# DiffVolume: Diffusion Models for Volume Generation in Limit Order Books

Zhuohan Wang
King's College London
London, United Kingdom
zhuohan.wang@kcl.ac.uk

Carmine Ventre
King's College London
London, United Kingdom
carmine.ventre@kcl.ac.uk

## Abstract

Modeling limit order books (LOBs) dynamics is a fundamental problem in market microstructure research. In particular, generating high-dimensional volume snapshots with strong temporal and liquidity-dependent patterns remains a challenging task, despite recent work exploring the application of Generative Adversarial Networks to LOBs. In this work, we propose a conditional **Diff**usion model for the generation of future LOB **Volume** snapshots (**DiffVolume**). We evaluate our model across three axes: (1) *Realism*, where we show that DiffVolume, conditioned on past volume history and time of day, better reproduces statistical properties such as marginal distribution, spatial correlation, and autocorrelation decay; (2) *Counterfactual generation*, allowing for controllable generation under hypothetical liquidity scenarios by additionally conditioning on a target future liquidity profile; and (3) *Downstream prediction*, where we show that the synthetic counterfactual data from our model improves the performance of future liquidity forecasting models. Together, these results suggest that DiffVolume provides a powerful and flexible framework for realistic and controllable LOB volume generation.

## CCS Concepts

• **Computing methodologies** → **Machine learning**.

## Keywords

Limit order books, Volume Generation, Diffusion models

**ACM Reference Format:**
Zhuohan Wang and Carmine Ventre. 2025. DiffVolume: Diffusion Models for Volume Generation in Limit Order Books. In *6th ACM International Conference on AI in Finance (ICAIF '25), November 15–18, 2025, Singapore, Singapore.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3768292.3770413

## 1 Introduction

Limit order books (LOBs) are the backbone of modern financial markets, capturing real-time information on pending buy and sell orders at various price levels [14]. Among the core components of LOBs, order volume at different price levels serves as a crucial indicator of market liquidity, price discovery, and trading opportunities. Accurate modeling of LOB volumes enables practitioners and researchers to improve trading strategies, optimize market making, and conduct robust market impact analyses [6]. Therefore, the task of volume generation is fundamental for both theoretical insights and practical financial applications.

Most prior research focuses primarily on event-level generation, with the objective of simultaneously generating price and volume information, which often leads to less accurate and realistic results [10, 22, 24]. In particular, recent work by Cont et al. [11] applies Generative Adversarial Networks (GANs) to LOB modeling and considers six price levels of a single big tick stock. GAN-based models are known to be prone to training instability and mode collapse, which can significantly impair their ability to replicate complex volume patterns across time and depth. Therefore, there remains a clear need for advanced modeling techniques that address these limitations in a comprehensive way.

To address these limitations, we introduce **DiffVolume**, a diffusion model-based framework specifically designed for the generation of future LOB volume snapshots. The proposed framework offers three main contributions:

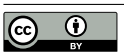
(1) We develop a novel diffusion model architecture conditioning on past volume trajectories and time of day, which captures the spatial correlation structures and intricate temporal dependencies that are present in LOB volume data;
(2) By additionally conditioning on a target future liquidity profile, we implement counterfactual generation under hypothetical liquidity scenarios;
(3) We demonstrate the quality and practical utility of the generated data by (i) evaluating the distribution of its data points vis-a-vis the original dataset; and (ii) showing how a downstream liquidity prediction task can be solved more effectively.

## 2 Related Work

We will discuss related research in three parts.

**Limit Order Book Dynamics.** The dynamics of LOBs have been extensively studied through various modeling approaches. Poisson processes models treat order arrivals and cancellations as independent events, characterized by a constant or time-varying intensity, providing a simplified representation of order flows [5]. Hawkes processes models extend this framework by incorporating self-exciting properties, effectively modeling the clustering of order events, and capturing the feedback loops inherent in market activity [3, 12]. Agent-based models simulate the interactions of heterogeneous traders with distinct strategies, allowing a detailed analysis of market microstructure phenomena and emergent behaviors such as volatility clustering and market crashes [1, 4, 8, 26].

**Generative Modeling of Limit Order Books.** Generative modeling of LOBs can be primarily classified into two categories: autoregressive models and GANs. Autoregressive models generate




*ICAIF '25, Singapore, Singapore*

ACM ISBN 979-8-4007-2220-2/25/11
https://doi.org/10.1145/3768292.3770413

LOB data sequentially, leveraging previous historical data points to predict future order book states. Hultin et al. [18] propose a model based on recurrent neural networks (RNNs) that decomposes the joint distribution of LOB transitions into conditional probabilities over order type, price, size, and time delay, each modeled with a dedicated RNN. Similarly, Nagy et al. [24] present an end-to-end autoregressive model using structured state-space models, which tokenizes message streams and generates realistic LOB transitions. GAN-based models, on the other hand, employ adversarial training to generate realistic synthetic limit order books. Li et al. [22] introduce Stock-GAN, a conditional Wasserstein GAN designed to generate realistic stock market order streams by capturing historical dependence and mimicking auction mechanisms. Coletta et al. [10] propose a Conditional GAN framework that reacts to current market states and allows agent interaction within a simulation, showing enhanced realism and responsiveness. Meanwhile, Cont et al. [11] develop a GAN framework that arguably learns the conditional distribution of future LOB volume states, capturing both stylized facts and some market impact patterns. However, their model can only be applied on large tick stock and limited price levels that are often asymmetric between the two sides of the book. DiffVolume accurately models the volume dynamics across many price levels for both bids and asks, no matter the tick size.

**Diffusion Models Theory.** The foundational idea of diffusion models is inspired by thermodynamic diffusion [28], which proposes to train generative models by adding Gaussian noise over multiple steps and learning to reverse the process. Ho et al. [15] are the first to propose the Denoising Diffusion Probabilistic Model. Their method involves adding noise in a forward process and denoising in a reverse process to generate high-quality images. Subsequent work focuses on improving sampling quality and training efficiency [25, 29]. Diffusion models are also closely related to score-based generative modeling, where the model learns the gradient of the data density through denoising score matching [30, 35]. This connection has been formally unified under the stochastic differential equation framework by [31]. Recent advances further clarify and optimize the design space of diffusion models under this unified view. For example, Karras et al. [19] analyze the score matching and noise schedules in depth, leading to improvements in sample quality and generation speed. This line of work provides the theoretical foundation for applying diffusion models to structured data domains such as financial data [21, 36].

## 3 Methodology

In this section, we will briefly summarize conditional denoising diffusion probabilistic models from the score-based perspective and describe the proposed DiffVolume architecture in detail.

### 3.1 Generative Diffusion Models

**Denoising Diffusion Probabilistic Models.** Denoising Diffusion Probabilistic Models (DDPMs) [9, 15, 23, 28] consist of two complementary stochastic processes, a forward (noising) process and a reverse (generative) process. In the forward process, Gaussian noise is progressively added to the data through a fixed Markov chain, transforming it into a nearly pure noise distribution. The reverse process learns to invert this transformation by iteratively denoising the corrupted data, thereby recovering samples from the data distribution via learned transitions.

