transaction data from the Nasdaq exchange in Stockholm. Statistical features of the generated data using the proposed model are compared to those of the training data. The Markov chain model is used as a proof of concept to show that the generative model is able to learn the dynamics of the Markov chain model. The Nasdaq Stockholm data is used to demonstrate that the generative model can learn to replicate many relevant features of the real market.

In the evaluation, comparisons of several relevant statistical features of the limit order book are made for the original and generated data. This includes frequencies and transition rates of different types of events, price distributions, order imbalance, price signatures and execution prices. In addition, the capability of the generative model to predict changes in the mid price is evaluated against state-of-the-art benchmarks based on deep learning for price prediction. Although the generative model is not trained specifically for price prediction it performs comparably to the best available deep learning algorithms.

This paper is outlined as follows. Section 2 presents additional background information about limit order books and the continuous time Markov chain model. The training data sets, constructed from generated data from the continuous time Markov chain model and real market data from the Nasdaq Stockholm exchange, are described in Section 3. The generative model, based on RNNs, is described in detail in Section 4, whereas the evaluation methods are presented in Section 5. Finally, the results of the evaluation and implementation details are provided in Section 6 and a short conclusion is given in Section 7.

2. The limit order book

In this section, an introduction to the modelling of a limit order book (LOB) is provided. First, the basic structure of LOBs is detailed, by describing the possible transitions due to incoming orders by market agents. A simple Markov chain model, which is a slight extension of the model proposed by Cont *et al.* (2010), is introduced and used as a reference model throughout this paper. Finally, additional features and variations of LOBs are discussed.

The LOB contains all posted volume at each price level for a specific asset on a market venue. The price levels are discrete and every asset has a given tick size that decides the difference between two adjacent price levels. There are three different types of orders that can change the current state of the LOB.

- *Market Order (MO)*: An aggressive order is matched against volume posted in the LOB and a trade is made.
- *Limit Order (LO)*: Volume is posted passively at one of the price levels available to buy or sell for.
- *Cancellation*: An already posted LO is cancelled and the corresponding volume is removed.

To distinguish between buy and sell orders the convention that negative volume represents buy limit orders and positive volume is representing sell limit orders will be used throughout the paper. We assume that it always holds that the highest price of the buy limit orders is lower than the lowest price of the sell limit orders. At each state of the LOB, the best bidding price is the best price level with at least one buy limit order, that is the highest price level with negative volume. This price level is called the bid and denoted by x^{bid}. Similarly, the best asking price is the best price level with at least one sell limit order, that is, the lowest price level with positive volume and this price level is called the ask and denoted by x^{ask}. The mid price is defined as the mean of the bid and ask and the quoted spread is the absolute difference between the bid and ask:

$$x^{\text{mid}} = \frac{x^{\text{bid}} + x^{\text{ask}}}{2} \tag{1}$$

$$x^{\text{spread}} = x^{\text{ask}} - x^{\text{bid}} \tag{2}$$

An example of an LOB is shown in figure 1 including its bid, ask, mid price and spread.

There are several ways of representing an LOB in array form. In theory, there is an infinite number of possible price levels for an asset, but in practice the price levels to consider must be truncated. In this paper, an LOB will be represented by the following two arrays: one for the asking side and one for the bidding side. The first one represents the asking side and is given by $(x^{\text{ask}}, y_{x^{\text{bid}}+1}, \ldots, y_{x^{\text{bid}}+d})$ where $y_{x^{\text{bid}}+i}$ is the volume at the price level i levels higher than the current bid and d is the number of levels to consider for each side. Similarly, the array for the bidding side is given by $(x^{\text{bid}} - x^{\text{ask}}, y_{x^{\text{ask}}-1}, \ldots, y_{x^{\text{ask}}-d})$ where $y_{x^{\text{ask}}-i}$ is the volume at the price level i levels lower than the current ask.

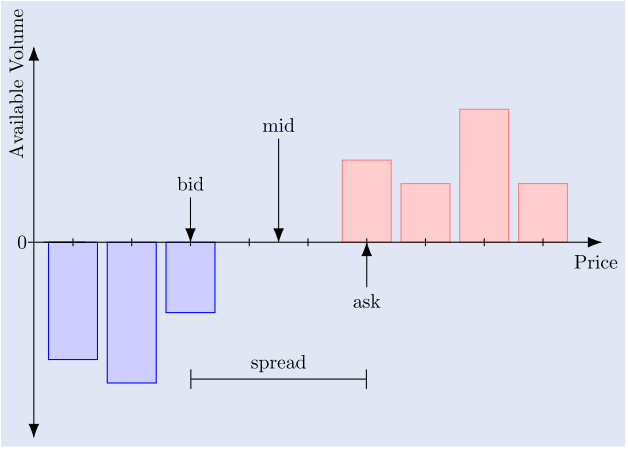

Figure 1. Example of a limit order book.

As orders are submitted by market agents the state of the LOB changes dynamically, each transition being the result of a submitted order. To manage the changes, a matching engine is used. Each market exchange has an algorithm establishing which transitions and trades that are possible. Most markets use a price-time priority to decide which LOs to match against an arriving MO. With price-time priority, the LOs offering the best price will be matched first against an MO, and among these the LO arriving earliest is first in the queue.

The dynamic evolution of the LOB is the aggregated result of all traders on the market. To explain the observed trading patterns of an LOB three primary classes of traders from high-frequency trading algorithms point of view are described in Cartea *et al.* (2015). The first one is fundamental traders, who are driven by economic fundamentals outside of the exchange. These traders are assumed to have very little short-term price information and can therefore be considered noise traders or liquidity traders on short time horizons. The second class of traders is informed traders, who make profit by leveraging information not yet reflected in the market prices. The third class is market makers, these are professional traders who facilitate exchange in assets, even though there is no centralized market maker in an exchange using LOBs.

2.1. A Markov chain model of the LOB

As a reference model of the LOB the model, we use a slight extension of the model introduced in Cont *et al.* (2010). The extension we use allows for random order sizes and has also been used for modelling the LOB of the EUR/USD exchange rate on a major foreign exchange market in Hult and Kiessling (2010). It models the LOB as a continuous time Markov chain whose state is the available volume at each

price level. The motivation for using a Markov chain model is that it is relatively easy to calibrate the model, using maximum likelihood, and value-iteration methods can be used to compute optimal execution strategies, at least if the state space is not prohibitively large. In the Markov chain model, transitions are due to the arrival of limit orders, market orders and cancellations of limit orders, where the arrival rate of each type of transition is a function of the state and the distributions of the size of market and limit orders are given by geometric distributions.

The model assumes that there is a fixed number of possible price levels denoted $n \in \mathbb{N}$, with $n \geq 2$, and the Markov chain consists of the current volumes at each price level at time t, $Y_t = (Y_{t,1}, \ldots, Y_{t,n})$ with $Y_{t,j} \in \mathbb{Z}$ for $j = 1, \ldots, n$. The space of feasible states for the LOB is denoted $\mathcal{S}$ and is a subset of $\mathbb{Z}^n$. With this notation, the bid and ask price levels for $y \in \mathcal{S}$ are defined by

$$x^{\text{bid}}(y) = \max\{x : y_x < 0\}, \quad \text{and}$$
$$x^{\text{ask}}(y) = \min\{x : y_x > 0\}.$$

It is assumed that $y_n > 0$ and $y_1 < 0$ for all $y \in \mathcal{S}$ and also that $x^{\text{bid}} < x^{\text{ask}}$ always holds.

The transition rates of the Markov chain are associated with arrivals of market and limit orders as well as cancellations and are parametrized by the state. Market orders have a state-independent rate, buy (sell) limit orders have rates that depends on the distance from the ask (bid), whereas cancellation buy (sell) rates are depending on the distance to the ask (bid) and proportional to the volume at that price level. More precisely, let $\hat{x} \in \mathbb{Z}^n$ be the unit vector with 1 in the xth coordinate and zeros elsewhere. Then the transition rates from a given state y of the LOB are given by,

$$\begin{aligned}
&\textit{sell limit order:} && y \to y + s\hat{x}, \quad x > x^{\text{bid}}(y), \quad s \geq 1,\\
&&& \text{with rate } p_L(s)\lambda_L^S(x - x^{\text{bid}}(y)),\\
&\textit{buy limit order:} && y \to y - s\hat{x}, \quad x < x^{\text{ask}}(y), \quad s \geq 1,\\
&&& \text{with rate } p_L(s)\lambda_L^B(x^{\text{ask}}(y) - x),\\
&\textit{cancel sell order:} && y \to y - \hat{x}, \quad x \geq x^{\text{ask}}(y),\\
&&& \text{with rate } \lambda_C^S(x - x^{\text{bid}}(y))\,|y_x|,\\
&\textit{cancel buy order:} && y \to y + \hat{x}, \quad x \leq x^{\text{bid}}(x),\\
&&& \text{with rate } \lambda_C^B(x^{\text{ask}}(x) - x)\,|y_x|,\\
&\textit{sell market order:} && y \to y + s\hat{x}, \quad x = x^{\text{bid}}(y), \quad 1 \leq s \leq |y_x|,\\
&&& \text{with rate } p_M(s)\lambda_M^S,\\
&\textit{buy market order:} && y \to y - s\hat{x}, \quad x = x^{\text{ask}}(y), \quad 1 \leq s \leq y_x,\\
&&& \text{with rate } p_M(s)\lambda_M^B,
\end{aligned}$$

where $\lambda_L^S(j), \lambda_L^B(j), \lambda_C^S(j), \lambda_C^B(j)$, $j \geq 1$ and λ_M^S, λ_M^B are non-negative parameters. Furthermore, s is the size of the order and $p_L(s)$ and $p_M(s)$ represent the size distribution for LOs and MOs, respectively, that have geometric distributions, such that,

$$p_L(s) = (\exp(\gamma_L) - 1)\exp(-\gamma_L s), \quad \text{and}$$
$$p_M(s) = (\exp(\gamma_M) - 1)\exp(-\gamma_M s),$$

for $s = 1, 2, \ldots$ and some $\gamma_L, \gamma_M > 0$. Throughout this paper, $\exp(\cdot)$ is used to denote the natural exponential function, and $\log(\cdot)$ will be used to denote the natural logarithm.

In the original article, the parameters are calibrated to 120 minutes of EUR/USD trade data where volumes are traded in units of million USD. The estimated parameters are displayed in table 1, with all other parameters assumed to be zero.

2.2. Extensions

The description of the LOB above is a simplification and certainly does not cover all existing LOBs. There are several additional features and variations of LOBs used in practice. Some of these are briefly mentioned in this section, see Gould *et al.* (2013) for a more comprehensive review.

Market fragmentation. A specific asset can often be traded on several different electronic trading platforms. This is referred to as market fragmentation. These different LOBs for the same asset cannot be considered independent, but should rather be modelled jointly.

Priority mechanism. There are exchanges with other priority mechanisms for limit orders than the price-time priority explained above. Pro-rata is another priority mechanism that is commonly used on futures markets. When there are several limit orders at the best price, all of these orders receive a share of the matching. The share is proportional to the order's fraction of the total volume available at the best price. Other possible priority mechanisms include price-size priority.

Hidden liquidity. Many exchanges provide functionality to allow traders to conceal their intention to trade. A common way is to offer other types of limit orders, such as hidden orders which do not display the quantity and icebergs that only partially display the quantity. There are also so-called dark pools. Some dark pools work similarly to a regular LOB, but all limit orders are entirely hidden. In other dark pools, orders are submitted only with a quantity and a direction, but without a specific price. These orders are then prioritised by arriving time and the matching occurs at the mid price of another LOB for the same asset.

Auctions. Another common feature of market exchanges are opening and closing auctions. The regular LOB trading is then suspended at the beginning and end of the trading day, and instead an auction system is used to match orders.

3. Description of the training data

The model proposed in this paper is modelling the full dynamics of the limit order book, which implies that the training data set must contain information about every single transition. Unfortunately, the well-studied FI-2010 data set is not sufficiently detailed to use as training data and we are not aware of other publicly available data sets that meet the requirements. To train the model two different sets of data are used. The first data set is synthetic, obtained by generating transitions from the continuous time Markov chain model described in Section 2.1. The second data set uses historical transition data from the Nasdaq Stockholm exchange.

Table 1. Parameters for the Markov chain model regulating the arrival rates of different types of orders as well as the size of the orders.

j:	1	2	3	4	5	Other parameters	
$\lambda_L^B(j)$	0.1330	0.1811	0.2085	0.1477	0.0541	λ_M^S	0.0467
$\lambda_L^S(j)$	0.1442	0.1734	0.2404	0.1391	0.0584	λ_M^B	0.0467
$\lambda_C^B(j)$	0.1287	0.1057	0.0541	0.0493	0.0408	γ_M	0.4955
$\lambda_C^S(j)$	0.1308	0.1154	0.0531	0.0492	0.0437	γ_L	0.5667

3.1. Synthetic Markov chain model data

The data set consists of synthetic data simulated according to the continuous time Markov chain model described in Section 2.1 with the parameters given in table 1, with some simplifying adjustments. To stabilize the simulation, the rates for buy and sell LOs are set to the mean of the corresponding buy and sell parameters in the table.

At each point in time, the first 10 price levels above the bid and the first 10 price levels below the ask are considered with the tick size being equal to 1. The available volume at price levels outside of this range is assumed to always be of unit size, with negative volume for price levels on the bidding side and positive volume for price levels on the asking side. With this assumption, the ask and bid will always be finite.

The data set used for training the RNN model contains a sequence of 2, 000, 000 events. These were obtained by first starting out with an LOB with unit volume on price levels above and at the ask, and negative unit volume on all price levels below the ask. The LOB is first simulated for 100, 000 events which are not used in the data set, since this period is considered the burn-in period. After the burn-in period, the following 2, 000, 000 events are used to form the training data set. The price levels are indexed such that the ask of the initial LOB of the training data set is labelled as zero.

3.2. Nasdaq data

The Nasdaq data is provided by the Swedish House of Finance Research Data Center† in the form of Nasdaq Historical ITCH files.‡ The data set is formed from, what is called, the order book view, which is composed of an ordered list of messages for every order added or edited for an instrument. Using these messages, it is possible to reconstruct the complete history of transitions of the LOB.

The instrument chosen is the SEB A stock on the Nasdaq Stockholm exchange and the complete history over 20 trading days during October 2019 is extracted as training data. These are all trading days available at the Swedish House of Finance Research Data Center, except of October 23rd which is considered an outlier and excluded. At this date, the SEB's quarterly finance report was released, causing the behaviour of the LOB to be considerably different to the other trading days.

