Controllable Financial Market Generation with Diffusion Guided Meta Agent

Yu-Hao Huang
huangyh@smail.nju.edu.cn
National Key Laboratory for Novel Software Technology,
Department of Computer Science and Technology, Nanjing University
Nanjing, China

Chang Xu
chanx@microsoft.com
Microsoft Research Asia
Beijing, China

Yang Liu
yangliu2@microsoft.com
Microsoft Research Asia
Beijing, China

Weiqing Liu
weiqing.liu@microsoft.com
Microsoft Research Asia
Beijing, China

Wu-Jun Li
liwujun@nju.edu.cn
National Key Laboratory for Novel Software Technology,
Department of Computer Science and Technology, Nanjing University
Nanjing, China

Jiang Bian
jiang.bian@microsoft.com
Microsoft Research Asia
Beijing, China

Abstract

Order flow modeling stands as the most fundamental and essential financial task, as orders embody the minimal unit within a financial market. However, current approaches often result in unsatisfactory fidelity in generating order flow, and their generation lacks controllability, thereby limiting their application scenario. In this paper, we advocate incorporating controllability into the market generation process, and propose a *Diffusion Guided meta Agent* (DiGA) model to address the problem. Specifically, we utilize a diffusion model to capture dynamics of market state represented by time-evolving distribution parameters about mid-price return rate and order arrival rate, and define a meta agent with financial economic priors to generate orders from the corresponding distributions. Extensive experimental results demonstrate that our method exhibits outstanding controllability and fidelity in generation. Furthermore, we validate DiGA's effectiveness as generative environment for downstream financial applications.

CCS Concepts

• **Computing methodologies → Machine learning algorithms**;
• **Applied computing → Electronic funds transfer**.

Keywords

Time Series Generation, Financial, Diffusion Model

Conference acronym 'XX, June 03–05, 2018, Woodstock, NY

ACM ISBN 978-1-4503-XXXX-X/18/06
https://doi.org/XXXXXXX.XXXXXXX

ACM Reference Format:
Yu-Hao Huang, Chang Xu, Yang Liu, Weiqing Liu, Wu-Jun Li, and Jiang Bian. 2018. Controllable Financial Market Generation with Diffusion Guided Meta Agent. In *Proceedings of Make sure to enter the correct conference title from your rights confirmation emai (Conference acronym 'XX)*. ACM, New York, NY, USA, 12 pages. https://doi.org/XXXXXXX.XXXXXXX

1 Introduction

Generative modeling has impacted a spectrum of machine learning applications, from natural language processing [5], media synthesis [27] to medical applications [2]. Advanced generative methods, such as diffusion models, have demonstrated potentials in producing realistic generation contents [17, 28, 31]. While financial technology is an application sector naturally involved with data of high intricacy, the literature has seldom explored modeling financial data generatively. Most current financial technology (FinTech) efforts take market replay [3] as environment, facing challenges in learning genuine market dynamics interactively. In contrast, market generation provides a more interactive and realistic environment for FinTech tasks, holding potentials for various downstream financial tasks.

Among categories of financial data, order represents the minimal unit of events within a financial market. Since investigating the intrinsic interactive logic and micro-structure of financial market is crucial among researchers [26], investors [12], and policy makers [24], it is important to constitute generated financial worlds that realistically simulate financial markets at order-level with interactiveness [12, 15].

Recent works have attempted to simulate order-level financial market with agent-based methods, either using rule-based agents [1, 6, 32] or learned agents [10, 21]. They aim at replicating "stylized facts" observed in real markets such as volatility clustering, but have resulted in limited fidelity. Ruled-based agents rely on over-simplified assumptions of the market without being trained on real market data, which hurts the simulation fidelity. Learned agents are trained with limited fraction of real world samples due to the intricacy of order-level data, resulting in lower coverage of financial market scenarios. More importantly, the focus of these works is primarily on narrowing the gap between the real and simulated market. The controllability to the generated market is absent from the literature, which is crucial for downstream tasks such as discovering counterfactual cases [15].

Different from existing works, we propose the incorporation of controllability in financial market modeling and formulating the problem as a conditional generation task. The objective involves constructing specific scenarios with varying levels of asset return, intraday volatility, and rare occurrences such as sharp drops or extreme amplitudes. To achieve this, a critical challenge lies in establishing a connection between control targets, such as desired market scenarios, and the generated orders. Inspired by recent advances in diffusion-based generative models with guidance techniques [17, 18, 27], the problem can be addressed by a conditional diffusion model. However, exercising control at the order level is quite challenging, as (1) the long and irregular length of real-world order flow samples block the feasibility of applying diffusion models directly on the raw order-level data, (2) linking the "macro" control target with the every "micro" order separately may not make sense due to the low signal-to-noise ratio in order flow.

In this paper, we present the problem formulation of controllable financial market generation, and propose a *Diffusion Guided meta Agent* (DiGA) model to address the problem. Specifically, we utilize a conditional diffusion model to capture dynamics of market state represented by time-evolving distribution parameters about mid-price return rate and order arrival rate, and define a meta agent with financial economic priors to samples orders from the distributions defined by the aforementioned parameters. With DiGA, we are able to control the generation to simulate order flow given target scenario with high fidelity. Our contributions can be summarized as follows:

- We formulate controllable financial market generation problem as a novel challenge for machine learning application in finance.
- To the best of our knowledge, DiGA stands as the pioneering model that integrates advanced diffusion-based generative models into the domain of financial market modeling.
- Our experiments, using extensive real stock market data, show DiGA excelling in controlling financial market generation and achieving the highest fidelity to stylized facts. Additionally, its effectiveness is validated with high-frequency trading task, indicating significant potential for assisting classical finance downstream.