Consider a noise strength sequence $0 < \beta_1 < \beta_2 < \cdots < \beta_N < 1$ and forward noising process $p(\mathbf{x}_i|\mathbf{x}_{i-1}) = \mathcal{N}(\mathbf{x}_i|\sqrt{1-\beta_i}\mathbf{x}_{i-1}, \beta_i\mathbf{I})$ where $\mathcal{N}(\mu, \nu)$ denotes a normal distribution with mean $\mu$ and variance $\nu$. With $\alpha_i := \prod_{j=1}^{i}(1-\beta_j)$, we have $p_{\alpha_i}(\mathbf{x}_i|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_i|\sqrt{\alpha_i}\mathbf{x}_0, (1-\alpha_i)\mathbf{I})$, and the perturbed data distribution is given by $p_{\alpha_i}(\tilde{\mathbf{x}}) = \int p_{\text{data}}(\mathbf{x})p_{\alpha_i}(\tilde{\mathbf{x}}|\mathbf{x})d\mathbf{x}$. The noise scale is predetermined to approximately satisfy $p_{\alpha_N}(\tilde{\mathbf{x}}) \sim \mathcal{N}(\tilde{\mathbf{x}}|\mathbf{0}, \mathbf{I})$, i.e., the terminal distribution is nearly independent of $p_{\text{data}}$. The reverse denoising process can be written as $p_\theta(\mathbf{x}_{i-1}|\mathbf{x}_i) = \mathcal{N}(\mathbf{x}_{i-1}|\frac{1}{\sqrt{1-\beta_i}}(\mathbf{x}_i + \beta_i\mathbf{s}_\theta(\mathbf{x}_i, i)), \beta_i\mathbf{I})$, where $\mathbf{s}_\theta(\mathbf{x}_i, i)$ is called the *score* [30]. The training objective is a sum of local denoising score-matching objectives, i.e., finding $\theta^*$ that minimizes

$$\sum_{i=1}^{N}(1-\alpha_i)\mathbb{E}_{p_{\text{data}}(\mathbf{x})}\mathbb{E}_{p_{\alpha_i}(\tilde{\mathbf{x}}|\mathbf{x})}\left[||\mathbf{s}_\theta(\tilde{\mathbf{x}}, i) - \nabla_{\tilde{\mathbf{x}}}\log p_{\alpha_i}(\tilde{\mathbf{x}}|\mathbf{x})||_2^2\right]. \quad (1)$$

**DDPMs and Stochastic Differential Equations.** Song et al. [31] demonstrate that DDPMs can be understood from the perspective of stochastic differential equations (SDEs). Let $\{\mathbf{x}(t)\}_{t=0}^{T}$ be a stochastic diffusion process indexed by a continuous time variable $t \in [0, T]$, evolving from $\mathbf{x}(0) \sim p_0$, the true data distribution, to $\mathbf{x}(T) \sim p_T$, approximately the tractable prior distribution. Denote the probability density function of $\mathbf{x}(t)$ by $p_t(\mathbf{x})$ and the transition kernel from $\mathbf{x}(s)$ to $\mathbf{x}(t)$ by $p_{st}(\mathbf{x}(t)|\mathbf{x}(s))$, for $0 \le s < t \le T$. Then, we can use an SDE to represent such a forward diffusion process:

$$d\mathbf{x} = \mathbf{f}(\mathbf{x}, t)\ dt + g(t)\ d\mathbf{w}, \quad (2)$$

where $\mathbf{f}(\mathbf{x}, t)dt$ is referred to as the *drift* term, and $g(t)d\mathbf{w}$ is referred to as the *diffusion* term. Here, $\mathbf{w}$ is a standard Wiener process and $d\mathbf{w} \sim \mathcal{N}(0, dt\mathbf{I})$. The synthetic data generation process is the reverse process of Eq. (2), which is also an SDE [2]:

$$d\mathbf{x} = [\mathbf{f}(\mathbf{x}, t) - g^2(t)\nabla_{\mathbf{x}}\log p_t(\mathbf{x})]\ dt + g(t)\ d\bar{\mathbf{w}}, \quad (3)$$

where $\bar{\mathbf{w}}$ is a reverse-time Wiener process and $\nabla_{\mathbf{x}}\log p_t(\mathbf{x})$ is the score of the marginal distribution corresponding to each $t$. It starts from an initial noise sample $\mathbf{x}(T) \sim p_T$ and gradually denoises it step by step following Eq. (3). Theoretically, if $T \to \infty$, we obtain $\mathbf{x}(0) \sim p_0$. To estimate $\nabla_{\mathbf{x}}\log p_t(\mathbf{x})$, the score network $\mathbf{s}_\theta(\mathbf{x}, t)$ is trained using the objective function

$$\kappa(t)\mathbb{E}_t\mathbb{E}_{\mathbf{x}(0)}\mathbb{E}_{\mathbf{x}(t)|\mathbf{x}(0)}\left[||\mathbf{s}_\theta(\mathbf{x}(t), t) - \nabla_{\mathbf{x}(t)}\log p_{0t}(\mathbf{x}(t)|\mathbf{x}(0))||_2^2\right], \quad (4)$$

where $\kappa : [0, T] \to \mathbb{R}^+$ is a positive weight and $t \sim \mathcal{U}[0, T]$. Eq. (4) is a continuous generalization of Eq. (1). Typically, the continuous form of the DDPM forward process is chosen to be

$$d\mathbf{x} = -\frac{\beta(t)}{2}\mathbf{x}\ dt + \sqrt{\beta(t)}\ d\mathbf{w}, \quad (5)$$

i.e., $\mathbf{f}(\mathbf{x}, t) = -\frac{\beta(t)}{2}\mathbf{x}$ and $g(t) = \sqrt{\beta(t)}$. Substituting $\mathbf{f}(\mathbf{x}, t)$ and $g(t)$ in (3), we can get the backward process in SDE form for DDPM.

**Conditional DDPMs.** How do we inject the conditioning $\mathbf{c}$ into the training and sampling process? Here we follow the *classifier-free guidance* approach [16], combining the conditional and unconditional models as follows:

$$\nabla_{\mathbf{x}}\log\tilde{p}(\mathbf{x}|\mathbf{c}) = \omega\nabla_{\mathbf{x}}\log p(\mathbf{x}|\mathbf{c}) + (1-\omega)\nabla_{\mathbf{x}}\log p(\mathbf{x}), \quad (6)$$

where $\nabla_{\mathbf{x}} \log p(\mathbf{x}|\mathbf{c})$ represents the conditional and $\nabla_{\mathbf{x}} \log p(\mathbf{x})$ represents the unconditional score, corresponding to the conditional and unconditional model distributions. Eq. (6) reduces to the unconditional score when $\omega = 0$, or recovers the conditional score when $\omega = 1$.

### 3.2 DiffVolume Architecture

DiffVolume is a conditional score-based diffusion model designed to estimate the score function $\mathbf{s}_\theta(\mathbf{x}(t), t, \mathbf{c})$. The architecture is shown in Figure 1. At each diffusion step $t$, the model takes as input a noised volume snapshot denoted by $\mathbf{x}(t)$. The architecture incorporates two embeddings *price level embeddings* $\mathbf{v}$, *time step embeddings* $t$, and an external *conditioning context* $\mathbf{c}$, which includes variables such as past volume trajectories, time of day and target future liquidity (if applicable). Details on the construction of $\mathbf{v}$ and $t$ are provided below, while the structure of $\mathbf{c}$ is described in Subsection 4.3.