To obtain satisfactory quality of the training data, some data cleaning is needed. Messages are removed if any of the following conditions are satisfied:

- Messages within 10 minutes from the beginning as well as the end of each day. This is due to the fact that the evolution of LOB just after opening the market, as well as prior to closing the market, is significantly different from other trading times. This removes 5.31% of the events. However, the orders from the first 10 minutes are used to form the initial state of the LOB.
- Messages for cancellation or MOs without information on when the corresponding LO was added. This removes 0.66% of the events.
- Messages concerning LOs that were not fully executed and/or cancelled but with LOs arriving later being executed at that price level. This should not be possible since later orders are placed after existing order in the queue of that price level. This removes 3.07% of the events.
- Messages concerning buy (sell) LOs that were not fully executed and/or cancelled but with sell (buy) LOs arriving later at the same price level. This removes 0.01% of the events.
- Messages concerning sell LOs on price levels lower or equal to the bid and similarly buy LOs on price levels higher or equal to the ask. This removes 0.32% of the events.

From the remaining messages, a sequence for each day is formed by considering the first 30 price levels above the bid and the first 30 levels below the ask at each point in time. The available volume at price levels outside of this range is assumed to be zero. This assumption leads to ignoring 0.75% of the events. The price levels are indexed such that the initial ask is set to be zero at each day.

The cleaning of the data removes in total 10.12% of the events. For each day, the total number of events after the cleaning ranges between 53, 056 and 112, 260 , with the total number of events for all days being 1, 462, 761.

Since the data contains orders with large sizes, some with order sizes larger than 50, 000, the data is normalized by dividing with the standard deviation of the sizes of all orders in the filtered training data. Note that both the order sizes

† https://www.hhs.se/en/houseoffinance/data-center/
‡ https://www.nasdaq.com/docs/2020/04/03/Nordic-Equity-TotalView—ITCH-3.03.5.pdf

themselves as well as the available volume at each price level are normalized in the same way.

4. A generative model of the LOB based on RNNs

This section contains a detailed description of the generative model proposed in this paper, including the network architecture, training and generation of LOB sequences.

Every transition of the LOB is characterized by an event type, a price level, an order size and a point in time. Rather than parametrizing the joint distribution of all of these variables directly, the joint probability of a transition is decomposed as a product of conditional probabilities, where the probability of event type is conditional on the state and history of the order book, the price level is conditional on the event type, state and history, the order size is conditional on the price level, event type, state and history, and the time delay from the last transition is conditional on the order size, price level, event type, state and history. In an LOB with d price levels tracked for each bid and ask side, consider the ith transition of the LOB and let e_i denote its event type (market order buy/sell, limit order buy/sell, cancel buy/sell), l_i the price level, s_i the order size and t_i the time from the most recent transition. With $\mathcal{F}_{i-1}$ representing the sigma field generated by all previous $i-1$ transitions, the joint probability of a transition is decomposed as the product

$$p(e_i, l_i, s_i, t_i \mid \mathcal{F}_{i-1}) = p(e_i \mid \mathcal{F}_{i-1})\, p(l_i \mid \mathcal{F}_{i-1}, e_i) \times p(s_i \mid \mathcal{F}_{i-1}, e_i, l_i) p(t_i \mid \mathcal{F}_{i-1}, e_i, l_i, s_i). \tag{3}$$

Note that, as shown later, the conditional distributions of the order size and the time from the most recent transition may be discrete or continuous. In the latter case, these distributions are absolutely continuous with respect to the Lebesgue measure, and in the above decomposition, $p(s_i \mid \mathcal{F}_{i-1}, e_i, l_i)$ and $p(t_i \mid \mathcal{F}_{i-1}, e_i, l_i, s_i)$ are the corresponding densities.

To model the conditional probabilities, separate RNNs will be used for each term in the product. Each of the RNNs will have its individual set of parameters and input variables and the index $k \in \{\text{event}, \text{level}, \text{size}, \text{time}\}$ is used to denote the RNN used to model the corresponding conditional probability in (3). The RNN model assumes that the relevant information contained in the sigma field $\mathcal{F}_{i-1}$ is captured by a smaller sigma field generated by the variables $a_{i-1}, f_{i-1}, h_{i-1}^{(k)}$, where $a_{i-1} = \{e_{i-1}, l_{i-1}, s_{i-1}, t_{i-1}\}$ contains the information about the most recent transition, f_{i-1} is a vector containing features of the present state of the LOB as explained in Section 2 and $h_{i-1}^{(k)}$ is a vector representing the internal memory of the RNN associated with the kth term of the product.

For each $k \in \{\text{event}, \text{level}, \text{size}, \text{time}\}$, let $g^{(k)}(h_{i-1}^{(k)}, \cdot)$ represent the RNN that maps the input, at the ith transition, to the output, while updating the internal memory $h_{i-1}^{(k)}$ to $h_i^{(k)}$. The outputs

$$\phi_i^{(\text{event})} = g^{(\text{event})}(h_{i-1}^{(\text{event})}, f_{i-1}, a_{i-1}),$$
$$\phi_i^{(\text{level})} = g^{(\text{level})}(h_{i-1}^{(\text{level})}, f_{i-1}, a_{i-1}, e_i), \quad \text{etc.}$$

provide the parameters for the corresponding conditional probability in the decomposition (3) and are described in detail in Section 4.2. A schematic graphical representation of the model is shown in figure 2.

Table 2. Network architecture hyperparameters for both datasets.

Hyperparameter	MC model data	Nasdaq data
Number of LSTM layers	2	2
Units per LSTM layer	32	32
Length of training sequences	20	20
Dropout	0.1	0.1
Learning rate	10^{-5}	10^{-4}
Batch size	128	128

4.1. *Network architecture*

The whole model consists of four RNN submodels, corresponding to the four conditional probabilities in (3). It is, of course, possible to use different network architectures for each RNN, but for simplicity the same architecture is used for all four RNNs. In table 2, the parameters of the network architecture are summarized.

Each RNN consists of a number of LSTM layers with layer normalization (Ba *et al.* 2016) followed by a dense layer and finally a masking layer. The masking layer makes sure that impossible samples are given the probability zero. Without this masking step, it could happen that during the generation there is an impossible transition sampled, not representing any of the transitions described in Section 2.1, that leads to an impossible state of the order book. Ideally, the probability of these impossible events happening during generation would already be low due to the training of the model, but to completely avoid these events their corresponding probabilities are set to zero in the masking layer and the remaining of the probabilities are then normalized to sum to 1.

The input to each network is described in figure 2, with the only processing being that variables modelled by categorical distributions are transformed using one-hot encoding. Also, in table 3 the number of inputs and outputs as well as number of trainable parameters for each RNN are summarized.

4.2. *Modelling the conditional probabilities*

Each of the conditional probabilities for e_i, l_i, s_i and t_i in (3) is parametrized by a vector, $\phi_i^{(k)}, k \in \{\text{event}, \text{level}, \text{size}, \text{time}\}$, whose values are the outputs of the corresponding RNN. These conditional probabilities are specified in detail in the following.

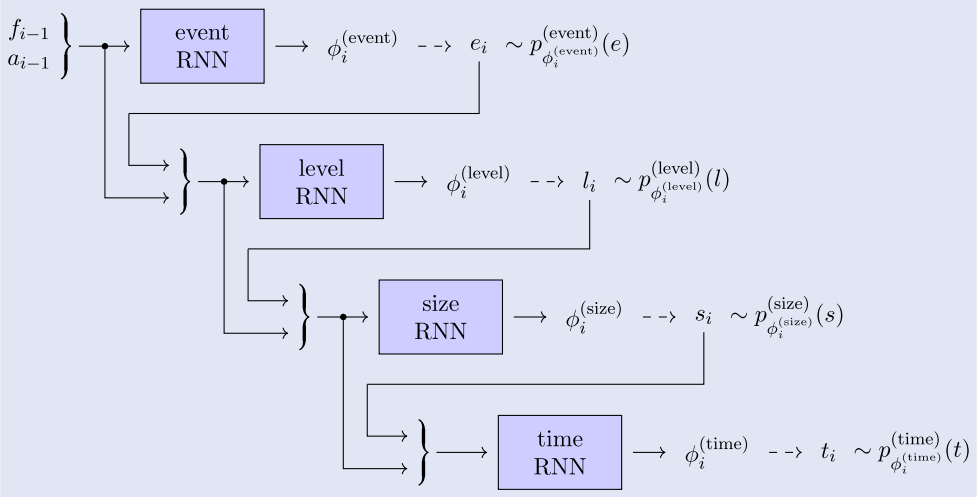

Figure 2. Graphical representation of the generative model.

Table 3. Network parameters for each submodel including number of inputs, number of outputs and total number of trainable parameters for each RNN.

	Markov chain model data		
	Inputs	Outputs	Trainable parameters
Event type	64	6	22,086
Price level	70	10	22,986
Order size	80	25	24,761
Time delay	105	1	27,169
	Nasdaq data		
	Inputs	Outputs	Trainable parameters
Event type	312	6	53,830
Price level	318	30	55,390
Order size	348	107	61,771
Time delay	455	107	75,467

4.2.1. Event type. The conditional distribution of the event type is a categorical distribution with six classes representing:

MO bid / MO ask / LO buy /
LO sell / Cancel LO buy /Cancel LO sell.

The event type network takes current order book features f_{i-1} and preceding event data a_{i-1} as input, updates the internal memory, $h_i^{(\text{event})}$, and outputs a vector $\phi_i^{(\text{event})}$ with six elements representing the unnormalized log probabilities,

$$\phi_i^{(\text{event})} = g^{(\text{event})}(h_{i-1}^{(k)}, f_{i-1}, a_{i-1}),$$

where $g^{(\text{event})}$ represents the event type RNN. The conditional probability of event type e_i in the ith transition is given by

$$p\left(e_i \mid \mathcal{F}_{i-1}\right) = p^{(\text{event})}_{\phi_i^{(\text{event})}}(e_i) = \varsigma\left(\phi_i^{(\text{event})}\right)_{e_i}, \quad e_i \in \{1, \ldots, 6\},$$

where ς denotes the softmax function, defined as $\varsigma(z)_j = \frac{\exp z_j}{\sum_{k=1}^{N} \exp z_k}$ for the jth component of $z \in \mathbb{R}^N$.

4.2.2. Price level. The conditional distribution of the price level is also a categorical distribution. When the event type e_i is LO buy or Cancel buy, the price level is represented by l_i, defined as the distance (the number of price levels) between the price level and the ask. Similarly, when the event type e_i is LO sell or Cancel sell, l_i is the distance between the price level and the bid. For market orders, the level is identified by the event type, since market orders always occur at the bid or ask and $l_i = 0$ in this case.

Note that the number of possible price levels is bounded by the total number of price levels on each side, d. Therefore the support of the conditional distribution of l_i has d price levels. The probability of some of them is, by definition of l_i, set to zero in the masking layer of the price level network.

The price level network takes current order book features f_{i-1}, preceding event data a_{i-1} and new event type e_i as input, updates the internal memory $h_{i-1}^{(\text{level})}$ to $h_i^{(\text{level})}$, and outputs a vector $\phi_i^{(\text{level})}$ with d elements representing the unnormalized log probabilities,

$$\phi_i^{(\text{level})} = g^{(\text{level})}(h_{i-1}^{(\text{level})}, f_{i-1}, a_{i-1}, e_i),$$

such that the conditional probability of price level l_i in the ith transition is given by

$$p\left(l_i \mid e_i, \mathcal{F}_{i-1}\right) = p^{(\text{level})}_{\phi_i^{(\text{level})}}(l_i) = \varsigma\left(\phi_i^{(\text{level})}\right)_{l_i}, \quad l_i \in \{1, \ldots, d\}.$$

4.2.3. Order size. The conditional distribution of the order size may be discrete or continuous, depending on the type of asset under consideration. If the posted volume on the order book tends to be small it is appropriate to use a discrete distribution. This is the case on some foreign exchange markets where the unit size of orders may be in million USD. On equity markets, on the other hand, the unit is often in number of shares, leading to high volumes, which makes it suitable to approximate the order size by another distribution that could be either discrete or continuous.

Markov chain data. For the synthetic Markov chain data, the parameters are selected to resemble a EUR/USD LOB

with relatively small volume. The conditional distribution of the order size is modelled as a discrete distribution, with the maximum possible order size set to be $s_{\max} = s_{\max}^{(\text{train})}$, the maximum order size in the training data.

For cancellations and market orders, the order size is bounded by the current available volume on the given price level, and the corresponding probabilities of order sizes larger than the available volume are set to zero in the masking layer of the order size network.

The order size network takes current order book features f_{i-1}, preceding event data a_{i-1}, new event type e_i and new price level l_i as input, updates the internal memory $h_{i-1}^{(\text{size})}$ to $h_i^{(\text{size})}$, and outputs a vector $\phi_i^{(\text{size})}$ with $s_{\max}$ elements representing the unnormalized log probabilities,

$$\phi_i^{(\text{size})} = g^{(\text{size})}(h_{i-1}^{(\text{size})}, f_{i-1}, a_{i-1}, e_i, l_i),$$

such that the conditional probability of the order size of the ith transition is given by

$$p(s_i \mid e_i, l_i, \mathcal{F}_{i-1}) = p_{\phi_i^{(\text{size})}}^{(\text{size})}(s_i) = \varsigma\left(\phi_i^{(\text{size})}\right)_{s_i}, \quad s_i \in \{1, \dots, s_{\max}\}.$$

Nasdaq data. For the Nasdaq data, the conditional distribution of the order size is modelled by a categorical distribution, but since the range of order sizes for the Nasdaq data is very large, the order sizes are binned into a more reasonable number of categories, with the choice of $c_s = 100$ categories used throughout this paper. The method for binning aims at having the same number of training samples in each bin, but takes into account that many training samples have identical order sizes and forms bins including only these common values in these cases. The full method is described in Algorithm 1. Additionally, bins larger than a threshold of 10 are split into two bins of equal size. Sampling from the distribution is then performed by first sampling the bin from the categorical distribution and then sampling a value uniformly from the interval corresponding to that bin.

For LOs, the order size is binned directly using Algorithm 1, while for other order types, the order size is modelled by its relative size to the available volume at the given price level, $\hat{s}_i \in (0, 1]$. The binning of $\hat{s}_i$ is also performed using Algorithm 1, separately for MOs and cancellations.