2 Preliminaries and related work

In this section, we briefly describe the preliminaries of limit order book, and review the related research work regarding financial market simulation and diffusion models.

2.1 Limit order book

The majority of financial markets around the world operate on a double-auction system, where orders are the minimal units of events. An order in the market consists of four basic elements: timestamp t, price p, quantity q, and order type o. There are a variety of order types in real markets, such as limit orders, market orders, cancel orders, conditional order, etc. In the literature, it is sufficient to represent trade decisions with limit orders and cancel orders [8].

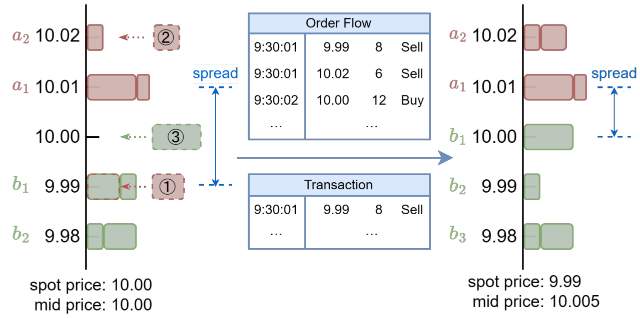


Figure 1: Limit order book and order flow

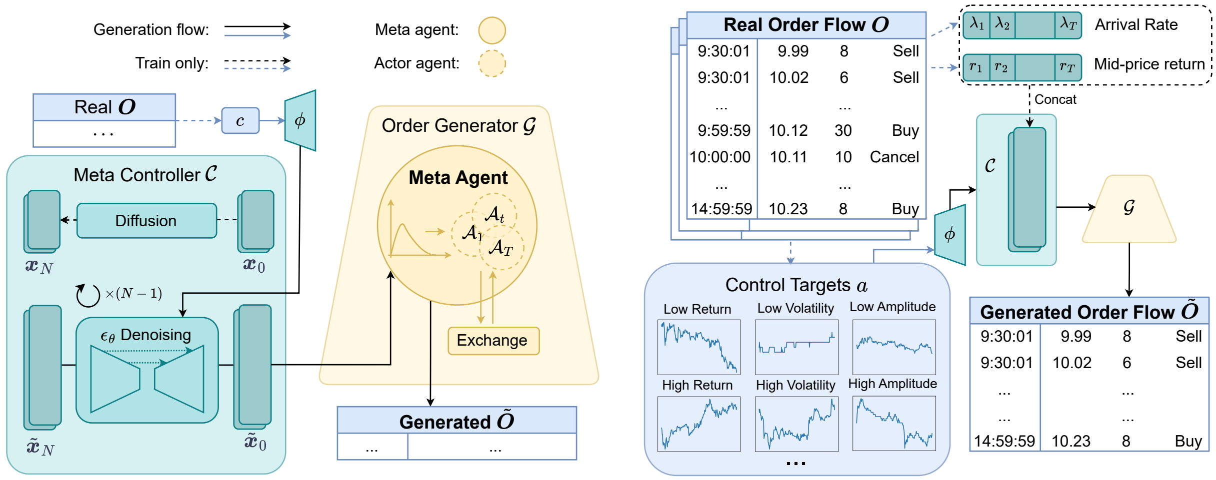
The output of market simulation model is a series of orders $\boldsymbol{O} = \{(t_1, p_1, q_1, o_1), (t_2, p_2, q_2, o_2), \ldots\}$, also known as the order flow, which constitutes the order book and price series. An order book is the collection of outstanding orders that have not been executed. Limit orders can be further categorized into buy limit orders and sell limit orders, with their prices known as bids and asks, respectively. The order book specifically consisting of limit orders are called the limit order book (LOB). The price at each timestamp is commonly determined by the price of the most recently executed order. Prices sampled at a certain frequency form the price series. An illustrative example of limit order book can be found in Figure 1.

2.2 Financial market simulation

Early works on market simulation has followed a multi-agent approach [23, 25, 26], using rule-based agents under simplified trading protocols to replicate "stylized facts" [13] such as volatility clustering. Chiarella and Iori [7], Chiarella et al. [8] extended the simulation analysis to order-driven markets, which more closely resemble the settings of most of current active stock markets. Subsequent works further customized agents' behaviors based on this protocol to better support research on decision-making [6, 32, 33]. With the recent success of machine learning, researcher has also employed neural networks as world agent to directly predict orders given history [10–12, 21]. In contrast, our model employs conditional diffusion model for controlling agent based models to generate orders.

2.3 Diffusion models

The diffusion probabilistic models, also known as the diffusion models, fit sequential small perturbations from a diffusion process to convert between known and target distributions [17, 28, 31]. Diffusion models have been built to generate data in different modalities, such as image [2, 14, 27, 30], audios [20, 22], videos [4, 16] and general time series [9, 19, 34]. Different from existing works, we are the first to apply diffusion model for generating financial market.



(a) Meta controller and order generator in DiGA.

(b) Data flow of DiGA.

Figure 2: Overview of DiGA model. Raw order flow are proccessed into market states for the meta controller to fit. The meta agent is guided by the meta controller to generate simulated order flow.

3 Method

In this section, we first present the problem formulation of controllable financial market generation. Then, we present the detailed architecture design of our *Diffusion Guided meta Agent* (DiGA) model.

3.1 Problem formulation of controllable financial market generation

Different from common market simulation that simulates unconditionally, controllable market generation aims to simulate order flows that correspond to certain scenario with generative models. Possible scenarios can be characterized by indicators that represent the aggregative statistics of an order flow, such as daily return, daily amplitude and intraday volatility. Formally, let $\mathcal{F}$ denote the function for calculating any of the indicators, given real order flow sample $\boldsymbol{O} \sim q(\boldsymbol{O})$, the value of indicator a can be calculated with $a = \mathcal{F}(\boldsymbol{O})$.