The input $\mathbf{x}(t)$ is first processed by a $1 \times 1$ convolutional layer followed by a ReLU activation. This projects the corrupted volume vector into a higher-dimensional latent feature space, making it more suitable for capturing complex interactions. The resulting latent feature is then augmented with learnable *price level embeddings* $\mathbf{v}$, where each discrete price level index is mapped to a continuous dense vector. These embeddings enable the model to distinguish between spatially different price levels in the order book. To capture the dependencies across price levels, the combined input is passed through a multi-head self-attention module [34].

To encode the diffusion step $t$, we adopt a sinusoidal embedding scheme followed by a two-layer MLP with SiLU activations. To incorporate the conditioning information from both $t$ and $\mathbf{c}$, we adopt the Feature-wise Linear Modulation (FiLM) mechanism [27]. Specifically, the FiLM layers learn affine transformations (scaling and shifting) applied to the intermediate features, parameterized by functions of $t$ and $\mathbf{c}$. This allows the model to adjust its internal computations dynamically, enabling more flexible and adaptive generation.

The core of the DiffVolume architecture is a stack of 32 residual convolutional layers, inspired by WaveNet-style dilated convolutions [33]. Each residual block begins with a dilated $3 \times 3$ convolutional layer, which increases the receptive field without increasing model depth. The output is split into two branches: one activated with a `tanh` function, and the other with a `sigmoid` function. The two activations are combined through element-wise multiplication, forming a *soft gating mechanism* that allows the network to selectively pass or suppress signals.

The gated output is then passed through two $1 \times 1$ convolutional layers, each followed by a ReLU activation. A residual connection is added between the input and output of each block, facilitating gradient flow and stability during training. In addition, each residual layer outputs a skip connection, which is accumulated across all layers and routed to the final output head. The aggregated skip connections are combined and passed through a final output sequence: a $1 \times 1$ convolution followed by ReLU activation, and another $1 \times 1$ convolution to produce the final prediction.

We briefly detail the network hyperparameters here. The backbone channel width is set to 256, with 32 gated residual blocks arranged in a dilated cycle $[1, 2, 4, 8, 16]$ repeated to cover all layers. The diffusion timestep embedding of $t$ is sinusoidal of size 128 followed by a two-layer MLP with SiLU activations. Multi-head attention with 32 heads is applied along the price-level axis.

This novel architecture enables the model to integrate fine-grained spatial dependencies within each price level and broader contextual signals across the book and conditioning inputs, allowing it to accurately model the high-dimensional structure of LOB volume trajectories.

## 4 Experimental Setup

In this section, we provide details on the dataset used in our experiments, the preprocessing steps applied to the raw data, as well as the training and sampling procedures.

### 4.1 Data

We use the LOBSTER data[1] as our LOB data source [17]. The platform provides Level-3 data, allowing for a full-fidelity reconstruction of the LOB dynamics. For each day, the dataset contains a message file, which consists of orders sent to an exchange (primarily market, limit, and (partial) cancellation orders), and a snapshot file, which includes the corresponding order book states. To construct our dataset, we sample one snapshot per second from the snapshot files. Each snapshot includes the top 10 levels on both the bid and ask sides, resulting in a volume representation of shape $[20, 1]$ per time step.

In particular, we train, evaluate, and test our model separately on four stocks, which are MU (Micron, big tick stock), AAPL (Apple, medium tick stock), ADBE (Adobe, small tick stock) and ZM (Zoom, small tick stock). Stocks are classified as big, medium, or small tick size according to how their tick size compares to the typical bid-ask spread. Variations in tick size can affect bid-ask spreads, order book depth, and execution costs, thus shaping the strategies of market participants and the dynamics of price formation [7]. For each stock, we use 16 consecutive trading days for training (1 February to 23 February 2023), 1 day for validation (24 February 2023), and the final 2 days for testing (27-28 February 2023).

### 4.2 Preprocessing

To mitigate the non-stationary behavior typically observed in limit order books—particularly around market open and close—we exclude the first and last 30 minutes of each continuous trading session from our analysis. As a result, we retain only snapshots from the 5.5-hour interval between 10:00 AM and 3:30 PM each day. To ensure data quality and prevent computational inconsistencies, we remove all snapshots with missing values. Given that such instances are extremely rare, their exclusion has a negligible effect on the overall statistical properties of the dataset. Setting the sampling frequency to 1 second results in $\sim$ 310k order book snapshots for training, $\sim$ 20k snapshots for validation, and $\sim$ 40k snapshots for testing.

To mitigate the impact of extreme outliers in volume data, we apply a clipping procedure that caps volume values at the 99th percentile for each price level. This enhances model stability and ensures that learning is not dominated by rare but disproportionately large volume observations. We also adopt a square-root-based

[1] https://lobsterdata.com/

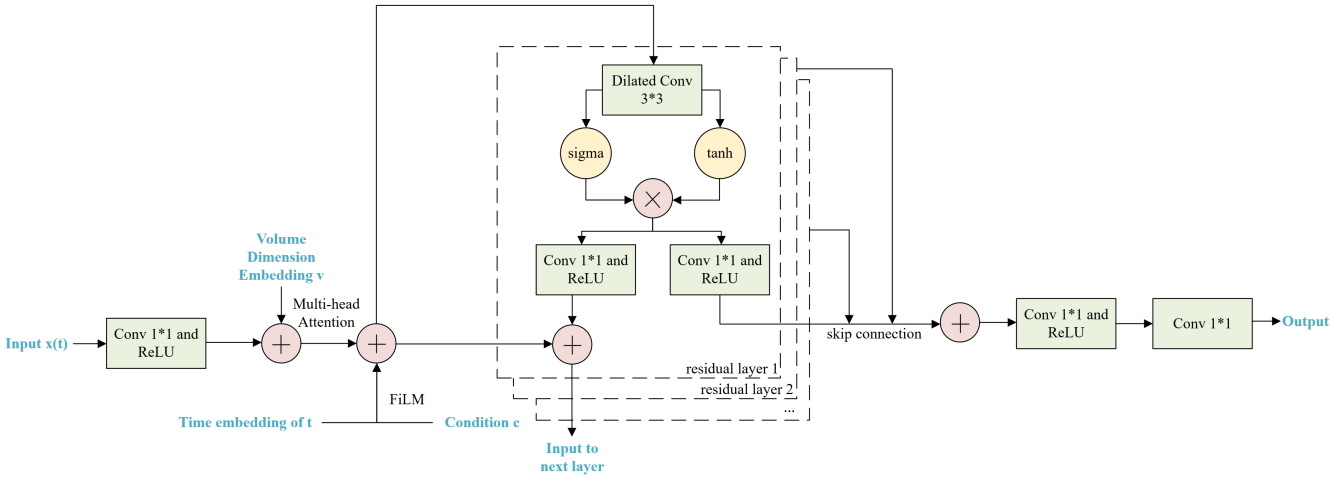



**Figure 1: Illustration of DiffVolume Architecture.**

normalization scheme as shown in Eq. (7). It is coupled with a scaling constant, to stabilize the input distribution and make the training process more effective:

$$x_{normalized} = \frac{\sqrt{x}}{const}, \tag{7}$$

where $const = 15$ during real training.