The order size network takes current order book features f_{i-1}, preceding event data a_{i-1}, event type e_i and price level l_i as input, updates the internal memory $h_{i-1}^{(\text{size})}$ to $h_i^{(\text{size})}$ and outputs a vector $\phi_i^{(\text{size})}$ with c_s elements representing the unnormalized log probabilities,

$$\phi_i^{(\text{size})} = g^{(\text{size})}(h_{i-1}^{(\text{size})}, f_{i-1}, a_{i-1}, e_i, l_i).$$

The conditional probability of the order size of the ith transition is then given by

$$p(s_i \mid e_i, l_i, \mathcal{F}_{i-1}) = p_{\phi_i^{(\text{size})}}^{(\text{size})}(s_i) = \begin{cases} \varsigma\left(\phi_i^{(\text{size})}\right)_{s_i}, & \text{if } e_i \text{ is LO}, \\ \varsigma\left(\phi_i^{(\text{size})}\right)_{\hat{s}_i}, & \text{otherwise.} \end{cases}$$

Algorithm 1: Binning of a variable.

Input: $\{d_i\}_{i=1}^n, c$
Output: $\{b_i\}_{i=1}^c$
$i \leftarrow 0$
$x \leftarrow \{d_i\}_{i=1}^n$
$v \leftarrow$ values in x occurring $\geq \frac{n}{c}$ times
while length$(v) > 0$ **do**
  for $k \in v$ **do**
    $i \leftarrow i + 1$
    $b_i \leftarrow \{k\}$
    remove from x all values equal to k
  end
  $v \leftarrow$ values in x occurring $\geq \frac{\text{length}(x)}{c-i}$ times
end
$b_{i+1} \leftarrow \left[\min x, \frac{1}{c-i}\text{-quantile of } x\right] \setminus \{b_m\}_{m=1}^i$
for $j \leftarrow i + 2$ **to** c **do**
  $b_j \leftarrow$
  $\left(\frac{j-i-1}{c-i}\text{-quantile of } x, \frac{j-i}{c-i}\text{-quantile of } x\right] \setminus \{b_m\}_{m=1}^i$
end
return $\{b_i\}_{i=1}^c$

4.2.4. Time delay since the previous transition. The parametric family for the conditional distribution of the time delay since the previous transition is selected after inspecting the training data and differs between the two data sets considered in this paper.

Markov chain data. For the synthetic Markov chain model data, the time delay between the events is modelled by an exponential distribution whose rate depends on the input parameters, with a maximum time given by multiplying the maximum time in the training data with a constant $c_t = 1.2$. The maximum time will not impact the likelihood on the training data, and during generation, if a sample from the exponential distribution is larger than the maximum time, it will simply be reduced to the maximum time.

The time delay network takes current order book features f_{i-1}, preceding event data a_{i-1}, event type e_i, price level l_i and order size s_i as input, updates the internal memory $h_{i-1}^{(\text{time})}$ to $h_i^{(\text{time})}$ and outputs a single $\phi_i^{(\text{time})}$ element representing the rate parameter of the exponential distribution,

$$\phi_i^{(\text{time})} = g^{(\text{time})}(h_{i-1}^{(\text{time})}, f_{i-1}, a_{i-1}, e_i, l_i, s_i).$$

The conditional probability of the time delay since the previous transition is given by

$$p(t_i \mid e_i, l_i, s_i, \mathcal{F}_{i-1}) = p_{\phi_i^{(\text{time})}}^{(\text{time})}(t_i) = \left(\left(1 + \phi_i^{(\text{time})}\right)^2 + \epsilon\right) \times \exp\left(-\left(\left(1 + \phi_i^{(\text{time})}\right)^2 + \epsilon\right) t_i\right), \quad t_i \geq 0,$$

with $\epsilon = 10^{-5}$.

Nasdaq data. For the Nasdaq data, the time delay since the previous transition is modelled by a categorical distribution,

with the time delays binned into a reasonable number of categories. Throughout this paper $c_t = 100$ categories are used. As for the order size, the binning of the time delays is performed using Algorithm 1 on the time delays in the training data and the sampling is done as explained for the order size. As for order sizes, bins larger than 10 are split into two.

The time delay network takes current order book features f_{i-1}, preceding event data a_{i-1}, event type e_i, price level l_i and order size s_i as input, updates the internal memory $h_{i-1}^{(\text{time})}$ to $h_i^{(\text{time})}$, and outputs a vector $\phi_i^{(\text{time})}$ with c_t elements representing the unnormalized log probabilities,

$$\phi_i^{(\text{time})} = g^{(\text{time})}(h_{i-1}^{(\text{time})}, f_{i-1}, a_{i-1}, e_i, l_i, s_i).$$

The conditional probability of the time delay since the previous transition is given by

$$p(t_i \mid e_i, l_i, s_i, \mathcal{F}_{i-1}) = p_{\phi_i^{(\text{time})}}^{(\text{time})}(t_i) = \varsigma\left(\phi_i^{(\text{time})}\right)_{t_i}, \quad t_i \in \{1, \ldots, c_t\}.$$

4.3. Training the generative model

The model is trained by maximizing the joint likelihood using backpropagation through time. The joint likelihood of all transitions of the LOB is computed by taking the product of the conditional distributions in (3), described in detail above, over all transitions. For a sequence of limit order book events characterized by event types $\mathbf{e} = \{e_i\}_{i=1}^n$, price levels $\mathbf{l} = \{l_i\}_{i=1}^n$, order sizes $\mathbf{s} = \{s_i\}_{i=1}^n$ and time delays $\mathbf{t} = \{t_i\}_{i=1}^n$ initialized with order book features f_0 and event $a_0 = \{e_0, l_0, s_0, t_0\}$ the joint likelihood is

$$p(\mathbf{e}, \mathbf{l}, \mathbf{s}, \mathbf{t} | f_0, a_0) = \prod_{i=1}^{n} p_{\phi_i^{(\text{event})}}^{(\text{event})}(e_i) p_{\phi_i^{(\text{level})}}^{(\text{level})}(l_i) p_{\phi_i^{(\text{size})}}^{(\text{size})}(s_i) p_{\phi_i^{(\text{time})}}^{(\text{time})}(t_i),$$

where $\phi_i^{(k)}$, $k \in \{\text{event}, \text{level}, \text{size}, \text{time}\}$ is the output of the event type, price level, order size and time delay RNNs, respectively at the ith transition. For numerical stability, it is more robust to work with the log likelihood.

The training is performed using the Adam optimizer, introduced in Kingma and Ba (2015), with the negative log likelihood as loss function. The training is interrupted once the log likelihood on the validation data has not been increased for 10 consecutive epochs and the weights are restored from the epoch with lowest validation loss. The hyperparameters for the training can be found in table 3. In the training of the model, the last 10% of the data is used as validation data and the rest as training data. For the Nasdaq data set, the last 10% is in terms of the last 10% of the days, leading to the last 2 days being validation data and the rest training data.

As usual with neural networks, the training is not guaranteed to find a global maximum of the likelihood and it is common that different initializations lead to different final weights of the model. It is also not always the case that a high likelihood directly corresponds to a well-behaving model and it is recommended to inspect the results of the model on the different evaluation metrics introduced later on to determine if the training actually found a desirable model.

4.4. Generating LOB dynamics using the RNN model

The procedure for generating data from the model is described in Algorithm 2. The generation is initialized with an event and continued until a specified end time T is reached. The output is the event information and order book for each event until the end time is reached.

Algorithm 2: Generating the dynamic evolution of an LOB.

Input: $e_0, l_0, s_0, t_0, f_0, T$
Output: $\{e_i, l_i, s_i, t_i, f_i\}_{i=0}^{n_T}$
Reset memory of all RNNs:
$h_0^{(\text{event})}, h_0^{(\text{level})}, h_0^{(\text{size})}, h_0^{(\text{time})}$
$i \leftarrow 0$
$\tau \leftarrow 0$
while $\tau < T$ **do**
  $i \leftarrow i + 1$
  $a_{i-1} \leftarrow \{e_{i-1}, l_{i-1}, s_{i-1}, t_{i-1}\}$
  $\phi_i^{(\text{event})} \leftarrow g^{(\text{event})}(h_{i-1}^{(\text{event})}, f_{i-1}, a_{i-1})$
  Sample $e_i \sim p_{\phi_i^{(\text{event})}}$
  $\phi_i^{(\text{level})} \leftarrow g^{(\text{level})}(h_{i-1}^{(\text{level})}, f_{i-1}, a_{i-1}, e_i)$
  Sample $l_i \sim p_{\phi_i^{(\text{level})}}$
  $\phi_i^{(\text{size})} \leftarrow g^{(\text{size})}(h_{i-1}^{(\text{size})}, f_{i-1}, a_{i-1}, e_i, l_i)$
  Sample $s_i \sim p_{\phi_i^{(\text{size})}}$
  $\phi_i^{(\text{time})} \leftarrow g^{(\text{time})}(h_{i-1}^{(\text{time})}, f_{i-1}, a_{i-1}, e_i, l_i, s_i)$
  Sample $t_i \sim p_{\phi_i^{(\text{time})}}$
  $f_i \leftarrow$ updated according to $\{f_{i-1}, e_i, l_i, s_i, t_i\}$
  $\tau \leftarrow \tau + t_i$
end
$n_T \leftarrow i$
return $\{e_i, l_i, s_i, t_i, f_i\}_{i=0}^{n_T}$

5. Evaluation methods

For the proposed generative model to be useful, it is necessary that, after training, Algorithm 2 produces a realistic evolution of the LOB. To evaluate the goodness-of-fit of the proposed generative model, a number of statistics are compared for the original data and the generated data.

There is not a lot of earlier work on evaluation metrics for generated LOB data, but Vyetrenko *et al.* (2020) proposed a number of reference metrics. They focused on replicating stylized facts seen in the real markets. Some of the evaluation methods presented in this section are based on the metrics they proposed, however, additional evaluation methods are introduced to capture the general behaviour and not to single out stylized facts to replicate.

5.1. Frequencies and rates of events types and order sizes

A basic evaluation of the generative model is to compare the empirical frequencies of different event types in the original training data and the generated data from the calibrated generative model. We consider the empirical frequencies of MO bid/ask events, LO buy/sell events at different price levels

from the best ask/bid, and Cancel buy/sell events at different price levels from the best ask/bid.

For the Markov chain model we may, in addition, compare estimated transition rates for the different event types. More precisely, we compare the true transition rates that were used to generate the original data, with estimated transition rates for the original data and with estimated transition rates for the data generated by the calibrated generative model. The transition rates are estimated following the procedure described in Hult and Kiessling (2010). Similarly, the parameters of the order size distribution γ_L and γ_M are estimated for the original data from the Markov chain model and the generated data from the generative model and compared to the true parameters used to generate the original data. A slight deviation from the procedure described in Hult and Kiessling (2010) is that for the maximum likelihood estimate of γ_M, the distribution of market order sizes is assumed to be a truncated distribution, since the order size is limited by the available volume on the bid or ask level. This leads to the maximum likelihood estimate not having an explicit expression, but it is easy to find an estimate using a numerical maximization of the likelihood.

In detail, assume that our data set includes n MOs with order sizes $S_1, \ldots, S_n$ each following a truncated geometric distribution with maximum size given by the current available volume on the best bid/ask level v_i. The probability for S_i being equal to j is given by

$$p(S_i = j) = \frac{\exp(\gamma_M) - 1}{1 - \exp(-\gamma_M v_i)} \exp(-\gamma_M j), \quad j = 1, \ldots, v_i$$

for some unknown $\gamma_M > 0$.

The joint log likelihood given $S_1 = s_1, \ldots, S_n = s_n$ and assuming independence:

$$\begin{aligned} &\log p(S_1 = s_1, \ldots, S_n = s_n) \\ &\quad = -\gamma_M \left(\sum_{i=1}^{n} s_i \right) + n \log(\exp(\gamma_M) - 1) \\ &\qquad - \sum_{i=1}^{n} \log(1 - \exp(-\gamma_M v_i)) \end{aligned}$$

If $v_i = \infty$ for all $i \in \{1, \ldots, n\}$, the maximum likelihood estimation of γ_M is given by $\hat{\gamma}_M = \log \frac{\sum_{i=1}^{n} s_i}{\sum_{i=1}^{n} s_i - n}$. However, if this is not the case, finding an explicit expression for $\hat{\gamma}_M$ is not obvious, and we resort to maximizing $\log p(S_1 = s_1, \ldots, S_n = s_n)$ with numerical methods using Brent's algorithm (Brent 1973).

5.2. Price distributions

To evaluate price distributions of the LOB, the marginal distribution of the mid price is considered for each point in time for the original data and the generated data. As an elementary comparison of the marginal distributions we consider the mean and standard deviation of the original data and the generated data at each point in time. An alternative comparison is to consider the Kolmogorov–Smirnov statistic for the mid-price returns, which is explained in more detail later in this section. Similarly, comparisons are made for the mean and standard deviation of the marginal distributions of the spread.

5.3. Stylized facts

As argued by Vyetrenko *et al.* (2020), replication of stylized facts is an important aspect to consider for the evaluation of LOB simulations. It should be noted that in our work a much shorter time horizon is considered compared to their setup, so it is not obvious that the same patterns will emerge in our datasets. From their experimental results, the adapted metrics that can be applied to our case are the following.

- Histogram of logarithm of the interarrival time between events
- Histogram of total number of events during 1 minute
- Histogram of 1 minute return distribution
- Histogram of autocorrelation of 10 seconds return with lag of one over each 1-minute sequence
- Autocorrelation of 10 seconds square returns as a function of time lag
- Histogram of correlation coefficient for each 1 minute sequence between the volatility, computed as the standard deviation of 1 second returns over 10 second disjoint windows, and the total order size for all events during the same 10 seconds.

5.4. Order imbalance

Another important feature of an LOB is the current order imbalance, as discussed in Cartea *et al.* (2015). The order imbalance for the top n levels is given by

$$\rho_n = \frac{V_n^{\text{bid}} - V_n^{\text{ask}}}{V_n^{\text{bid}} + V_n^{\text{ask}}},$$

where V_n^{bid} is the volume posted on the first n price levels on the bidding side and V_n^{ask} is the volume posted on the first n price levels on the asking side. Note that ρ_n is a number in the interval $[-1, 1]$. In this paper, the order imbalance is evaluated by considering the distribution of the order imbalance. The distribution of the order imbalance can be represented visually by constructing a histogram of the order imbalance at times $0, \Delta, 2\Delta, \ldots, T$ for some time difference Δ for each sequence of length T in the data set. For the experiments in this paper, the number of price levels is $n = 5$, the length of the sequences is $T = 60\,s$ and the time difference is set to $\Delta = 0.1\,s$.