In controllable financial market generation, a market generator $\mathcal{M}$ provides a conditional sampler $p_{\mathcal{M}}(\boldsymbol{O}|\cdot)$. Given control target as a required market scenario specified by indicator of value a, we denote the conditional sampling output as $\tilde{O} \sim p_{\mathcal{M}}(\boldsymbol{O}|a)$. The controllability objective is to minimize the distance between the conditional input a and the indicator representing the same aggregative statistics of generated order flow $\tilde{a} = \mathcal{F}(\tilde{\boldsymbol{O}})$:

$$\min_{\mathcal{M}} \mathbb{E}_{a,\tilde{a}} \left[\|\tilde{a} - a\|^2 \right] = \min \mathbb{E}_{a,\tilde{\boldsymbol{O}} \sim p_{\mathcal{M}}(\boldsymbol{O}|a)} \left[\|\mathcal{F}(\tilde{\boldsymbol{O}}) - a\|^2 \right]. \quad (1)$$

In addition, it is also important for the model to generate order flow with high fidelity in the context of controllable market. The fidelity objective is aligned with general market simulation problems, defined as minimizing the distribution divergence measure $\mathcal{D}$ on "stylized facts" between the real and generated order flow. Here, the "stylized facts" can also be expressed as a set of aggregative statistics to be calculated with their corresponding formula. Let $\mathcal{F}'$ denotes an arbitrary function of stylized fact and $p(\cdot)$ denotes probability density, the fidelity objective writes:

$$\min_{\mathcal{M}} \mathbb{E}_{\boldsymbol{O} \sim q(\boldsymbol{O}), \tilde{\boldsymbol{O}} \sim p_{\mathcal{M}}(\boldsymbol{O}|\cdot)} \mathcal{D} \left(p(\mathcal{F}'(\tilde{\boldsymbol{O}})) \parallel p(\mathcal{F}'(\boldsymbol{O})) \right). \quad (2)$$

3.2 Diffusion guided meta agent model

Order flow data is highly intricate and noisy, with a huge amount typically counting for tens of thousand per stock daily which is computational challenging. Thus it is non-trivial modeling the distribution of order flow that covers diverse scenarios. Moreover, linking a "macro" control target with every "micro" order separately may not make sense due to the low signal-to-noise ratio in order flow. Instead of fitting the distribution of raw order flow directly with a diffusion model, we design a two-stage model that exhibits greater efficiency.

Our model comprises of two modules. The first module is a meta controller $\mathcal{C}$ that learns the intraday dynamics of market states $\boldsymbol{x}$ regarding a scenario c, as the distribution $q(\boldsymbol{x}|c)$, using a conditional diffusion model. The second module is an order generator $\mathcal{G}$ that contains a simulated exchange and a meta agent. The meta agent is incorporated with financial economics prior, as well as guided by the meta controller, to generate order through a stochastic process. Figure 2 provides the overview of DiGA model and the model can be expressed as $\mathcal{M} = \{\mathcal{C}, \mathcal{G}\}$.

3.2.1 *Meta controller.*

For market states $\boldsymbol{x}$ to represent intraday dynamics regarding given scenario c, they should be evolving with time and be of close causal connection with c. We choose extracting the minutely mid-price return rate $\boldsymbol{r}$ as well as order arrival rate $\boldsymbol{\lambda}$ form the real order flow data as market states $\boldsymbol{x} = \{\boldsymbol{r}, \boldsymbol{\lambda}\}$.

While the number of trading minutes are fixed across trading days, it is natural to treat the stacked market states of each trading

Table 1: K-L divergence of stylized facts distribution between real and simulated order flow.

	A-Main				ChiNext			
Model	**MinR**	**RetAC**	**VolC**	**OIR**	**MinR**	**RetAC**	**VolC**	**OIR**
RFD	1.198	5.010	0.839	0.015	0.272	2.987	0.691	0.022
RMSC	2.640	10.170	1.237	0.563	1.371	7.461	0.668	0.588
LOBGAN	0.151	**1.903**	1.101	0.309	0.135	**1.711**	0.507	0.282
DiGA	**0.084**	2.781	**0.273**	**0.009**	**0.079**	1.997	**0.218**	**0.009**

day as a sample. Consequently, the training objective for fitting the distribution of the market states can be expressed as:

$$\min \mathbb{E}_{\boldsymbol{x}} \mathcal{D}\left(p_{\mathcal{G}}(\boldsymbol{x}) \parallel q(\boldsymbol{x})\right). \tag{3}$$

With the training set of market states $\{\boldsymbol{x} \sim q(\boldsymbol{x})\}$, we first generate the diffusion latent variable series with $\boldsymbol{x}_n = \sqrt{\bar{\alpha}_n}\boldsymbol{x}_0 + \sqrt{1-\bar{\alpha}_n}\boldsymbol{\epsilon}$, where $\boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathbf{I})$, n is the maximum diffusion step and $\bar{\alpha}_n$ is a transformation of diffusion variance schedule $\{\beta_n \in (0,1)\}_{n=1,\ldots,N}$. Then we adopt the $\boldsymbol{\epsilon}$-parameterized denoising diffusion probabilistic model (DDPM) [17] to predict noise from the noisy series $\boldsymbol{x}_n$ for a given denoising time step n, $\hat{\boldsymbol{\epsilon}} = \boldsymbol{\epsilon}_\theta(\boldsymbol{x}_n, n)$, where θ denotes model parameters. The corresponding training objective can be written as:

$$L_M := \mathbb{E}_{\mathcal{H}_e(\boldsymbol{O}), \boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}), n}[\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\boldsymbol{x}_n, n)\|^2]. \tag{4}$$

With sampling of $\tilde{\boldsymbol{x}}_0$ done iteratively using:

$$\tilde{\boldsymbol{x}}_{n-1} = \frac{1}{\sqrt{\alpha_n}}(\tilde{\boldsymbol{x}}_n - \frac{1-\alpha_n}{\sqrt{1-\bar{\alpha}_n}}\boldsymbol{\epsilon}_\theta(\tilde{\boldsymbol{x}}_n, n)) + \sigma_n \mathbf{z}, \tag{5}$$

from $\boldsymbol{x}_N \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ at time step N to $\boldsymbol{x}_0$ at time step 0, where $\boldsymbol{z} \sim \mathcal{N}(0, \mathbf{I})$ and $\sigma_n = \sqrt{\beta_n}$. With $\tilde{x}_0$, the meta controller is able to guide the meta agent in the order generator to generate orders.