## 4.3 Training and Sampling

We consider the task of modeling the conditional distribution of future LOB volume snapshots given multiple conditioning variables. Let $\mathbf{x}_{(t-L+1:t)} \in \mathbb{R}^{L\times D}$ denote the sequence of past volume snapshots over $L$ time steps, where each vector $\mathbf{x}_{(i)} \in \mathbb{R}^D$ represents volume across $D$ price levels (20 levels in our case) and $x^j_{(i)}$ represents the volume value at the $j$-th price level in time step $i$. Our objective is to generate a realistic sequence of future volumes $\mathbf{x}_{(t+1:t+L)} \in \mathbb{R}^{L\times D}$ based on a *conditioning context* $\mathbf{c}$, which includes the past volume trajectory $\mathbf{x}_{(t-L+1:t)}$, the time of day $\tau_{(t+1:t+L)}$ and, in counterfactual settings, the future liquidity indicator $\lambda_{(t+1:t+L)}$. The length $L$ is set to 32 in the training and sampling stage.

The time of day $\tau_{(t+1:t+L)}$ is constructed as a normalized scalar ratio reflecting the elapsed time since the market opening, calculated as

$$\tau_{(i)} = \frac{\Delta t_i}{T_{total}}, \quad for\ i \in \{t+1, ...t+L\}, \tag{8}$$

where $\Delta t_i$ is the number of seconds elapsed from the start of the trading session, and $T_{total} = 19800$ seconds denotes the total trading duration (5.5 hours). The resulting sequence $\tau_{(t+1:t+L)} \in \mathbb{R}^{L\times 1}$ serves as a continuous temporal encoding aligned with each future snapshot. By conditioning on the time of the day, we provide temporal context that captures non-stationary behaviors over intraday horizons.

The future liquidity indicator $\lambda$ provides a coarse but informative signal reflecting the expected market depth over the prediction horizon. Each value in $\lambda$ is calculated as the sum of the volume of the corresponding LOB snapshot:

$$\lambda_{(i)} = \sum_{j=1}^{D} x^j_{(i)}, \quad for\ i \in \{t+1, ...t+L\}. \tag{9}$$

Conditioning the model on this sequence, we enable controlled generation, allowing the model to produce volume patterns that not only exhibit realistic microstructure dynamics but also align with target liquidity profiles.

To effectively train the conditional diffusion model and prevent overfitting or instability, we adopt the early stopping [13] and exponential moving average (EMA) mechanism of the network weights [25]. Early stopping is used based on the validation loss, where training is terminated if the validation loss does not improve by at least 0.001 over a patience window of 100 epochs. EMA parameters are updated after each optimization step as $\theta_{EMA} \leftarrow \alpha\cdot\theta_{EMA}+(1-\alpha)\cdot\theta$, where $\alpha = 0.999$. We train the model using Adam optimizer [20] with a learning rate of $1\times 10^{-4}$ and a batch size of 64. The diffusion process follows the DDPMs parameterization with 100 discrete noise scales, and we adopt the ancestral sampling procedure [15], a widely used method in DDPMs, to generate LOB volume snapshots.

# 5 Experimental Results

In this section, we address three key research questions. (1) To what extent do the generated samples exhibit statistical fidelity to real-world limit order book data? (2) Can DiffVolume generate meaningful counterfactual samples in response to varying liquidity conditions? (3) Are the counterfactually generated samples informative enough to support downstream applications, such as future liquidity prediction?

## 5.1 Realism

We begin by assessing the realism of the generated LOB volumes from different models, focusing on their ability to replicate key statistical properties observed in real market data. We compare four sources of volume data: (i) the empirical testing dataset (*Real*), (ii) diffusion model without future liquidity condition (*Diff Uncond*),

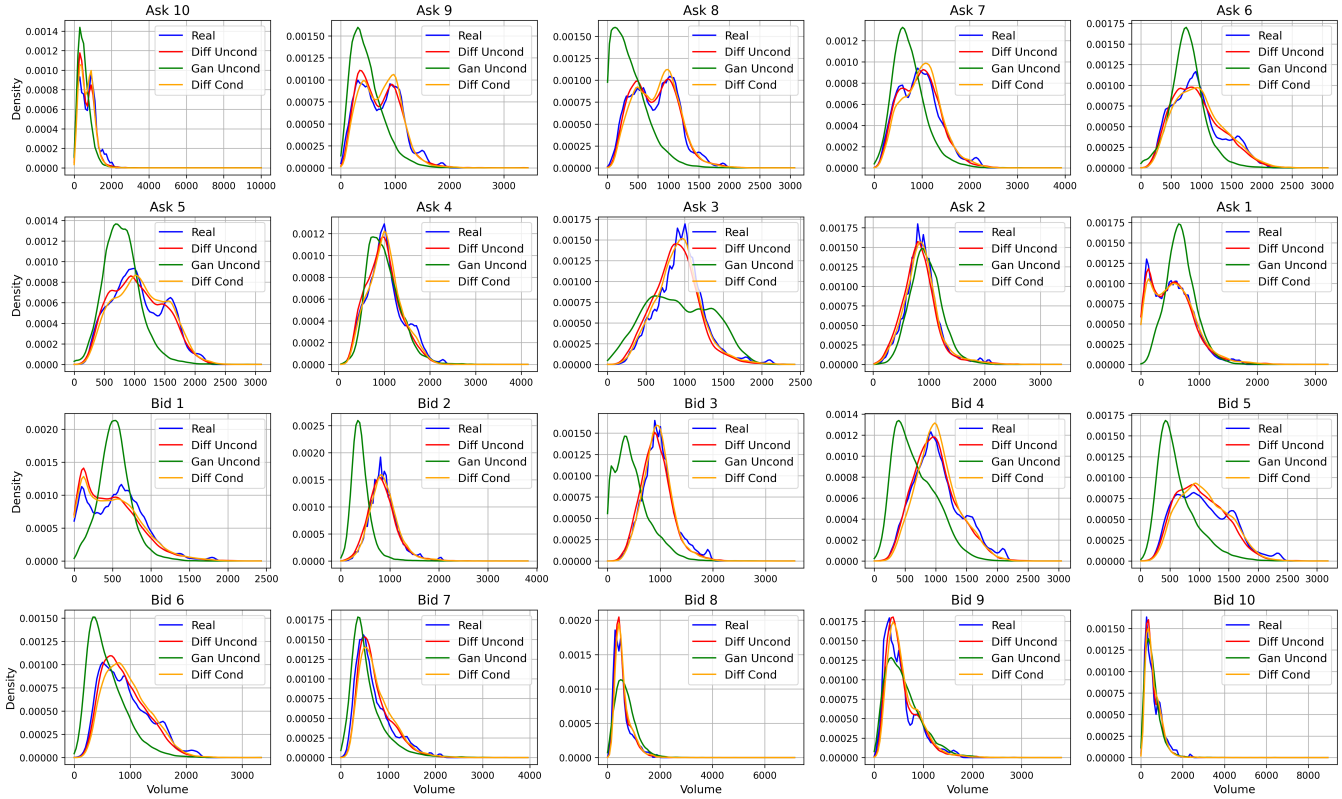



**Figure 2: Marginal Volume Distribution.**

(iii) WGAN-GP model without future liquidity condition (*GAN Uncond*), and (iv) diffusion model with future liquidity condition (*Diff Cond*). To ensure a fair comparison of generative quality, we primarily focus on *GAN Uncond* and *Diff Uncond*, where no strong future information is provided during training. The *Diff Cond* model, although conditioned on future liquidity profiles, is included here for completeness, as it plays a key role in later sections and shows promise for future applications. We will take MU as an illustrative example for our plots and summarize the performance for the four stocks in Table 1.