Another relevant characteristic is the rate at which the order imbalance transitions between different regimes. Similarly to Cartea *et al.* (2015), the order imbalance regimes are formed as five intervals of equal length, such that ρ_n belongs to regime i if $\rho_n \in \left[\frac{2i-7}{5}, \frac{2i-5}{5}\right)$ and $\rho_n = 1$ belongs to regime 5. The frequencies of the LOB transitions from one regime to another, when sampled at equally spaced time points, are computed.

5.5. Market impact of market orders

5.5.1. Price signatures for each market order. When using the model to backtest an execution strategy, it is of high importance that the market impact of trades is realistic. One approach to evaluate the market impact is to consider what happens with the mid price after buy MOs and sell MOs. To this end the framework of price signatures described in Oomen (2019) is adopted.

For an MO, with order size s executed at time t_0, the price signature is given by

$$\tilde{x}(\delta) = \frac{x^{\text{mid}}_{t_0+\delta} - x^{\text{mid}}_{t_0}}{s}, \quad \text{for } \delta \in \left[0^-, T\right]$$

where x^{mid}_t is the mid price at time t and T is a chosen end time. Note that $x^{\text{mid}}_{t_0+\delta}$ for $\delta = 0^-$ is the mid price immediately before the market order is executed.

The price signatures are discretized and computed for each $\delta \in \{0^-, \Delta, 2\Delta, \dots, T\}$ for some time difference Δ for each MO executed at least T before the end of its sequence, except for MOs with order sizes lower than a threshold, which are removed to make the price signatures more stable. The price signatures are then aggregated separately for the bid and ask side, to compute and display their means.

For the experiments in this paper, the length of the sequences is $T = 30\,s$ and the time difference is set to $\Delta = 0.1\,s$. The threshold for the size of MOs is one for the Markov chain model data while for the Nasdaq data the threshold is set to 10 shares.

5.5.2. Price signatures when executing multiple market orders. Extending the procedure above, we evaluate the aggregated market impact when executing multiple MOs with some time delay between each order. For the Markov chain model, both the original Markov chain model and the trained RNN model can be used for simulation. This means that it is possible to interact with the simulation and study how the original Markov chain model behaves compared to the learned RNN model when executing multiple market orders with a fixed delay between each order.

For the Nasdaq data, it is not possible to evaluate the price signature of multiple orders, since there is no way of interacting with the real data except from placing orders on the real exchange, which is not feasible.

5.6. Kolmogorov–Smirnov statistic

So far, all mentioned evaluations have been rendering visualizations of different statistics of the LOB data, by comparing the statistics of the training data and generated data. However, it is also desirable to have more quantitative metrics to facilitate comparing different models easier. To this end, the Kolmogorov–Smirnov (KS) statistic is computed for the distribution of different statistics of the data. The KS statistic measures the largest difference of the cumulative distribution function for two samples,

$$D_{KS} = \sup_x |F_1(x) - F_2(x)|,$$

where F_1 and F_2 are the empirical cumulative distribution functions for the first and second samples. See Massey Jr (1951) for more details on the KS statistic.

The KS statistic is evaluated for all event types, price levels, order sizes and time delays respectively for the original and generated data. Furthermore, the KS statistic is also computed for order imbalance, spread size and mid-price returns for equally spaced time points of all sequences. In our experiments, the time points are $0.1\,s$ apart. Lastly, the KS statistic is also computed for volatility of $0.1\,s$ returns of the mid price over a specific time horizon chosen to be $5\,s$.

The KS statistics are benchmarked against a naive model, which uses the same type of distributions as the generative model as described in Section 4.2, but removing the dependence on the current state and history of the LOB. Instead the parameters of the distributions are estimated from the overall frequencies of the training data set.

5.7. Mid-price prediction

As mentioned in Section 1, there is an extensive literature on machine learning methods for LOB data where the main objective is the prediction of mid prices. Even though price prediction is not the main objective of our generative model, it can still be used for evaluation purposes and compared to the state-of-the-art methods such as the DeepLOB model introduced by Zhang *et al.* (2019). The DeepLOB model uses convolutional filters together with LSTM units, where the convolutional filters are supposed to capture the spatial structure of an LOB by finding filters extracting meaningful features of an LOB, while the LSTM units then capture the longer time dependencies.

Our setup is similar to the one described in Zhang *et al.* (2019), where the aim is to classify whether the mid price will go up, down or remain stationary during a prediction horizon of n events. For an event happening at time t_0, the average mid price over the latest n events is compared with the average mid price over the following n events. The evaluation is performed by computing

$$m_- = \frac{1}{n}\sum_{j=0}^{n-1} x^{\text{mid}}_{-j}, \quad m_+ = \frac{1}{n}\sum_{j=1}^{n} x^{\text{mid}}_{j}, \quad r = m_+ - m_-,$$

with x^{mid}_j being the mid price j events before t_0 if $j < 0$, the current mid price if $j = 0$ and the mid price j events after t_0 if $j > 0$.

In Zhang *et al.* (2019), the authors divide r by m_-, but this step is omitted here since the data is already centred such that the initial ask is zero and the price differences are computed in terms of number of ticks.

The change is labelled as up if $r > \Delta r$, down if $r < -\Delta r$ and stationary otherwise for some threshold Δr. The prediction horizon is set to $n = 50$, Δr is chosen such that the three classes are roughly balanced, for our data sets $\Delta r = 0.15$ for the Markov chain model data and $\Delta r = 0.45$ for Nasdaq data. To predict the mid price with the generative model, a number of trajectories are simulated all starting with the observed recent history, over the last 100 events, and simulated forward for another n time steps. Then r is computed for each of these

simulated trajectories and the mean of these are used as an estimate of the true value.

The DeepLOB model is trained on the same data sets as our generative model. The training is interrupted once the validation loss has not decreased for 10 epochs and the weights from the epoch with the lowest validation loss is used. Previously unseen data is used as test data. For the Markov chain model, the test data is constructed by a sequence of 100,000 events, from these 10,000 outcomes are sampled uniformly to form the test data set. For the Nasdaq data, the test data is constructed by the events from 4 November 2019. From these events, 10,000 outcomes are sampled uniformly to form the test data. The prediction results are compared using accuracy, recall, precision and F1-score.

6. Experiments

The model is implemented in Python using TensorFlow (Abadi *et al.* 2015) and the code is available at https://github.com/lobrnn/lob-rnn. For each dataset, the generative RNN model is trained with five different seeds and for the evaluations typically the average and standard deviation for these five runs are reported.

The data sets used for evaluation are created by choosing 1000 disjoint sequences of 60 seconds from the training data set. For each such sequence, the generative RNN model simulates a sequence of 60 seconds starting from the initial event of the original sequence and these simulated sequences form the data set for the generative RNN model.

6.1. Results on Markov chain model data

The motivation for training the RNN model on data generated from the Markov chain LOB model is to investigate to what extent the RNN model can learn to replicate the behaviour of the Markov chain model.

The loss during the training on the Markov chain model data is shown in figure A1, with each run represented by its own colour. For all five different seeds, the loss is well-behaving and decreasing nicely for all four submodels.

Frequencies of overall types of events are shown in figure A2 and more detailed event types including the distance from bid/ask are shown in figure A3, while the estimates of the true rates for the Markov chain model is shown in figure A4. For both frequencies and rates, the generated data is behaving very similarly to the original data.

For the mid price and the spread, the mean and standard deviation over time is shown in figures A5 and A6. As time increases, the standard deviations of the mid-price increase, while the means stay close at zero. The same pattern can be seen for both the original and the generated data. For the spread, the distribution looks rather constant over time, and again the generated data behaves very similarly as the original data.

In figure A7, different stylized facts are visualized for the original and the generated data, with the generated data closely resembling the original data for all of these.

The order imbalance histogram and transition probabilities of regime shifts are shown in figures A8 and A9. The histograms for the original data and generated data are close to identical. For the transition probabilities, the differences for the original data and the generated data are small, but somewhat larger for the regimes representing highly imbalanced LOBs. These cases are rare, and it is not clear if the generative RNN model is not as accurate for these cases or if there are just too few samples in these regions to get good estimates of the transition probabilities.

Price signatures for each MO with mean and standard deviation are shown in figure A10. For both the original and generated data, the mean of the price signature changes directly after the MO, in opposite directions for ask and bid MO and the price signatures are almost identical for the original and generated data.

The price processes for mid, ask and bid when executing five unit MOs with 10 seconds time delay are displayed in figure A12 and the corresponding execution cost is visualized in a histogram in figure A11. As for the single MO, the prices increase following each MO. Again, the original data and the generated data exhibit similar behaviour.

The KS statistics are compared in table A1 for the RNN model and the naive model. The RNN model outperforms the naive model for all distributions. Both models perform very well for the distributions of event type, price level, order size and time delay, which is to be expected since both models are directly maximizing the likelihood of these variables during the training. For the other distributions, the KS statistics are overall somewhat larger for both models, but the RNN model performs significantly better.

The results of comparing the performance of mid-price prediction between state-of-the-art deep learning algorithms and our generative RNN model are shown in table A2. The DeepLOB model is overall slightly better than our model on this task.

6.2. Results on Nasdaq data

The loss during the training on the Nasdaq data is shown in figure A13, with each run represented by its colour. Overall the loss for each submodel decreases satisfactory for both the training and validation loss for all of the five different seeds.

Frequencies of overall types of events are shown in figure A14 and more detailed event types including the distance from bid/ask is shown in figure A15. As for the Markov chain model data, the frequencies of the generated data are close to the frequencies of the original data, even though there is a bit more of a difference for the Nasdaq data.

For the mid price and spread, the mean and standard deviation over time is shown in figures A16 and A17. The pattern is analogous to the Markov chain model data, such that as time increases, the standard deviation of the mid price increases, while the mean stays at zero. The same pattern can be seen for both the original and the generated data, even though the mean of the generated data varies more for the generated data while the standard deviations of the generated data are slightly smaller. For the spread, the distribution looks rather

constant over time, and the generated data behaves very similarly as the original data with just slightly larger standard deviation.

In figure A18, the stylized facts are visualized for the original and the generated data. Overall, the generated data behaves similarly to the original data for all of these, but it can be noted that the original data has a fatter tail for the distribution of total number of events per minute and there is also some differences between the distributions of the volume and volatility correlation.

The order imbalance histogram and regime shifts are shown in figures A19 and A20. Again, the generated data and the original data behaves similarly, even though some smaller differences can be noted. As for the Markov chain model data, the estimated transition probabilities are not as accurate for the transition of the LOB when the order imbalance is either very high or very low. The difference is particularly clear for the regimes of low order imbalance in this case. It is the case here as well that it is not clear if the generative RNN model is not as accurate for these cases or if there are just too few samples in these regions to get good estimates of the transition probabilities.

Price signatures for MOs are shown in figure A21. The original data and the generated data exhibit similar behaviour with a price change following the MOs, even though the original data has slightly larger changes for both ask and bid MOs.

The KS statistics are compared in table A3. All values are reasonable for the RNN model, even though they are larger than for the Markov chain data. This reflects that it is a harder task to generate data according to the Nasdaq data than the Markov chain data. The numbers suggest possibilities to improve the modelling of order size in particular. Similarly to the Markov chain model data, both the RNN model and the naive model perform well for the distributions of event type, price level, order size and time delay, while for order imbalance and spread size the RNN model significantly outperforms the naive model.

The results on the prediction problem are shown in table A4. The DeepLOB model and the generative RNN model achieve comparable overall performance, but the generative RNN model performs slightly better on macro F1-score and macro precision.

7. Conclusion

We have proposed a generative model for LOBs based on RNNs. A transition of the LOB is modelled by an event type, a price level, an order size and a time delay. By decomposing the joint transition probability as a product of conditional probabilities, RNNs are trained to capture the complete dynamics of the LOB.

We also propose a number of evaluation metrics related to trading execution, including mid-price prediction, order imbalance and price signatures when executing MOs.

It is demonstrated that the generative RNN model is able to replicate the dynamics of a synthetic Markov chain LOB model as well as the characteristics of real order book data from Nasdaq Stockholm. However, the results show that there is potential to find hyperparameters and distributions more suitable to the data, which we have not had the possibility to test thoroughly. It would also be interesting to see the results on a broader selection of financial instruments.

Furthermore, other possible future directions are to extend the model to account for extensions of the LOB, such as market fragmentation, hidden liquidity and auctions.

Disclosure statement

No potential conflict of interest was reported by the author(s).

Funding

This work was partially supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation.

Data availability statement

The data that supports the findings of this study is available from the Swedish House of Finance Research Data Center. Restrictions apply to the availability of the data, which was used under license for this study. Data is available at https://www.data.houseoffinance.se/ with the permission of the Swedish House of Finance.

ORCID

Hanna Hultin http://orcid.org/0000-0002-0067-4908
Henrik Hult http://orcid.org/0000-0001-9210-121X

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G.S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y. and Zheng, X., TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. 2015. Software available from https://www.tensorflow.org,

Alfonsi, A., Fruth, A. and Schied, A., Optimal execution strategies in limit order books with general shape functions. *Quant. Finance*, 2010, **10**, 143–157.

Almgren, R. and Chriss, N., Optimal execution of portfolio transactions. *J. Risk*, 2001, **3**, 5–39.

Ba, J.L., Kiros, J.R. and Hinton, G.E., Layer normalization. 2016. arXiv preprint arXiv:1607.06450.

Bertsimas, D. and Lo, A.W., Optimal control of execution costs. *J. Financial Mark.*, 1998, **1**, 1–50.

Brent, R.P., *Algorithms for Minimization Without Derivatives*, 1973 (Prentice-Hall: Englewood Cliffs, NJ).

Buehler, H., Gonon, L., Teichmann, J. and Wood, B., Deep hedging. *Quant. Finance*, 2019, **19**, 1271–1291.

Buehler, H., Horvath, B., Lyons, T., Perez Arribas, I. and Wood, B., A data-driven market simulator for small data environments. 2020. Available at SSRN 3632431.

Byrd, D., Hybinette, M. and Balch, T.H., ABIDES: Towards high-fidelity multi-agent market simulation. In *Proceedings of the 2020 ACM SIGSIM Conference on Principles of Advanced Discrete Simulation*, pp. 11–22, 2020 (Association for Computing Machinery: New York, NY).

Cartea, Á., Jaimungal, S. and Penalva, J., *Algorithmic and High-Frequency Trading*, 2015 (Cambridge University Press: Cambridge, UK).

Cont, R. and De Larrard, A., Price dynamics in a Markovian limit order market. *SIAM J. Financ. Math.*, 2013, **4**, 1–25.

Cont, R., Stoikov, S. and Talreja, R., A stochastic model for order book dynamics. *Oper. Res.*, 2010, **58**, 549–563.

Dixon, M., Sequence classification of the limit order book using recurrent neural networks. *J. Comput. Sci.*, 2018, **24**, 277–286.