To further exert order generation with control targets, we apply conditioning on the output of diffusion model and thus control the generated order flow. Following common practice [27], we implement conditional $\boldsymbol{\epsilon}$-parameterized noise predictor $\boldsymbol{\epsilon}_\theta(\boldsymbol{x}_n, n, c)$ for sampling with control target c.

Specifically, we adopt indicators commonly used to describe the state of financial markets as control targets. These indicators include daily return, amplitude, and volatility. All of these indicators can be numerically derived from return rates of the price series. Controlling these indicators allows the generated order flow to satisfy the need for analyzing markets under a wide range of specific scenarios that can be characterized by these indicators.

To align the control targets with the model, we introduce a target-specific feature extractor $\boldsymbol{\phi}$ that projects target indicators into latent representations $\boldsymbol{\phi}(\boldsymbol{c})$. We propose two types of condition encoders for exerting control. The first one is *discrete* control encoder, in which conditioning are mapped into a predefined number of discrete bins, with their indexes treated as class labels, and an embedding matrix is learned to extract the latent representation. The second type is *continuous* control encoder, where a fully connected network is employed to directly map conditions into latent representations. We train $\boldsymbol{\phi}$ concurrently with the conditional sampler and the training objective can then be written as follows:

$$L_C := \mathbb{E}_{\mathcal{H}_e(\boldsymbol{O}), \boldsymbol{c}, \boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}), n}[\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\boldsymbol{x}_n, n, \boldsymbol{\phi}(\boldsymbol{c}))\|^2]. \tag{6}$$

For both methods, we incorporate classifier-free guidance [18] to perform control. During the training phase, we jointly train unconditional and conditional samplers by randomly dropping out conditions. During sampling, a linear combination of conditional and unconditional score estimates is performed:

$$\tilde{\epsilon}_{\theta,\phi}(\boldsymbol{x}_n, n, \boldsymbol{c}) = (1-s)\boldsymbol{\epsilon}_\theta(\boldsymbol{x}_n, n) + s\boldsymbol{\epsilon}_\theta(\boldsymbol{x}_n, n, \boldsymbol{\phi}(\boldsymbol{c})), \tag{7}$$

where s is the conditioning scale that controls the strength of guidance. Sampling is done iteratively similar to Equation 5, where:

$$\boldsymbol{x}_{n-1} = \frac{1}{\sqrt{\alpha_n}}(\boldsymbol{x}_n - \frac{1-\alpha_n}{\sqrt{1-\bar{\alpha}_n}}\tilde{\epsilon}_{\theta,\phi}(\boldsymbol{x}_n, n, \boldsymbol{c})) + \sigma_n \mathbf{z}. \tag{8}$$

In practice, we adopt DDIM sampling [29] for better efficiency. As for the model backbone, we adopt a U-Net that is primarily built from 1D convolution layers, while sharing parameters across diffusion time steps. We refer the reader to Appendix B for the details of DDPM model.

3.2.2 Order generator. The order generator consists of a simulated exchange, and a meta agent. The simulated exchange replicates the double-action market protocol on which the majority of financial markets are operating. It facilitates the agent-market interaction and providing the basis of producing realistic financial market generations. The meta agent is the representative of all traders in the generated market, serving as a world agent. Different from existing works that has used learned agent as world agent, our meta agent is grounded with financial economics prior and is guided by the meta controller.

Specifically, the meta agent generated orders following a stochastic process, whose key parameters are determined by the meta controller. For every trading minute t in the trading day, the meta agent "wake up" by a time interval δ_i sampled from a exponential distribution $f(\delta_i; \lambda_t) = \lambda_t e^{-\lambda_t \delta_i}$, where i is the total number of executed wake-ups for this trading day and λ_t is given by the meta controller. Upon each wake-up, the meta agent generate an actor agent $\mathcal{A}_i$ within the family of heterogeneous agents [8], who makes decisions following the optimization of CARA utility function given the market observations. The order generation procedures are in Algorithm ?? and described as follows:

- The actor agent is initialized with random holding positions S and the corresponding amount of cash C, as well as the

Table 2: MSE between the targeted indicator and the generated aggregative statistics of generated order flow. Best results are highlighted with bold face.