**Marginal Volume Distributions.** Figure 2 presents kernel density estimates of marginal volume distributions at each price level. Real LOB volumes exhibit multi-modal and heavy-tailed characteristics. GAN-generated samples fail to reproduce these higher-order statistics, often yielding overly smooth unimodal distributions. Both *Diff Uncond* and *Diff Cond* accurately capture the skewness and kurtosis of empirical data, outperforming *GAN Uncond*.

**Average Volume per Price Level.** Figure 3 displays the average volume across each of the 10 ask and 10 bid price levels. The real volume exhibits a clear symmetric hump-shaped structure around the best bid and ask prices, with the highest concentration of liquidity at levels 3–6. *GAN Uncond* underestimates volume at nearly all levels. In contrast, *Diff Uncond* and *Diff Cond* closely match the empirical averages, suggesting its ability to reproduce realistic liquidity profiles.

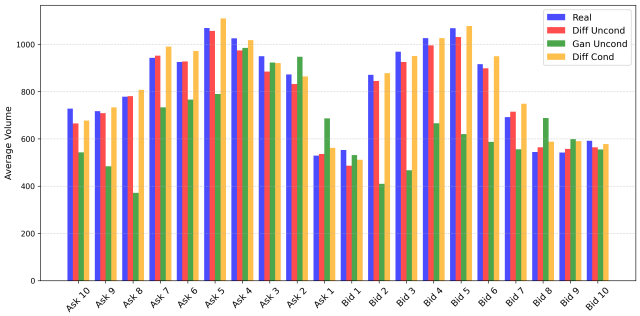



**Figure 3: Average Volume per Price Level.**

**Cross-sectional and Temporal Difference Correlation.** Figure 4 shows the cross-sectional volume correlation across 20 price levels (Ask 10 to Bid 10). Compared with *GAN Uncond*, *Diff Uncond* and *Diff Cond* can better capture the spatial structure in the upper left quadrant (Ask 10 to Ask 7) and inter-side interactions (Ask 2 to Ask 6, Bid 2 to Bid 6). Figure 5 is calculated from the first-order differences of the volume snapshots ($\Delta \mathbf{x}_{(i)} = \mathbf{x}_{(i)} - \mathbf{x}_{(i-1)}$). A notable characteristic in the temporal difference correlation structure is the negative correlation at adjacent price levels, which is captured by all three models. However, *Diff Uncond* and *Diff Cond* do better in the upper left quadrant (Ask 10 to Ask 7).

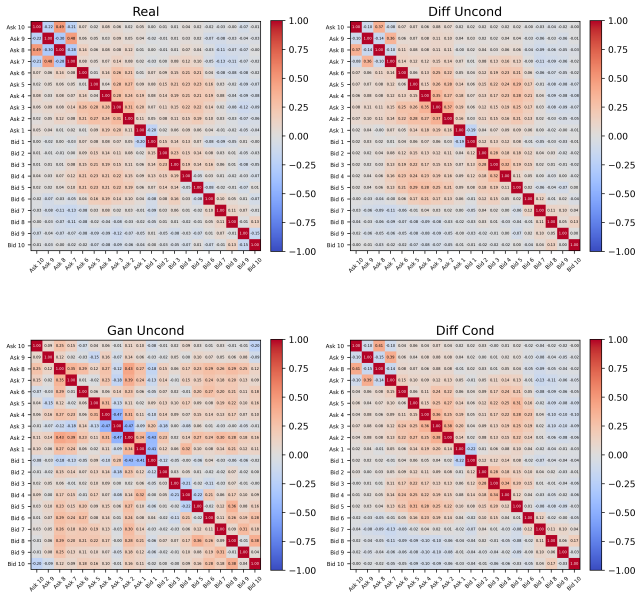



**Figure 4: Cross-sectional Volume Correlation.**

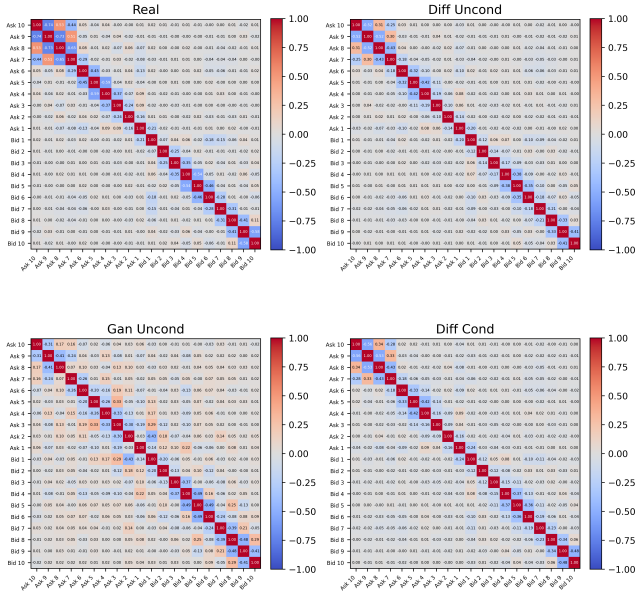



**Figure 5: Temporal Difference Volume Correlation.**

**Autocorrelation Decay.** Finally, we evaluate the long memory properties of the generated volume series by analyzing autocorrelation decay at each price level, shown in Figure 6 as a log-log plot. In empirical financial time series, volume autocorrelation is known to approximately follow a scaling law [14]:

$$ACF(l) \sim C \cdot l^{-\gamma}, \tag{10}$$

where $ACF(l)$ denotes the lag-$l$ autocorrelation. The conditional diffusion model closely replicates these dynamics, while the unconditional diffusion model slightly underperforms with faster decay rates. In contrast, the GAN-based approach demonstrates unstable and unrealistic temporal correlations.

To quantitatively evaluate the realism of the generated volume distributions against empirical data, we employ three divergence metrics: Wasserstein distance, Kullback–Leibler (KL) divergence, and Kolmogorov–Smirnov (KS) statistic. The results, summarized in Table 1, consistently indicate that the proposed diffusion-based models substantially outperform the GAN baseline in all four stocks, validating their effectiveness in capturing the empirical volume distribution. Specifically, *Diff Cond* achieves slightly better performance than its unconditional counterpart *Diff Uncond*, highlighting the benefit of incorporating conditioning information. Additionally, *Diff Uncond* outperforms *Diff-CSDI Uncond* model, a diffusion model variant based on the CSDI architecture [32], in almost all cases. This result underscores the superiority of our proposed DiffVolume architecture in modeling limit order book volume distributions.