Doering, J., Fairbank, M. and Markose, S., Convolutional neural networks applied to high-frequency market microstructure forecasting. In *2017 9th Computer Science and Electronic Engineering Conference, CEEC 2017 – Proceedings*, pp. 31–36, 2017 (IEEE: Colchester).

Gers, F.A., Schmidhuber, J. and Cummins, F., Learning to forget: Continual prediction with LSTM. *Neural Comput.*, 2000, **12**, 2451–2471. https://doi.org/10.1162/089976600300015015.

Gould, M.D., Porter, M.A., Williams, S., McDonald, M., Fenn, D.J. and Howison, S.D., Limit order books. *Quant. Finance*, 2013, **13**, 1709–1742.

Graves, A., Generating sequences with recurrent neural networks. arXiv preprint arXiv:1308.0850, 2013.

Graves, A., Mohamed, A.R. and Hinton, G., Speech recognition with deep recurrent neural networks. In *Proceedings of the 2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 6645–6649, 2013 (IEEE: Vancouver, BC).

Hochreiter, S. and Schmidhuber, J., Long short-term memory. *Neural Comput.*, 1997, **9**, 1735–1780.

Horvath, B., Muguruza, A. and Tomas, M., Deep learning volatility: A deep neural network perspective on pricing and calibration in (rough) volatility models. *Quant. Finance*, 2021, **21**, 11–27.

Huang, W., Lehalle, C.A. and Rosenbaum, M., Simulating and analyzing order book data: The queue-reactive model. *J. Am. Stat. Assoc.*, 2015, **110**, 107–122.

Huang, W. and Rosenbaum, M., Ergodicity and diffusivity of Markovian order book models: A general framework. *SIAM J. Financ. Math.*, 2017, **8**, 874–900.

Hult, H. and Kiessling, J., Algorithmic trading with Markov chains, 2010. Preprint. Available at: https://www.researchgate.net/publication/268032734_ALGORITHMIC_TRADING_WITH_MARKOV_CHAINS.

Kercheval, A.N. and Zhang, Y., Modelling high-frequency limit order book dynamics with support vector machines. *Quant. Finance*, 2015, **15**, 1315–1329.

Kingma, D.P. and Ba, J.L., Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015 – Conference Track Proceedings*, pp. 1–15, 2015 (ICLR: San Diego, CA).

Kondratyev, A. and Schwarz, C., The market generator. 2019. Available at SSRN 3384948.

Kondratyev, A., Schwarz, C. and Horvath, B., Data anonymisation, outlier detection and fighting overfitting with restricted Boltzmann machines. *SSRN Electron. J.*, 2020, **2020**.

Massey Jr, F.J., The Kolmogorov–Smirnov test for goodness of fit. *J. Am. Stat. Assoc.*, 1951, **46**, 68–78.

Muni Toke, I. and Yoshida, N., Modelling intensities of order flows in a limit order book. *Quant. Finance*, 2017, **17**, 683–701.

Ntakaris, A., Magris, M., Kanniainen, J., Gabbouj, M. and Iosifidis, A., Benchmark dataset for mid-price forecasting of limit order book data with machine learning methods. *J. Forecast.*, 2018, **37**, 852–866.

Obizhaeva, A.A. and Wang, J., Optimal trading strategy and supply/demand dynamics. *J. Finan. Mark.*, 2013, **16**, 1–32.

Oomen, R., Price signatures. *Quant. Finance*, 2019, **19**, 733–761.

Parlour, C.A. and Seppi, D.J., Limit order markets: A survey. In *Handbook of Financial Intermediation and Banking*, Vol. 5, pp. 63–95, 2008 (Elsevier: Amsterdam).

Passalis, N., Tsantekidis, A., Tefas, A., Kanniainen, J., Gabbouj, M. and Iosifidis, A., Time-series classification using neural bag-of-features. In *Proceedings of the 2017 25th European Signal Processing Conference (EUSIPCO)*, pp. 301–305, 2017 (IEEE: Kos, Greece).

Sadhwani, A., Giesecke, K. and Sirignano, J., Deep learning for mortgage risk. *J Financ. Econ.*, 2021, **19**, 313–368.

Sirignano, J.A., Deep learning for limit order books. *Quant. Finance*, 2019, **19**, 549–570.

Sutskever, I., Vinyals, O. and Le, Q.V., Sequence to sequence learning with neural networks. In *Proceedings of the Advances in Neural Information Processing Systems*, pp. 3104–3112, 2014 (Curran Associates, Inc.: Montréal).

Tsantekidis, A., Passalis, N., Tefas, A., Kanniainen, J., Gabbouj, M. and Iosifidis, A., Forecasting stock prices from the limit order book using convolutional neural networks. In *Proceedings of the 2017 IEEE 19th Conference on Business Informatics (CBI)*, Vol. 1, pp. 7–12, 2017.

Tsantekidis, A., Passalis, N., Tefas, A., Kanniainen, J., Gabbouj, M. and Iosifidis, A., Using deep learning to detect price change indications in financial markets. In *Proceedings of the 2017 25th European Signal Processing Conference (EUSIPCO)*, pp. 2511–2515, 2017 (IEEE: Kos, Greece).

Vyetrenko, S., Byrd, D., Petosa, N., Mahfouz, M., Dervovic, D., Veloso, M. and Balch, T., Get real: Realism metrics for robust limit order book market simulations. In *Proceedings of the First ACM International Conference on AI in Finance*, New York, NY, pp. 1–8, 2020 (Association for Computing Machinery: New York, NY).

Wiese, M., Bai, L., Wood, B. and Buehler, H., Deep hedging: Learning to simulate equity option markets. In *NeurIPS 2019 Workshop on Robust AI in Financial Services: Data, Fairness, Explainability, Trustworthiness, and Privacy*, Vancouver, 2019.

Wiese, M., Knobloch, R., Korn, R. and Kretschmer, P., Quant GANs: Deep generation of financial time series. *Quant. Finance*, 2020, **20**, 1419–1440.

Zhang, Z., Zohren, S. and Roberts, S., DeepLOB: Deep convolutional neural networks for limit order books. *IEEE Trans. Signal. Process.*, 2019, **67**, 3001–3012.

Appendices

Appendix 1. Results for the Markov chain data

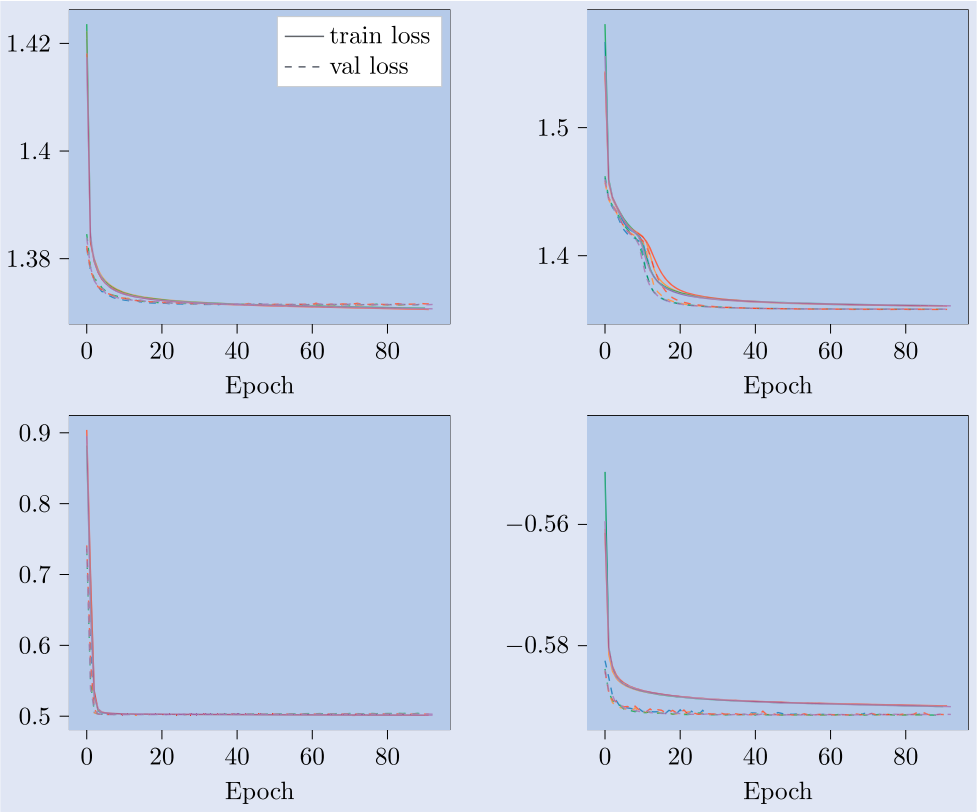

Figure A1. Loss function during training on Markov chain model data. Upper left is the loss for event type, upper right loss for price level, lower left loss for order size and lower right loss for time delay. Each run is represented by its own colour.

Figure A2. Frequencies of event types for original Markov chain model data and generated data.

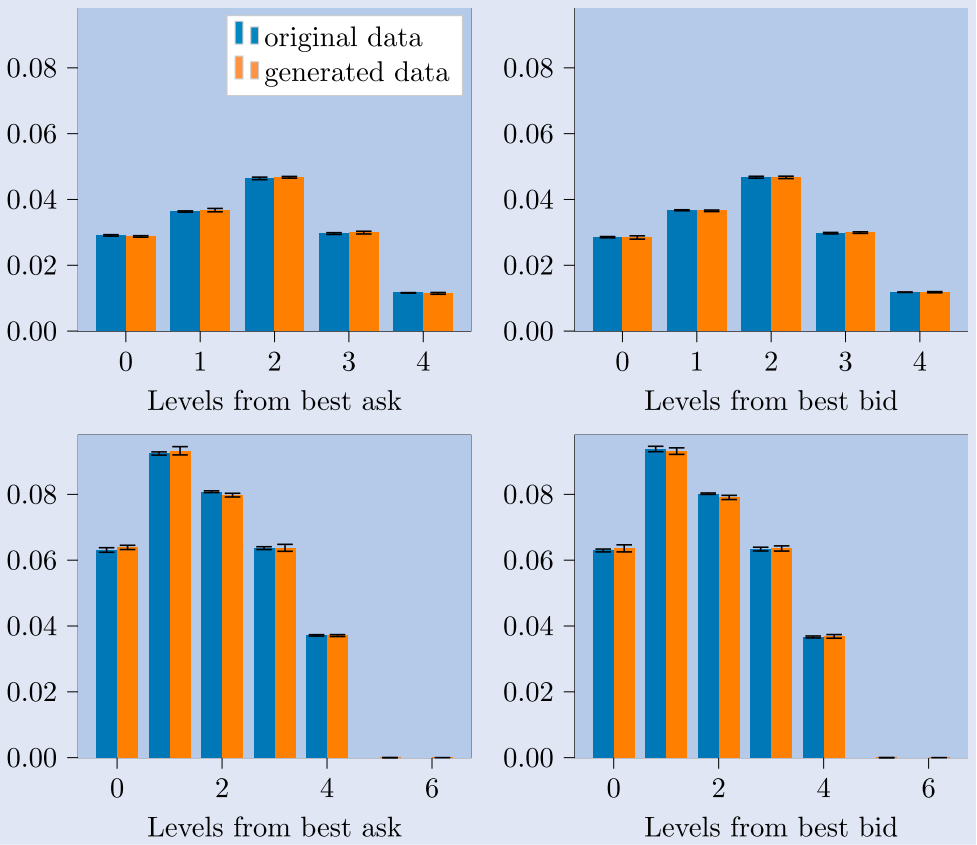

Figure A3. Frequencies of event types and levels for original Markov chain model data and generated data. Upper left is the frequencies for buy limit orders, upper right frequencies for sell limit orders, lower left frequencies for cancellations of buy orders and lower right frequencies for cancellations of sell orders.

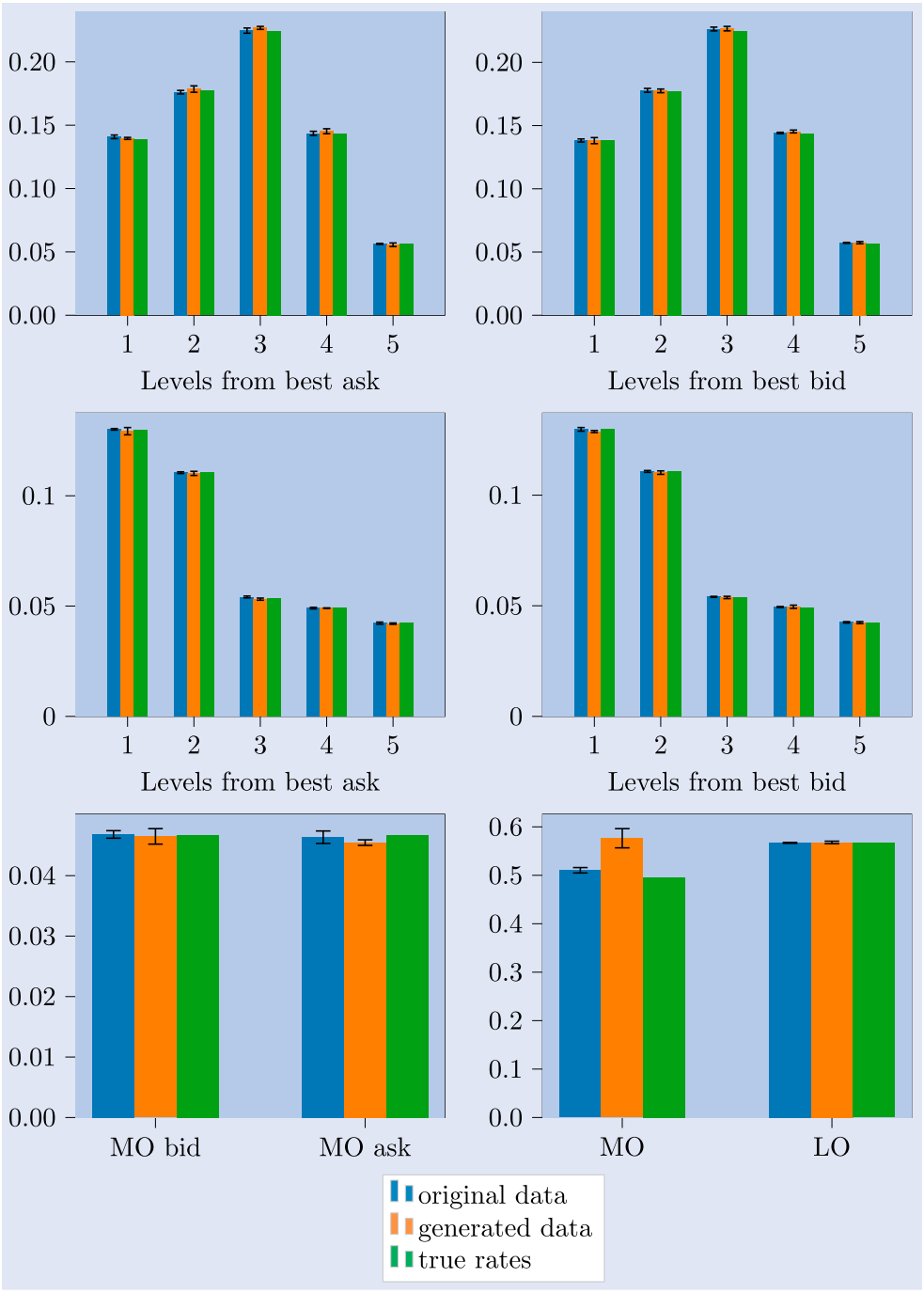

Figure A4. Rates of event types for original Markov chain model data and generated data together with true rates. Upper left is the rates for buy limit orders, upper right rates for sell limit orders, middle left rates for cancellations of buy orders and middle right rates for cancellations of sell orders. Lower left are the rates for bid and ask MOs and lower right the size parameters for MOs and LOs.