		A-Main					ChiNext				
Target	**Method**	**Lower**	**Low**	**Medium**	**High**	**Higher**	**Lower**	**Low**	**Medium**	**High**	**Higher**
Return	No Control	1.443	0.583	0.529	0.813	2.337	0.979	0.684	0.992	1.718	3.923
	Discrete	1.055	0.494	0.228	0.429	0.664	1.285	0.807	0.243	**0.413**	0.869
	Continuous	**0.206**	**0.178**	**0.161**	**0.184**	**0.212**	**0.584**	**0.539**	**0.342**	0.449	**0.840**
Amplitude	No Control	0.521	0.268	0.268	0.699	3.298	1.130	0.638	0.427	0.608	2.763
	Discrete	**0.049**	0.088	0.309	0.502	0.930	**0.057**	0.134	0.346	0.523	**0.963**
	Continuous	0.054	**0.076**	**0.149**	**0.247**	**0.348**	0.110	**0.116**	**0.255**	**0.437**	0.973
Volatility	No Control	0.021	0.115	0.431	1.209	4.288	0.029	0.246	0.713	1.737	5.221
	Discrete	0.016	0.123	0.383	0.890	2.393	0.029	0.188	0.481	**0.948**	**2.257**
	Continuous	**0.011**	**0.104**	**0.318**	**0.774**	**2.389**	**0.028**	**0.178**	**0.473**	1.016	2.631

random weights g_f, g_c, g_n of its three heterogeneous components, namely fundamental, chartist and noise. These random variables are samples from independent exponential distribution, with configuration that the expectation of fundamental weight is highest among the three components.

- The actor agent then produce an estimation of objective future return $\hat{r}$ as the average of fundamental, chartist and noise with weights above. Fundamental is the r_t determined by the meta controller. Chartist is the historical average return $\bar{r}$ obtained from simulated exchange. Noise is a small Gaussian perturbation r_σ. Consequently, $\hat{r} = \frac{g_f r_t + g_c \bar{r} + g_n r_\sigma}{g_f + g_c + g_n}$.
- With the estimate return, the actor estimate future price $\hat{p}_t = p_t \exp(\hat{r})$, the actor agent is able to obtain its specific demand function $u(p) = \frac{\ln(\hat{p}_t/p)}{aVp}$, where a is risk averse coefficient and V is history price volatility, by deriving from CARA utility on future wealth [8], as well as the lowest order price p_l at which the demand function is satisfied $p_l(u(p_l) - S) = C$.
- Finally, the actor agent samples its order price uniformly between its lowest price and estimated price $p_i \sim \mathcal{U}(p_l, \hat{p})$, as well as obtain order volume $q_i = u(p_i) - S$ and order type $o_i = \text{sign}(q_i)$ where $o_i = 1$ indicates buy order and $o_i = 0$ indicates sell order.

The generated order is then recorded as $\boldsymbol{o}_i = (t_i, p_i, q_i, o_i) \sim p(\boldsymbol{o}|r_t, \lambda_t, \gamma)$, where $t_i = \sum_{j=1}^{i} \delta_i$ and γ is the set of fixed parameters for meta agent.

Generation ends at t_{max} when the next t_i will be greater than the total time lengths of trading hours of a trading day. The generated order flow is recorded as:

$$\tilde{\boldsymbol{O}} = \{\boldsymbol{o}_1, \ldots, \boldsymbol{o}_{max}\} \sim p(\boldsymbol{O}|\tilde{\boldsymbol{x}}, \gamma), \quad (9)$$

where $\tilde{\boldsymbol{x}}$ is generated by the meta controller.

4 Experiments

In this section, we present settings and results for experiments on real-world datasets to evaluate controllability and fidelity for DiGA. Additionaly, we show results on case study regarding the helpfulness of DiGA in a high-frequency trading reinforcement learning task.

4.1 Dataset and model configurations

We conduct the experiments on two tick-by-tick order datasets over China A-share market: *A-Main* and *ChiNext*. After preprocessing, 316,287 date-stock pair samples in A-Main and 122,574 in ChiNext are used. For each dataset, we take 5,000 samples each for validation and test, with all of the rest samples for training. For more dataset preprocessing details, please refer to Appendix A

We train the diffusion model on each dataset for 10 epochs, with 200 diffusion steps with AdamW optimizer. There are 256 samples per mini-batch and the learning rate is $1e^{-5}$. For both the discrete and continuous control models, we use a probability of 0.5 to randomly drop conditions during training. We set a pseudo initial stock price at 10 for every generation run. For more parameter settings details, please refer to Appendix B.3.

4.2 Evaluation on controlling financial market generation

In this experiment, we evaluate DiGA's capability for controlling market generation. For enabling DiGA to simulate with control targets as input, we train DiGA with four indicators respectively: *return, amplitude* and *volatility*, all of which are indicators that represent a vast range of market scenarios. For each indicator, we first retrieve its empirical distribution from real-world order flow dataset. Then we partition the values into five bins according to uniform percentiles, each representing *lower, low, mid, high* and *higher* cases for the scenario characterized by corresponding indicator. We train DiGA with discrete control encoder using these case types as class labels, and train DiGA with continuous control encoder using the exact value of indicators after normalization. On testing, the class labels are directly used as control target for DiGA with discrete control encoder, and we extract the median value of real samples from each bin to represent corresponding scenario as the control target for DiGA with continuous control encoder.

Table 2 shows the **mean squared error (MSE)** between the indicators computed from simulated order flow and the control target, i.e. the median value of each of the five bins, for DiGA with both discrete and continuous control encoder. The results are averaged from 3 runs with different random seeds. The row "No Control" presents the results from a variant of DiGA by removing the condition mechanism in meta controller, whose generations are independent to the control target. From Table 2, DiGA successfully achieves the effect of control by keeping a relatively small error between realized indicator value and control target, while the method without control tend to be either random or repeating some particular scenarios.

Figure 3(a) illustrates the mid-price series of controlled generation samples for each scenario, as well as the distribution of indicators in 200 independent generations runs for each scenario. Results show that the sampled price curves successfully represent the desired scenario, and the distribution of the four indicators are shifted correctly following the control target. This demonstrates capability of DiGA to control financial market generation.