## 5.2 Counterfactual Generation under Extreme Liquidity Scenarios

To evaluate the controllability of *Diff Cond* under varying liquidity conditions, we design a counterfactual generation experiment based on liquidity-driven partitioning of the testing dataset. Specifically, we sort all snapshots into five bins (denoted as Bin 1 to Bin 5) based on ascending order of liquidity $\lambda$. Bin 1 represents the lowest liquidity regime, whereas Bin 5 corresponds to the highest. The pairwise discrepancies are quantified using the Wasserstein distance, and the results are reported in Table 2.

We first compare `Real_bin VS Fake_bin`, where `Real_bin` refers to the subset of real snapshots that fall into each of the five liquidity bins, and `Fake_bin` denotes the corresponding generated samples from *Diff Cond*, conditioned on the past volume history, time of day, and the actual future liquidity value associated with each snapshot in the bin. `Real_all` denotes the entire testing dataset and `Fake_all` denotes the generated dataset by following exact liquidity conditions of `Real_all`. As shown in Table 2, the Wasserstein distances between `Real_bin` and `Fake_bin` are consistently lower than those between `Real_all` and `Fake_bin`. This confirms that *Diff Cond* is highly effective in generating samples closely aligned with the intended liquidity regime.

To further assess the model's ability to perform counterfactual generation, we fix the past volume trajectory $\mathbf{x}_{(t-L+1:t)}$ and time of day condition $\tau$ while altering the target liquidity condition $\lambda$ to simulate extreme scenarios:

- `Fake_OL_all` (Over Liquidity): For every snapshot in the test set, we randomly sample a future liquidity profile from Bin 5 (highest liquidity) and generate new snapshots using the fixed past conditions, time of day and sampled high liquidity.
- `Fake_UL_all` (Under Liquidity): Similarly, we sample the liquidity condition from Bin 1 (lowest liquidity) while keeping the rest unchanged.

The results show clear trends. The Wasserstein distances between `Real_bin` and `Fake_OL_all` decrease progressively from Bin 1 to Bin 5. This indicates that the samples generated under the fixed high liquidity condition resemble more closely those in the high-liquidity bin, validating the model's ability to simulate high-liquidity scenarios regardless of the original snapshot conditions. Conversely,

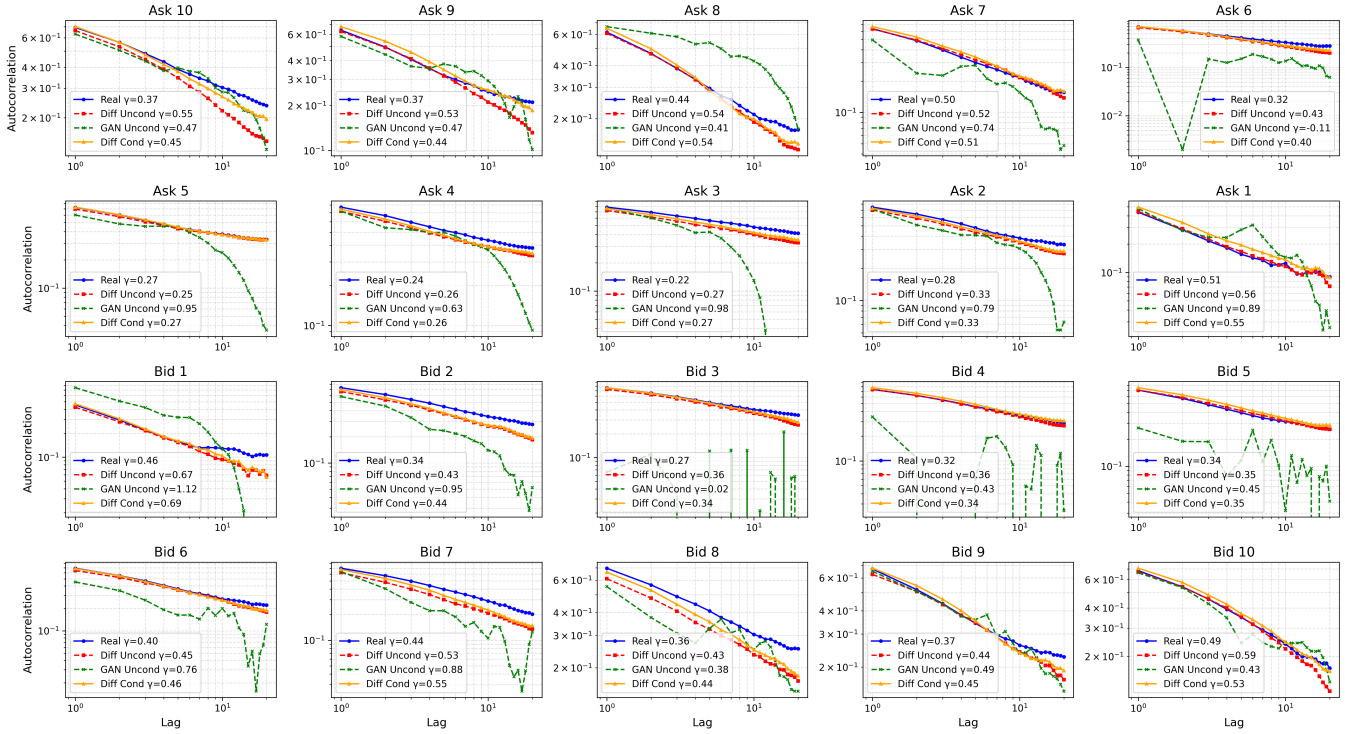



**Figure 6: Autocorrelation Decay.**

| | MU | | | AAPL | | | ADBE | | | ZM | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Wasserstein | KL | KS | Wasserstein | KL | KS | Wasserstein | KL | KS | Wasserstein | KL | KS |
| Diff Uncond | **40.34** | **0.256** | **0.091** | 165.783 | 0.291 | 0.25 | 12.169 | **0.344** | **0.136** | 24.509 | 0.199 | 0.135 |
| GAN Uncond | 229.106 | 0.687 | 0.31 | 190.955 | 0.508 | 0.258 | 30.398 | 1.155 | 0.348 | 94.96 | 0.635 | 0.335 |
| Diff Cond | 42.603 | 0.266 | 0.094 | **102.264** | **0.143** | 0.166 | **11.426** | 0.387 | 0.137 | **16.541** | **0.193** | **0.123** |
| Diff-CSDI Uncond | 172.513 | 0.485 | 0.137 | 265.967 | 0.058 | **0.138** | 47.583 | 0.5 | 0.212 | 100.475 | 0.342 | 0.178 |

**Table 1: Realism Comparison.**

the distances between `Real_bin` and `Fake_UL_all` increase from Bin 1 to Bin 5, suggesting that the low-liquidity condition effectively shifts the output distribution towards the characteristics of low-liquidity environments.