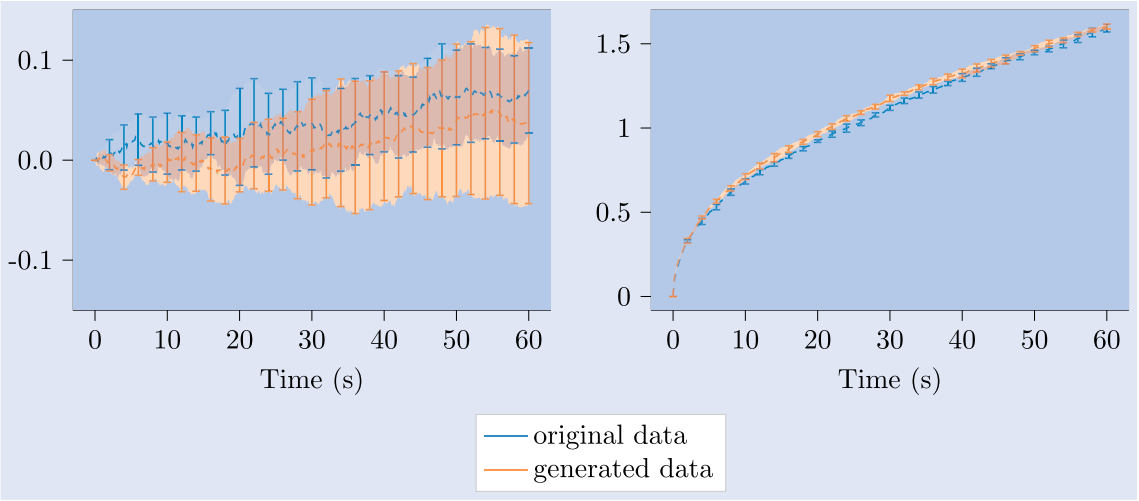

Figure A5. Mid-price over time for original Markov chain model data and generated data. Left for average and right for standard deviation.

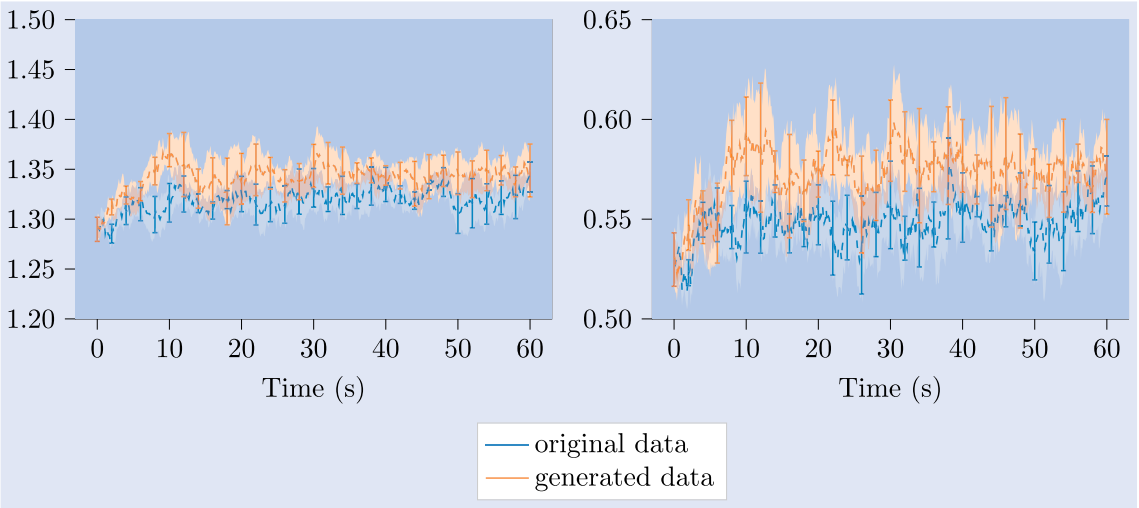

Figure A6. Spread over events for original Markov chain model data and generated data. Left for average and right for standard deviation.

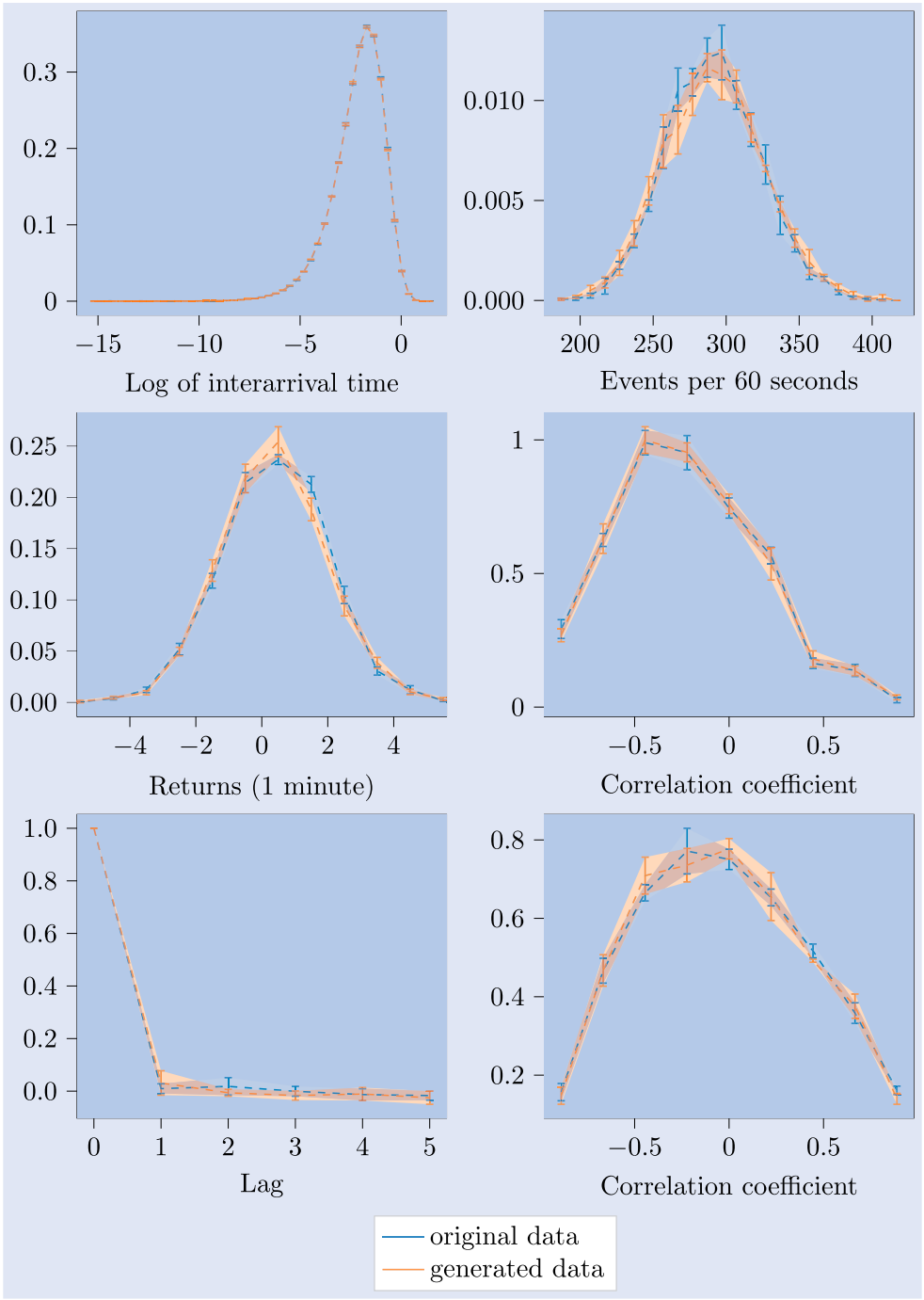

Figure A7. Visualizations of stylized facts for original Markov chain model data and generated data. Upper left is normalized histogram of logarithm of interarrival times, upper right normalized histogram of number of events per 60 seconds, middle left normalized histogram of 1 minute returns, middle right normalized histogram of autocorrelation of 10 seconds return with lag of one over each 1 minute sequence, bottom left autocorrelation of square returns as a function of lag and bottom right normalized histogram of volume/volatility correlation.

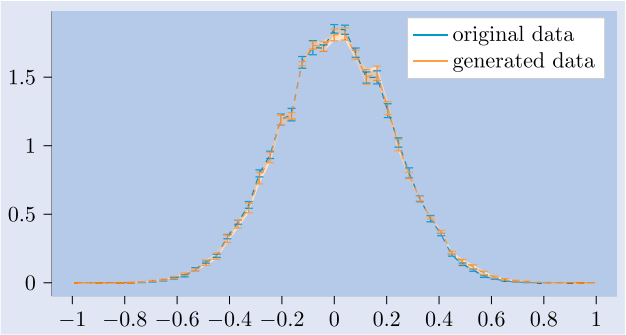

Figure A8. Normalized histogram of order imbalance as defined in Section 5.4 for original Markov chain model data and generated data.

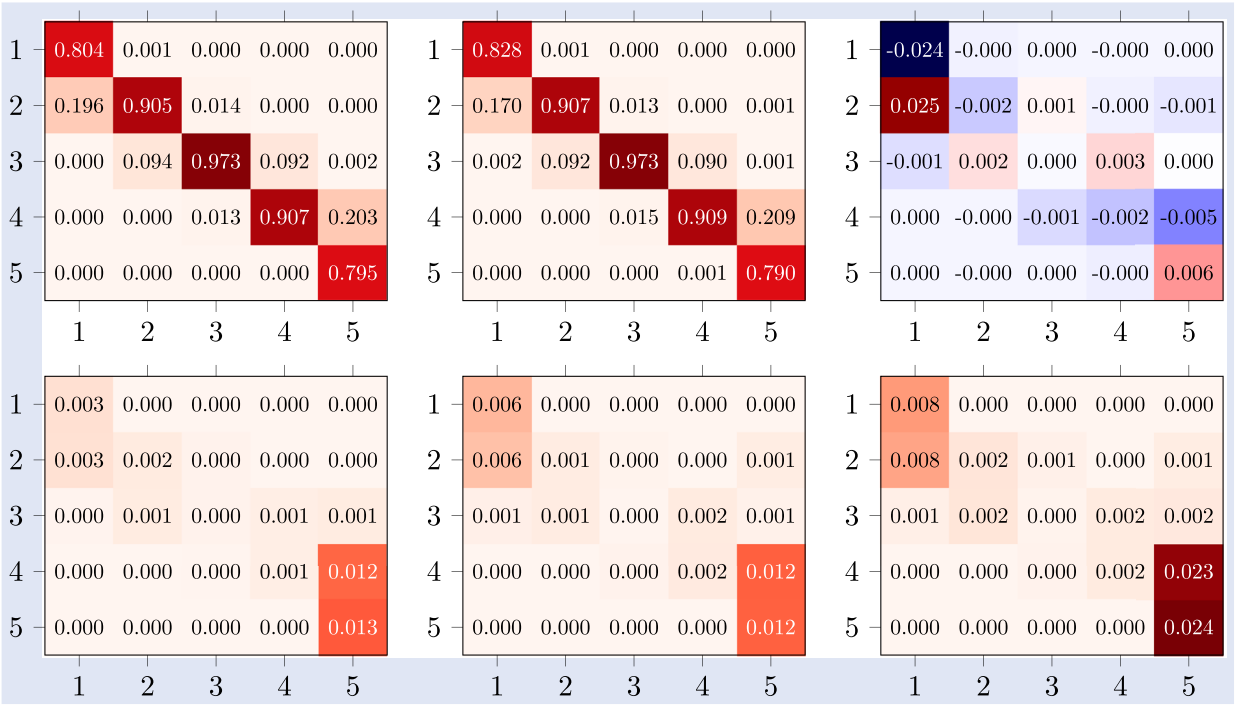

Figure A9. Transitions between order imbalance regimes as described in Section 5.4 for original Markov chain model data and generated data. Left for original data, middle for generated data and right the difference between these, with upper figures for the average of the five runs and bottom for standard deviations over these five runs.

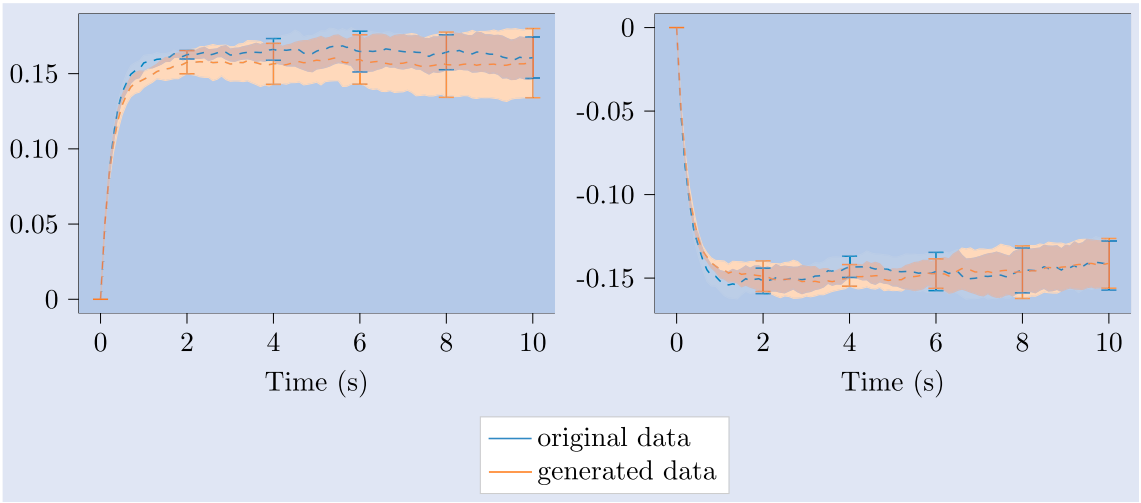

Figure A10. Mid-price impact of market orders as defined in Section 5.5, with ask MO to the left and bid MO to the right, for original Markov chain model data and generated data.