4.3 Evaluation on generation fidelity

We evaluate generation fidelity of DiGA and compare the results with market simulation method baselines with both rule-based agents and learning-based agents:

- **RFD** [32] is a multi-agent based market simulation configuration with random fundamental and diverse agent types, consisting of 1 market maker, 25 momentum agents, 100 value agents and 5,000 noise agents.
- **RMSC** [1] is the reference market simulation configuration introduced with the ABIDES-gym, which contains all RFD agents and 1 extra percentage-of-volume (POV) agent who provide extra liquidity to the simulated market.
- **LOBGAN** [10] trains a Conditional Generative Adversarial Network with to generate next order conditioned on market history.

We focus on the distribution discrepancy of several "stylized facts", which are statistics regarding asset returns and order book, between the real and simulated markets. The selected statistics, as listed below, are among the most representative features in the domain of the financial market micro-structure:

- **Minutely Log Returns (MinR)** are the log difference between two consecutive prices sampled by minutes.
- **Return Auto-correlation (RetAC)** is the value of linear auto-correlation function calculated between the return array and its lagged array. Empirical studies on real market data have discovered absence of auto-correlation while the lag is not big enough.
- **Volatility Clustering (VolC)** is the value of linear auto-correlation function of the squared returns and their lag. It shows the empirical fact that volatile events tend to appear in cluster with time.
- **Order Imbalance Ratio (OIR)** is proportion difference between best bid volume and best ask volume. It represents the tendency for trading of market participants.

We demonstrate results sampled with the unconditional version of our method DiGA in the experiment. Figure 3(b) shows the results for fidelity comparison, where the distribution of statistics on real market data is displayed as *Real* in golden solid line. Overall, DiGA stays the closest to *Real* compared with other methods. We capture the differences between real and simulated market quantitatively using **Kullback-Leibler divergence (K-L)**. Table 1 provides thequantitative metrics, showing that DiGA can reach the best K-L divergence among the most of the statistics. Although LOBGAN obtains the lowest RetAC divergence because of its auto-regressive nature when generating orders, DiGA still outperform LOBGAN on other metrics by a large margin. Overall, the results demonstrate DiGA's superiority on generating realistic market dynamic, achieving the state-of-the-art performance regarding fidelity in market simulation.

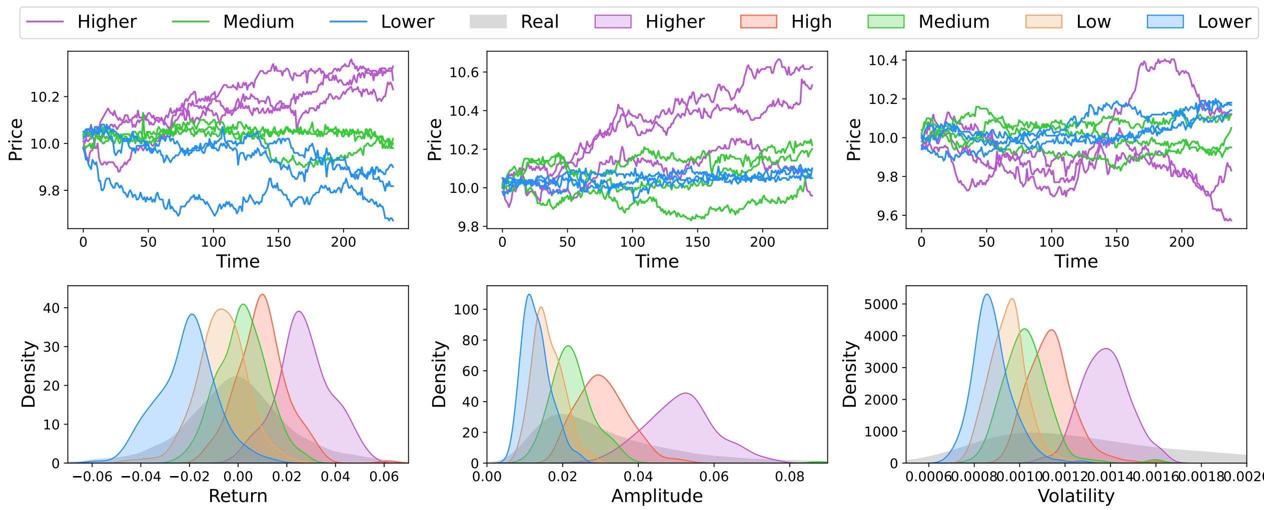
4.4 Evaluation for downstream task: high-frequency trading with reinforcement learning

We evaluate the helpfulness of DiGA as the environment for training reinforcement learning (RL) algorithms to perform high-frequency trading.

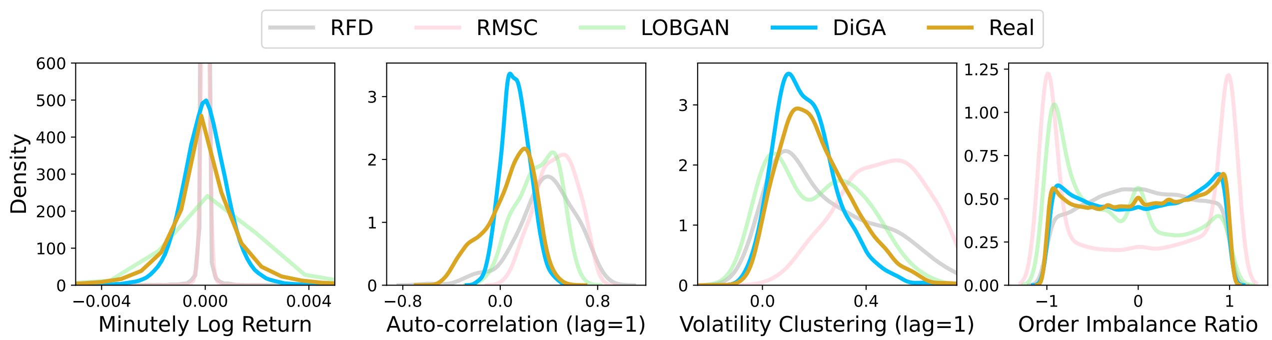
4.4.1 Settings. We train a trading agent with simulated market as training environment using the A2C algorithm, to optimize for a high-frequency trading task. Agents take an discrete action every 10 seconds. Possible actions include either buy or sell at any one of the best 5 level with integer volume ranging from 1 to 10 units, as well as an option not to submit any order. The observation space is configured as the price changes of last 20 seconds, 10-level bid-ask price-volume pairs and the account status including the current amount of capital, position and cash of agents. Each agent is trained on simulated stock market produced by either history replay, RFU, DiGA and a variant of DiGA that removes the conditioning mechanism of meta controller (DiGA-c). Each train lasts for 200 episodes, where each episode represents a full trading day with four trading hours. Afterwards, we test the agents with an environment replaying out-of-sample real market data for 50 episodes. Tests are repeated three times using three non-overlap periods of out-of-sample real market data. All trains share the same set of RL hyper-parameters for fairness.