**A Case Study on AAPL.** We observe that AAPL exhibits an anomalous pattern across bins 3 to 5. Specifically, the Wasserstein distances in `Real_bin VS Fake_bin` are unexpectedly higher than those in `Real_all VS Fake_bin`, which is contrary to the trend observed in MU, ADBE, and ZM. Furthermore, in the counterfactual setting (`Real_bin VS Fake_OL_all`), the expected monotonic decrease in distance from bin 1 to bin 5 is not strictly preserved. Upon examining the underlying data distributions, we find that the liquidity profiles between AAPL's training and testing datasets differ more substantially than in other stocks. In particular, the average liquidity values across the five bins in the training dataset are $\sim [1.08, 1.36, 1.54, 1.73, 2.07] \times 10^4$, while in the testing dataset they are $\sim [1.54, 1.74, 1.87, 2.00, 2.20] \times 10^4$. Notably, the average liquidity value of bin 3 in the training dataset is approximately equal to that of bin 1 in the testing dataset. This upward shift indicates a distributional drift in liquidity levels between the training and testing phases, which causes the model to misinterpret the intended conditioning during generation when extrapolating to high-liquidity regimes.

## 5.3 A Downstream task: Future Liquidity Prediction

To evaluate the utility of synthetic data generated by the proposed *Diff Cond* model, we conduct a downstream task of short-term liquidity forecasting across the four stocks. Specifically, we aim to predict the aggregate LOB volume over the next $Horizon \in \{10, 30\}$ seconds, given a history window of 32 snapshots. Formally, the prediction target at time $t$ is defined as follows:

$$y_t = \sum_{i=1}^{H} \sum_{j=1}^{20} x^j_{(t+i)}. \tag{11}$$

We employ a LightGBM regressor trained on 27–28 February 2023, and test it on a held-out dataset on March 1, 2023. The model uses the gradient boosting decision tree (GBDT) framework with key

| | Liquidity Quintile | | | | |
|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 |
| MU | | | | | |
| Real_bin VS Fake_bin | 41.195 | 43.223 | 48.12 | 46.884 | 54.666 |
| Real_all VS Fake_bin | 163.107 | 79.411 | 55.664 | 108.858 | 154.762 |
| Real_bin VS Fake_OL_all | 306.239 | 215.215 | 143.541 | 142.479 | 137.262 |
| Real_bin VS Fake_UL_all | 120.942 | 206.558 | 293.692 | 357.38 | 426.005 |
| AAPL | | | | | |
| Real_bin VS Fake_bin | 74.576 | 89.737 | 105.091 | 118.66 | 143.808 |
| Real_all VS Fake_bin | 223.924 | 143.918 | 98.399 | 62.155 | 70.803 |
| Real_bin VS Fake_OL_all | 194.573 | 123.719 | 115.833 | 134.759 | 177.102 |
| Real_bin VS Fake_UL_all | 81.01 | 162.991 | 225.618 | 288.545 | 387.232 |
| ADBE | | | | | |
| Real_bin VS Fake_bin | 9.566 | 10.172 | 10.817 | 12.085 | 16.13 |
| Real_all VS Fake_bin | 16.363 | 10.146 | 11.207 | 15.574 | 26.742 |
| Real_bin VS Fake_OL_all | 42.698 | 31.313 | 24.686 | 18.839 | 14.745 |
| Real_bin VS Fake_UL_all | 11.187 | 8.188 | 13.509 | 21.307 | 36.276 |
| ZM | | | | | |
| Real_bin VS Fake_bin | 13.958 | 15.316 | 15.024 | 17.637 | 26.046 |
| Real_all VS Fake_bin | 46.654 | 20.906 | 19.718 | 28.967 | 65.739 |
| Real_bin VS Fake_OL_all | 112.867 | 84.289 | 67.306 | 49.439 | 27.286 |
| Real_bin VS Fake_UL_all | 16.607 | 26.199 | 43.166 | 63.149 | 102.469 |

**Table 2: Evaluation of Counterfactual Generation**

| | Horizon = 10 | | | Horizon = 30 | | |
|---|---|---|---|---|---|---|
| | MSE | MAE | $R^2$ | MSE | MAE | $R^2$ |
| MU | | | | | | |
| Real | 162896410 | 9841 | 0.7996 | 1584188284 | 30962 | 0.7782 |
| Real + Synthetic | 145313593 | 9590 | 0.8212 | 1439704748 | 30288 | 0.7985 |
| Percentage | 10.79% | 2.55% | 2.70% | 9.12% | 2.17% | 2.61% |
| AAPL | | | | | | |
| Real | 180833992 | 9831 | 0.717 | 1796092385 | 31156 | 0.6644 |
| Real + Synthetic | 137597830 | 8892 | 0.7846 | 1601342963 | 29751 | 0.7008 |
| Percentage | 23.91% | 9.55% | 9.43% | 10.84% | 4.51% | 5.48% |
| ADBE | | | | | | |
| Real | 3358510 | 1464 | 0.4009 | 30205360 | 4422 | 0.1906 |
| Real + Synthetic | 3239457 | 1428 | 0.4222 | 28691234 | 4228 | 0.2312 |
| Percentage | 3.54% | 2.46% | 5.31% | 5.01% | 4.39% | 21.30% |
| ZM | | | | | | |
| Real | 705321941 | 23866 | -2.8648 | 7676911350 | 81013 | -4.3042 |
| Real + Synthetic | 652072041 | 22684 | -2.573 | 6195517329 | 71388 | -3.2806 |
| Percentage | 7.55% | 4.95% | 10.19% | 19.30% | 11.88% | 23.78% |

**Table 3: Liquidity Prediction Results.**

hyperparameters set to balance predictive accuracy and generalization: the number of leaves is set to 31 with unrestricted tree depth, a learning rate of 0.1, and 100 boosting iterations.

We compare two training scenarios: `Real` represents that the model is trained solely on the real dataset, while `Real + Synthetic` represents that the model is trained on an augmented dataset combining `Real_all`, `Fake_all`, `Fake_OL_all` and `Fake_UL_all` introduced in Subsection 5.2. Results are reported in Table 3, using Mean Squared Error (MSE), Mean Absolute Error (MAE), $R^2$, along with corresponding percentage improvement. By augmenting the training data with generated samples, we achieve consistent improvements in forecasting short-term liquidity. These results demonstrate that DiffVolume can also be practically beneficial by providing unseen samples in downstream prediction tasks.

## 6 Conclusion

In this paper, we propose a novel conditional denoising diffusion probabilistic model for generating future limit order book (LOB) volume snapshots, conditioned on historical order flow, time of day, and optionally target future liquidity levels. Through extensive empirical evaluations, we show that it achieves high realism, closely replicating statistical properties of real market volumes. Second, it supports counterfactual generation, enabling scenario-based simulations under hypothetical liquidity conditions. Third, it proves to be useful for a short-term liquidity prediction task by augmenting real data with synthetic samples. These results highlight the potential of conditional diffusion models as a robust and flexible generative framework for high-frequency financial modeling. Future work includes extending the model to multi-asset settings, joint price-volume generation, and applications to market simulation.