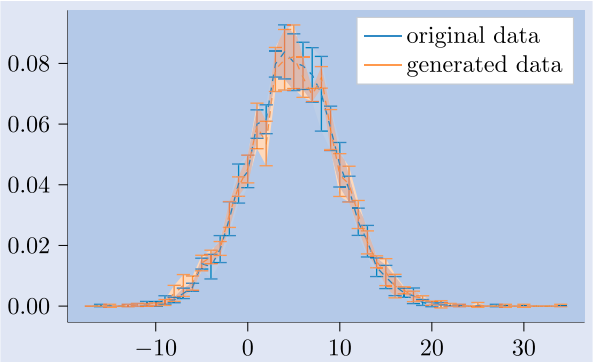

Figure A11. Normalized histogram of execution cost when executing five market orders as described in Section 5.5 for original Markov chain model and the generative RNN model.

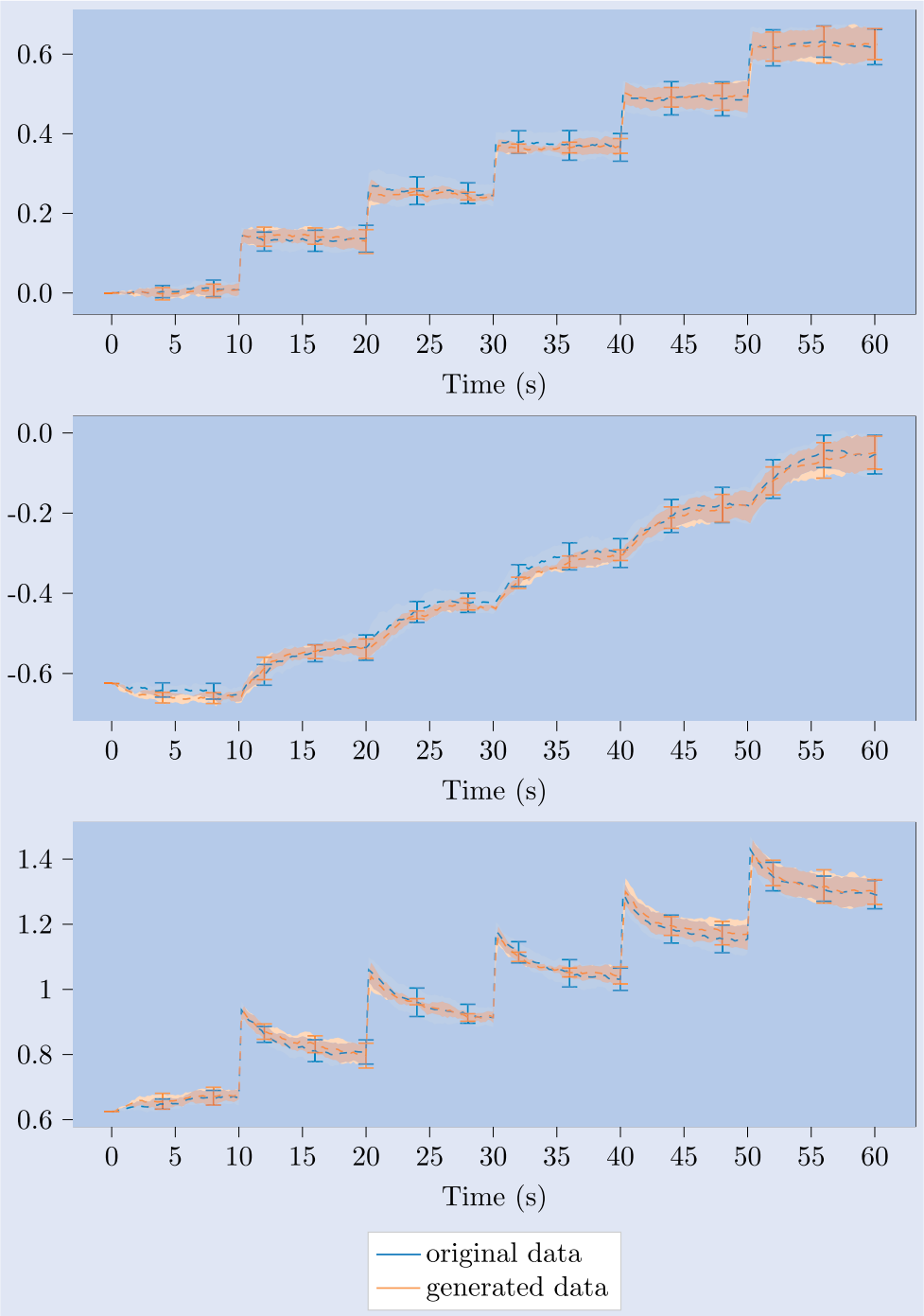

Figure A12. Price impact when executing five market orders as described in Section 5.5 for original Markov chain model and generative RNN model. Top for mid, middle for bid and bottom for ask.

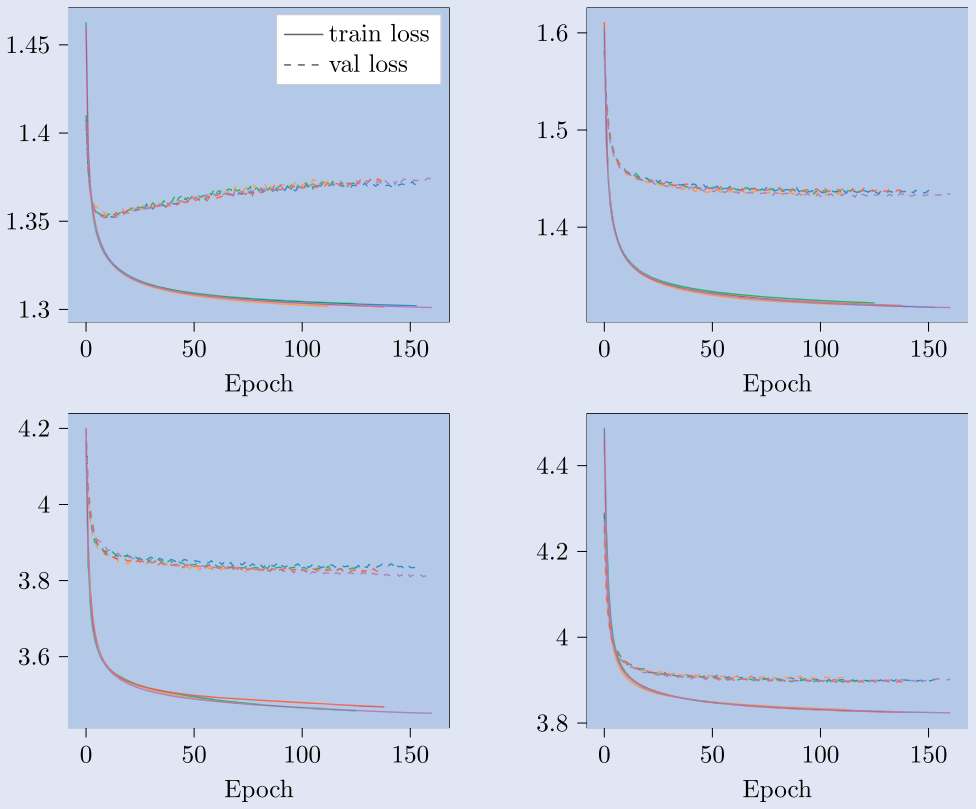

Figure A13. Loss function during training on Nasdaq data. Upper left is the loss for event type, upper right loss for price level, lower left loss for order size and lower right loss for time delay. Each run is represented by its own colour.

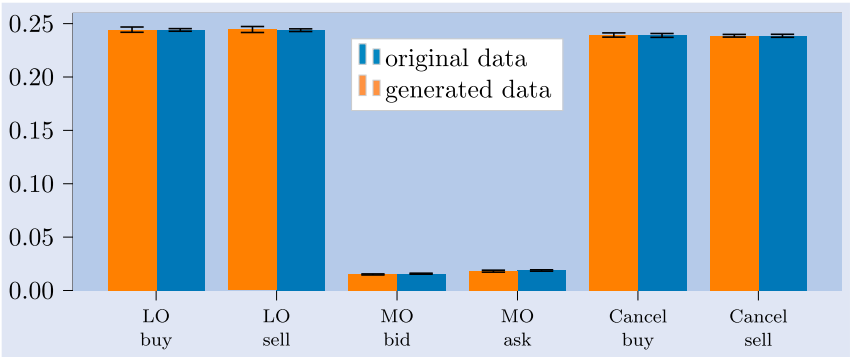

Figure A14. Frequencies of event types for original Nasdaq data and generated data.

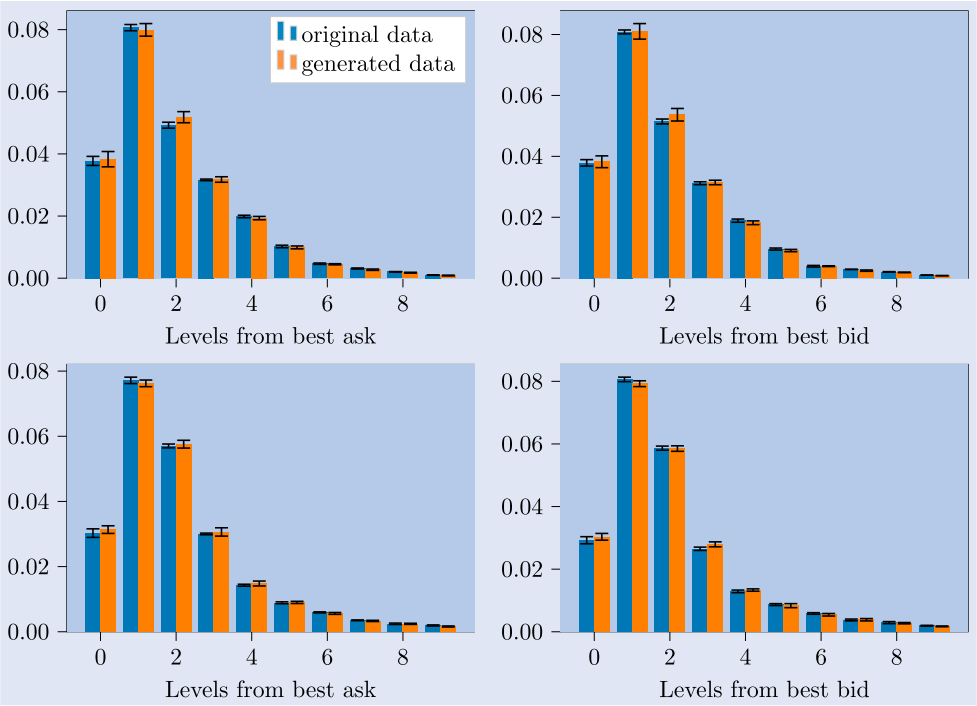

Figure A15. Detailed frequencies of event types for original Nasdaq data and generated data. Upper left is the frequencies for buy limit orders, upper right frequencies for sell limit orders, lower left frequencies for cancellations of buy orders and lower right frequencies for cancellations of sell orders.

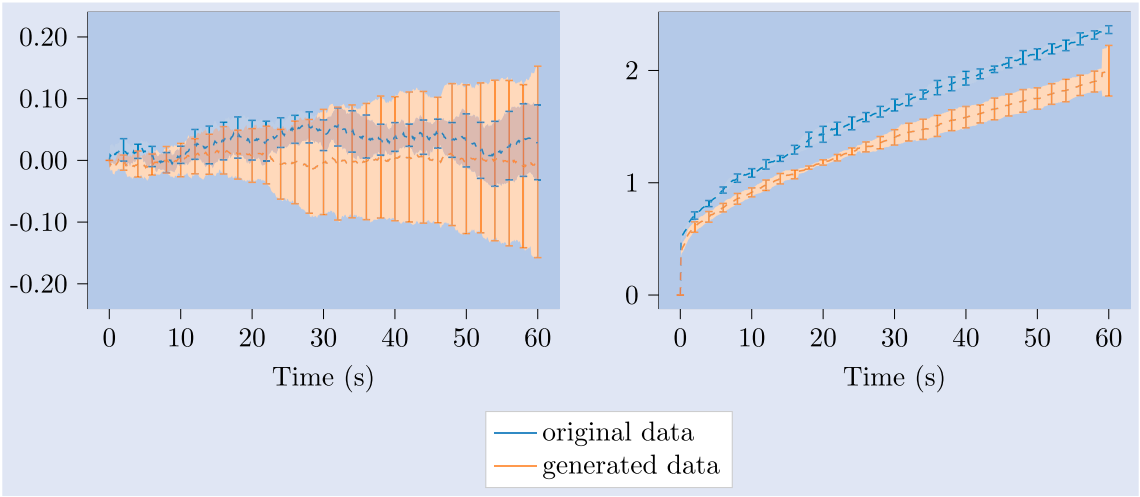

Figure A16. Mid-price over time for original Nasdaq data and generated data. Left for average and right for standard deviation.