We evaluate the performance of RL trading task with four metrics. **Daily return (Ret)** is the mean return rate across episodes, for assessing profitability. **Daily volatility (Vol)** is the standard deviation of daily return for assessing risk management, daily **Sharpe ratio (SR)** is the division of daily return by volatility for assessing return-risk trade-off ability. **Maximum drawdown (MDD)** is the largest intraday profit drop happens during testing which assesses the performance under extreme cases.

4.4.2 Results. Table 3 displays the average numerical results for the high-frequency trading task. The trading agent trained with DiGA generated environment has demonstrated the highest daily return, daily Sharpe ratio and maximum drawdown against all baselines, and the near best performance on daily volatility. These results shows that the DiGA generated environment has help the trading agent learn a better policy. Moreover, the inferior results obtained



(a)



(b)

Figure 3: Experiment results. (a) Aggregated price curves demos (first row) and the distribution of targeted indicators (second row). For the first row, each curve represents the order flow of one day. For the second row, each colored density represents the distribution of targeted indicator computed from generation results. (b) Comparison of stylized facts distribution across baselines. The x-axis is the stylized fact values and the y-axis is the density.

with DiGA-c indicates the importance of performing control on the training environment for a more sufficient exploration of trading agent.

We further conduct a case study on the behavior of trading agents trained on different simulated environments to investigate possible reason for the outperforming results of agents trained with DiGA. The case shows a trading day representing an extreme volatile scenario with sharp price surge and drop of up to five percents happens in minutes. The return rate of different agents in the same trading day is shown in Figure 4. It is shown that the agent trained with market replay environment learns a passive investment strategy that follows the changes of asset price very closely. The agent trained with RFU environment tries to catch the trend, but suffers from a sudden drop that reverts the trend. The agent trained with DiGA-c fails to take action in this extreme market. In contrast, the agent trained with DiGA has avoided the extreme event and trades to take profits afterwards. The result showcases the potential that the diversed generated scenario from DiGA has provided experience for agent to learn making better action in these cases.

5 Conclusion and future work

In this paper, we present the problem formulation of controllable financial market generation, and propose a *Diffusion Guided meta Agent* (DiGA) model to address the problem. Specifically, we utilize a diffusion model to capture dynamics of market state represented by time-evolving distribution parameters about mid-price return rate and order arrival rate, and define a meta agent with financial economic priors to generate orders from the corresponding distributions. Extensive experimental results demonstrate that our method exhibits outstanding controllability and fidelity in simulation. While we focus on generating order flow of one individual stock for each time in this work, one future work is to consider the correlation among multiple assets for generating more realistic markets.

Table 3: Full out-of-sample test results (in percentage). Best results are highlighted with bold face.

Environment	Period	Ret(%)(↑)	Vol(↓)	SR(↑)	MDD(%)(↑)
Replay	1	0.031	0.502	0.012	-1.198
	2	0.036	0.323	0.023	-0.936
	3	-0.040	0.413	0.007	-1.264
	Average	$0.009_{\pm 0.043}$	$0.413_{\pm 0.090}$	$0.014_{\pm 0.008}$	$-1.133_{\pm 0.173}$
RFD	1	-0.004	0.265	-0.012	-1.004
	2	0.009	0.084	0.044	-0.426
	3	-0.005	0.128	0.001	-0.980
	Average	$0.000_{\pm 0.008}$	$0.159_{\pm 0.094}$	$0.011_{\pm 0.029}$	$-0.803_{\pm 0.327}$
DiGA-c	1	0.037	0.320	0.025	-1.250
	2	0.017	0.074	0.060	-0.472
	3	-0.009	0.046	-0.068	-0.422
	Average	$0.015_{\pm 0.023}$	$\mathbf{0.147_{\pm 0.151}}$	$0.006_{\pm 0.066}$	$\mathbf{-0.715_{\pm 0.464}}$
DiGA	1	0.033	0.550	0.023	-1.488
	2	0.046	0.330	0.084	-1.188
	3	0.009	0.352	0.040	-1.264
	Average	$\mathbf{0.029_{\pm 0.019}}$	$0.411_{\pm 121}$	$\mathbf{0.049_{\pm 0.031}}$	$-1.313_{\pm 0.156}$

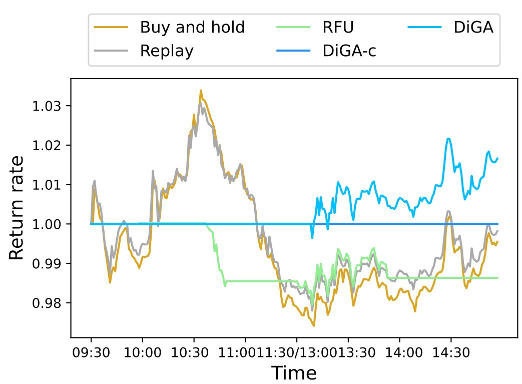


Figure 4: The return rate of different agents in the same trading day. The x-axis denotes time frame in minutes. The y-axis denotes the cumulative return rate obtained by the trading agent.