## Acknowledgments

The authors would like to thank Nikolas Nüsken from the Department of Mathematics at King's College London for his insightful feedback and helpful suggestions on the manuscript.

## References

[1] Selim Amrouni, Aymeric Moulin, Jared Vann, Svitlana Vyetrenko, Tucker Balch, and Manuela Veloso. 2021. ABIDES-gym: gym environments for multi-agent discrete event simulation and application to financial markets. In *Proceedings of the Second ACM International Conference on AI in Finance*. 1–9.

[2] Brian DO Anderson. 1982. Reverse-time diffusion equation models. *Stochastic Processes and their Applications* 12, 3 (1982), 313–326.

[3] Emmanuel Bacry, Iacopo Mastromatteo, and Jean-François Muzy. 2015. Hawkes processes in finance. *Market Microstructure and Liquidity* 1, 01 (2015), 1550005.

[4] Tucker Hybinette Balch, Mahmoud Mahfouz, Joshua Lockhart, Maria Hybinette, and David Byrd. 2019. How to evaluate trading strategies: Single agent market replay or multiple agent interactive simulation? *arXiv preprint arXiv:1906.12010* (2019).

[5] Luc Bauwens and Nikolaus Hautsch. 2009. Modelling financial high frequency data using point processes. In *Handbook of financial time series*. Springer, 953–979.

[6] Jean-Philippe Bouchaud, Julius Bonart, Jonathan Donier, and Martin Gould. 2018. *Trades, quotes and prices: financial markets under the microscope.* Cambridge University Press.

[7] Antonio Briola, Silvia Bartolucci, and Tomaso Aste. 2025. HLOB–Information persistence and structure in limit order books. *Expert Systems with Applications* 266 (2025), 126078.

[8] David Byrd, Maria Hybinette, and Tucker Hybinette Balch. 2019. Abides: Towards high-fidelity market simulation for AI research. *arXiv preprint arXiv:1904.12066* (2019).

[9] Stanley Chan et al. 2024. Tutorial on diffusion models for imaging and vision. *Foundations and Trends® in Computer Graphics and Vision* 16, 4 (2024), 322–471.

[10] Andrea Coletta, Matteo Prata, Michele Conti, Emanuele Mercanti, Novella Bartolini, Aymeric Moulin, Svitlana Vyetrenko, and Tucker Balch. 2021. Towards realistic market simulations: a generative adversarial networks approach. In *Proceedings of the Second ACM International Conference on AI in Finance*. 1–9.

[11] Rama Cont, Mihai Cucuringu, Jonathan Kochems, and Felix Prenzel. 2023. Limit order book simulation with generative adversarial networks. *Available at SSRN 4512356* (2023).

[12] Vladimir Filimonov and Didier Sornette. 2012. Quantifying reflexivity in financial markets: Toward a prediction of flash crashes. *Physical Review E—Statistical, Nonlinear, and Soft Matter Physics* 85, 5 (2012), 056108.

[13] Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. 2016. *Deep learning*. Vol. 1. MIT press Cambridge.

[14] Martin D Gould, Mason A Porter, Stacy Williams, Mark McDonald, Daniel J Fenn, and Sam D Howison. 2013. Limit order books. *Quantitative Finance* 13, 11 (2013), 1709–1742.

[15] Jonathan Ho, Ajay Jain, and Pieter Abbeel. 2020. Denoising diffusion probabilistic models. *Advances in neural information processing systems* 33 (2020), 6840–6851.

[16] Jonathan Ho and Tim Salimans. 2021. Classifier-free diffusion guidance. *NeurIPS 2021 Workshop on Deep Generative Models and Downstream Applications* (2021).

[17] Ruihong Huang and Tomas Polak. 2011. Lobster: Limit order book reconstruction system. *Available at SSRN 1977207* (2011).

[18] Hanna Hultin, Henrik Hult, Alexandre Proutiere, Samuel Samama, and Ala Tarighati. 2023. A generative model of a limit order book using recurrent neural networks. *Quantitative Finance* 23, 6 (2023), 931–958.

[19] Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. 2022. Elucidating the design space of diffusion-based generative models. *Advances in Neural Information Processing Systems* 35 (2022), 26565–26577.
[20] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. *International Conference on Learning Representations* (2015).
[21] Kelvin JL Koa, Yunshan Ma, Ritchie Ng, and Tat-Seng Chua. 2023. Diffusion variational autoencoder for tackling stochasticity in multi-step regression stock price prediction. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*. 1087–1096.
[22] Junyi Li, Xintong Wang, Yaoyang Lin, Arunesh Sinha, and Michael Wellman. 2020. Generating realistic stock market order streams. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 727–734.
[23] Calvin Luo. 2022. Understanding diffusion models: A unified perspective. *arXiv preprint arXiv:2208.11970* (2022).
[24] Peer Nagy, Sascha Frey, Silvia Sapora, Kang Li, Anisoara Calinescu, Stefan Zohren, and Jakob Foerster. 2023. Generative AI for end-to-end limit order book modelling: A token-level autoregressive generative model of message flow using a deep state space network. In *Proceedings of the Fourth ACM International Conference on AI in Finance*. 91–99.
[25] Alexander Quinn Nichol and Prafulla Dhariwal. 2021. Improved denoising diffusion probabilistic models. In *International conference on machine learning*. PMLR, 8162–8171.
[26] Mark Paddrik, Roy Hayes, Andrew Todd, Steve Yang, Peter Beling, and William Scherer. 2012. An agent based model of the E-Mini S&P 500 applied to Flash Crash analysis. In *2012 IEEE conference on computational intelligence for financial engineering & economics (CIFEr)*. IEEE, 1–8.
[27] Ethan Perez, Florian Strub, Harm De Vries, Vincent Dumoulin, and Aaron Courville. 2018. Film: Visual reasoning with a general conditioning layer. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 32.
[28] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. 2015. Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*. PMLR, 2256–2265.
[29] Jiaming Song, Chenlin Meng, and Stefano Ermon. 2021. Denoising diffusion implicit models. *International Conference on Learning Representations* (2021).
[30] Yang Song and Stefano Ermon. 2019. Generative modeling by estimating gradients of the data distribution. *Advances in neural information processing systems* 32 (2019).
[31] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. 2021. Score-based generative modeling through stochastic differential equations. *International Conference on Learning Representations* (2021).
[32] Yusuke Tashiro, Jiaming Song, Yang Song, and Stefano Ermon. 2021. CSDI: Conditional score-based diffusion models for probabilistic time series imputation. *Advances in Neural Information Processing Systems* 34 (2021), 24804–24816.
[33] Aäron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew W. Senior, and Koray Kavukcuoglu. 2016. WaveNet: A Generative Model for Raw Audio. In *Proceedings of the 9th ISCA Speech Synthesis Workshop (SSW)*. 125–131.
[34] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems* 30 (2017).
[35] Pascal Vincent. 2011. A connection between score matching and denoising autoencoders. *Neural computation* 23, 7 (2011), 1661–1674.
[36] Zhuohan Wang and Carmine Ventre. 2024. A Financial Time Series Denoiser Based on Diffusion Models. In *Proceedings of the 5th ACM International Conference on AI in Finance*. 72–80.