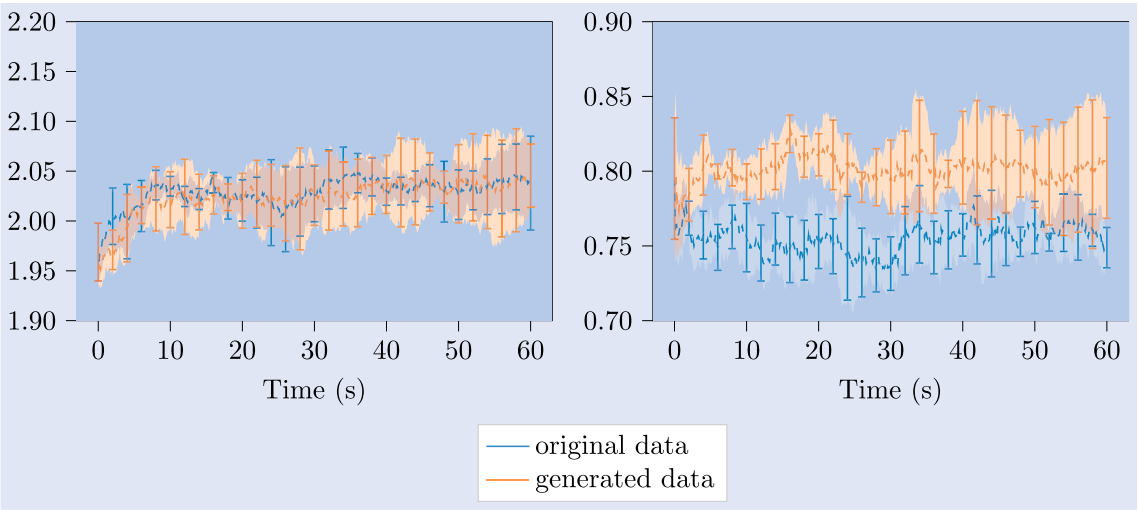

Figure A17. Spread over time for original Nasdaq data and generated data. Left for average and right for standard deviation.

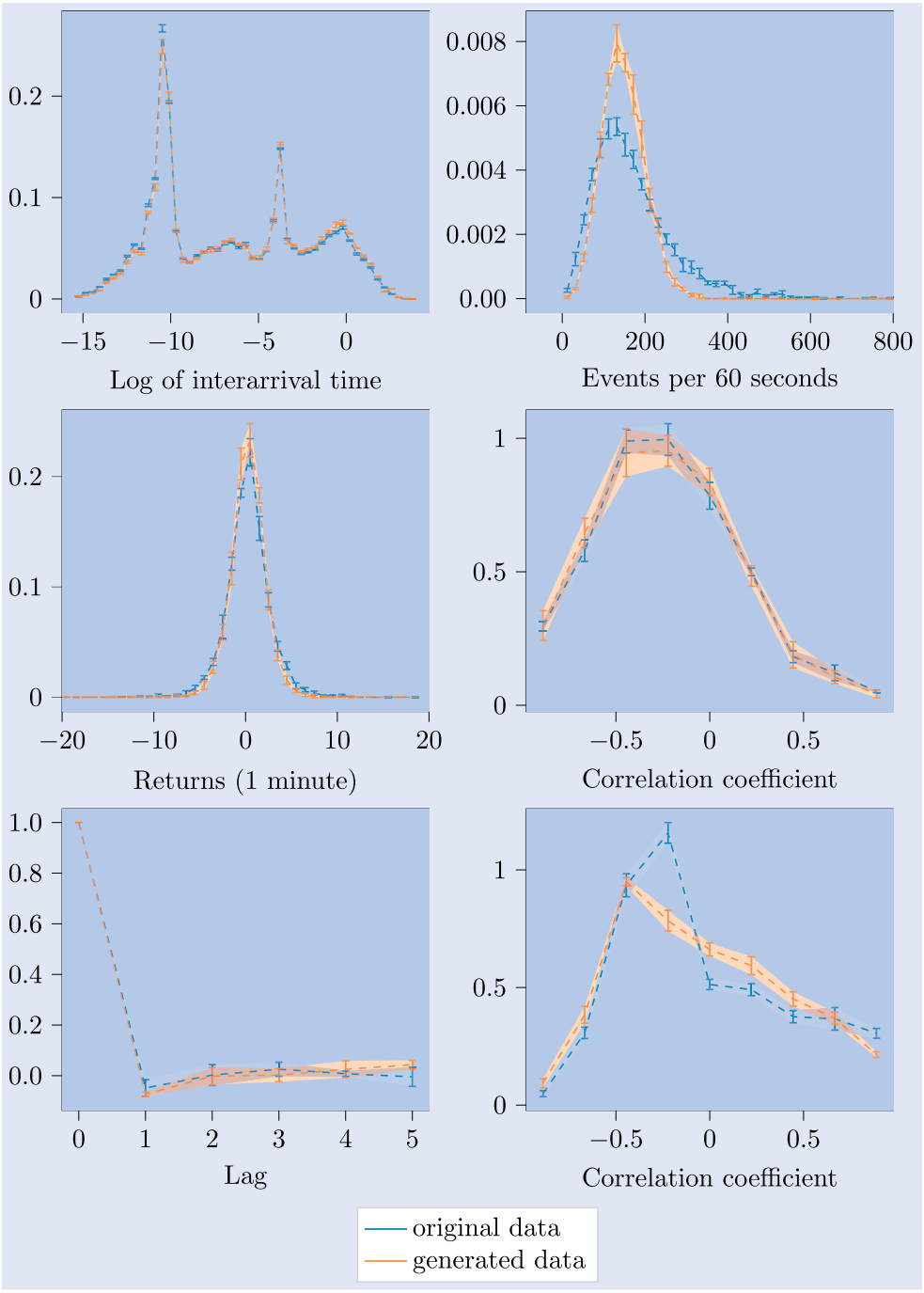

Figure A18. Visualizations of stylized facts introduced in Section 5.3 for original Nasdaq model data and generated data. Upper left is normalized histogram of logarithm of interarrival times, upper right normalized histogram of number of events per 60 seconds, middle left normalized histogram of 1 minute returns, middle right normalized histogram of autocorrelation of 10 seconds return with lag of one over each 1 minute sequence, bottom left autocorrelation of square returns as a function of lag and bottom right normalized histogram of volume/volatility correlation.

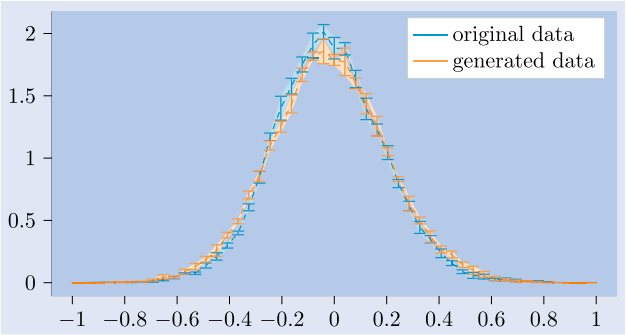

Figure A19. Normalized histogram of order imbalance as defined in Section 5.4 for original Nasdaq data and generated data.

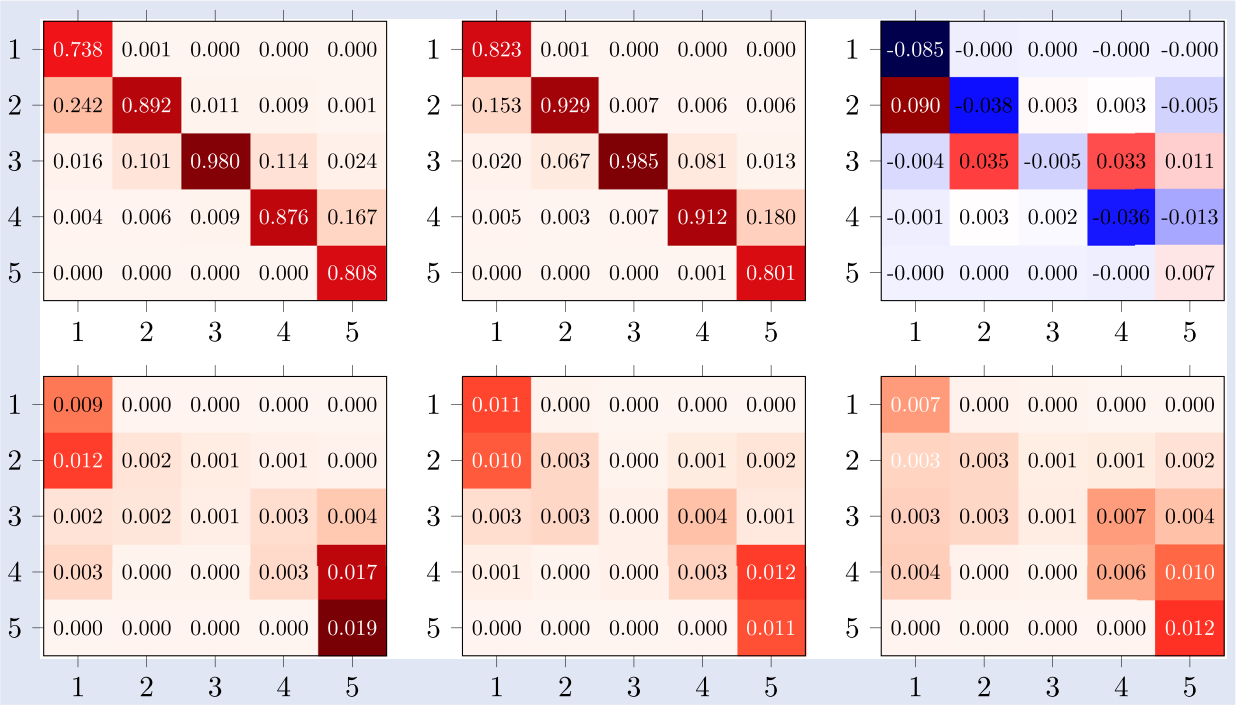

Figure A20. Transitions between order imbalance regimes as described in Section 5.4 for original Nasdaq data and generated data. Left for original data, middle for generated data and right the difference between these, with upper figures for the average of the five runs and bottom for standard deviations over these five runs.

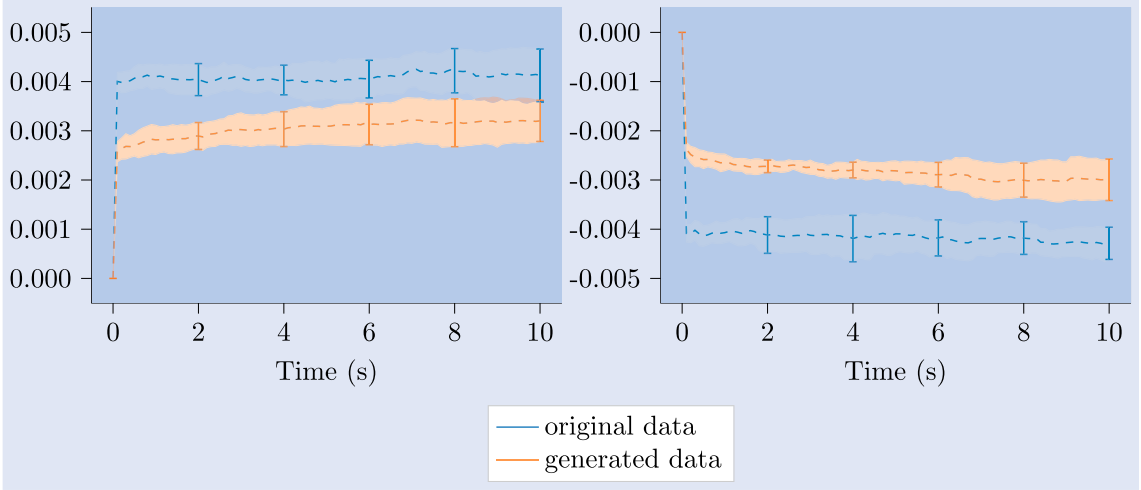

Figure A21. Mid-price impact of market orders as defined in Section 5.5, with ask MO to the left and bid MO to the right, for original Nasdaq data and generated data.

Appendix 2. Results on the Nasdaq data

Table A1. KS statistics as explained in Section 5.6 for original data from Markov chain model and the corresponding data generated by the RNN model and the naive model respectively.

Distribution	RNN model	Naive model
Event type	**0.00127 ± 0.000925**	0.0079 ± 0.00113
Price level	**0.00296 ± 0.000439**	0.0372 ± 0.00262
Order size	**0.00131 ± 0.00042**	0.00217 ± 0.000273
Time delay	**0.00246 ± 0.00121**	0.0089 ± 0.00111
Order imbalance	**0.018 ± 0.0115**	0.123 ± 0.00547
Spread size	**0.0158 ± 0.00829**	0.112 ± 0.00427
Mid-price returns	**0.000573 ± 0.000182**	0.0115 ± 0.00024
Mid-price volatility	**0.0269 ± 0.00487**	0.294 ± 0.00373

Note: The mean and standard deviation over five runs is presented for each statistic, and the lower average scores per distribution type are printed in bold. Note that the RNN model is overall more accurate than the naive model and in particular the RNN model outperforms the naive model significantly for order imbalance, spread size and mid-price volatility.

Table A2. Metrics of mid-price prediction experiment as described in Section 5.7 for Markov chain model data.

Score	RNN model	DeepLOB
Accuracy	0.616 ± 0.00211	**0.653 ± 0.00629**
Macro F1	0.619 ± 0.00207	**0.653 ± 0.00651**
Macro Precision	0.623 ± 0.00299	**0.660 ± 0.00586**
Macro Recall	0.616 ± 0.00182	**0.652 ± 0.00646**

Note: The mean and standard deviation over five runs is presented for each metric, and the higher average scores per metric are printed in bold.

Table A3. KS statistics as explained in Section 5.6 for original Nasdaq data and the corresponding data generated by the RNN model and the naive model respectively.

Distribution	RNN model	Naive model
Event type	**0.00263 ± 0.0014**	0.00307 ± 0.00117
Price level	**0.0132 ± 0.0024**	0.0162 ± 0.00669
Categorical order size	0.0148 ± 0.00234	**0.00416 ± 0.00147**
Absolute order size	**0.0383 ± 0.00355**	0.143 ± 0.00262
Categorical time delay	0.0284 ± 0.000777	**0.00501 ± 0.00252**
Absolute time delay	0.0288 ± 0.000589	**0.00559 ± 0.00288**
Order imbalance	**0.0287 ± 0.00654**	0.122 ± 0.00388
Spread size	**0.0352 ± 0.00642**	0.479 ± 0.0119
Mid-price returns	**0.00113 ± 0.000334**	0.00197 ± 0.000284
Mid-price volatility	**0.0793 ± 0.0123**	0.083 ± 0.00646

Note: The mean and standard deviation over five runs is presented for each statistic, and the lower average scores per distribution type are printed in bold. The RNN model has overall reasonable performance and outperforms the naive model significantly for absolute order size, order imbalance and spread size.

Table A4. Metrics of mid-price prediction experiment as described in Section 5.7 for Nasdaq data.

Score	RNN model	DeepLOB
Accuracy	**0.637 ± 0.00746**	0.631 ± 0.00494
Macro F1	**0.644 ± 0.00719**	0.632 ± 0.00553
Macro Precision	**0.660 ± 0.00817**	0.634 ± 0.00703
Macro Recall	**0.637 ± 0.00735**	0.631 ± 0.00457

Note: The mean and standard deviation over five runs is presented for each metric, and the higher average scores per metric are printed in bold.