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 ACM International Conference on AI in Finance.*

[2] Marco Aversa, Gabriel Nobis, Miriam Hägele, Kai Standvoss, Mihaela Chirica, Roderick Murray-Smith, Ahmed M. Alaa, Lukas Ruff, Daniela Ivanova, Wojciech Samek, Frederick Klauschen, Bruno Sanguinetti, and Luis Oala. 2023. DiffInfinite: Large Mask-Image Synthesis via Parallel Random Patch Diffusion in Histopathology. In *Advances in Neural Information Processing Systems.* 78126–78141.

[3] 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? *CoRR* abs/1906.12010 (2019).

[4] Tim Brooks, Bill Peebles, Connor Holmes, Will DePue, Yufei Guo, Li Jing, David Schnurr, Joe Taylor, Troy Luhman, Eric Luhman, Clarence Ng, Ricky Wang, and Aditya Ramesh. 2024. Video generation models as world simulators. (2024). https://openai.com/research/video-generation-models-as-world-simulators

[5] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language Models are Few-Shot Learners. In *Advances in Neural Information Processing Systems.* 1877–1901.

[6] David Byrd, Maria Hybinette, and Tucker Hybinette Balch. 2020. ABIDES: Towards High-Fidelity Multi-Agent Market Simulation. In *Proceedings of the 2020 ACM SIGSIM Conference on Principles of Advanced Discrete Simulation.*

[7] Carl Chiarella and Giulia Iori. 2002. A simulation analysis of the microstructure of double auction markets. *Quantitative finance* 2, 5 (2002), 346.

[8] Carl Chiarella, Giulia Iori, and Josep Perello. 2009. The Impact of Heterogeneous Trading Rules on the Limit Order Book and Order Flows. *Journal of Economic Dynamics and Control* 33, 3 (2009), 525–537.

[9] Andrea Coletta, Sriram Gopalakrishnan, Daniel Borrajo, and Svitlana Vyetrenko. 2023. On the Constrained Time-Series Generation Problem. In *Advances in Neural Information Processing Systems.* 61048–61059.

[10] Andrea Coletta, Joseph Jerome, Rahul Savani, and Svitlana Vyetrenko. 2023. Conditional Generators for Limit Order Book Environments: Explainability, Challenges, and Robustness. *CoRR* abs/2306.12806 (2023).

[11] Andrea Coletta, Aymeric Moulin, Svitlana Vyetrenko, and Tucker Balch. 2022. Learning to simulate realistic limit order book markets from data as a World Agent. In *Proceedings of the ACM International Conference on AI in Finance.*

[12] 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 ACM International Conference on AI in Finance.*

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

[14] Prafulla Dhariwal and Alexander Quinn Nichol. 2021. Diffusion Models Beat GANs on Image Synthesis. In *Proceedings of Advances in Neural Information Processing Systems*.

[15] Jian Guo, Saizhuo Wang, Lionel M. Ni, and Heung-Yeung Shum. 2023. Quant 4.0: Engineering Quantitative Investment with Automated, Explainable and Knowledge-driven Artificial Intelligence. *CoRR* abs/2301.04020 (2023).

[16] Yuwei Guo, Ceyuan Yang, Anyi Rao, Zhengyang Liang, Yaohui Wang, Yu Qiao, Maneesh Agrawala, Dahua Lin, and Bo Dai. 2024. AnimateDiff: Animate Your Personalized Text-to-Image Diffusion Models without Specific Tuning. *International Conference on Learning Representations* (2024).

[17] Jonathan Ho, Ajay Jain, and Pieter Abbeel. 2020. Denoising Diffusion Probabilistic Models. In *Proceedings of Advances in Neural Information Processing Systems*.

[18] Jonathan Ho and Tim Salimans. 2022. Classifier-Free Diffusion Guidance. *CoRR* abs/2207.12598 (2022).

[19] Marcel Kollovieh, Abdul Fatir Ansari, Michael Bohlke-Schneider, Jasper Zschiegner, Hao Wang, and Yuyang (Bernie) Wang. 2023. Predict, Refine, Synthesize: Self-Guiding Diffusion Models for Probabilistic Time Series Forecasting. In *Advances in Neural Information Processing Systems*. 28341–28364.

[20] Zhifeng Kong, Wei Ping, Jiaji Huang, Kexin Zhao, and Bryan Catanzaro. 2021. DiffWave: A Versatile Diffusion Model for Audio Synthesis. In *Proceedings of the International Conference on Learning Representations*.

[21] Junyi Li, Xintong Wang, Yaoyang Lin, Arunesh Sinha, and Michael P. Wellman. 2020. Generating Realistic Stock Market Order Streams. In *Proceedings of the AAAI Conference on Artificial Intelligence, AAAI*.

[22] Haohe Liu, Zehua Chen, Yi Yuan, Xinhao Mei, Xubo Liu, Danilo Mandic, Wenwu Wang, and Mark D Plumbley. 2023. AudioLDM: Text-to-Audio Generation with Latent Diffusion Models. In *Proceedings of the International Conference on Machine Learning*.

[23] Thomas Lux and Michele Marchesi. 1999. Scaling and criticality in a stochastic multi-agent model of a financial market. *Nature* 397, 6719 (1999), 498–500.

[24] Takanobu Mizuta. 2020. An agent-based model for designing a financial market that works well. In *Proceedings of the IEEE Symposium Series on Computational Intelligence*.

[25] R.G. Palmer, W. Brian Arthur, John H. Holland, Blake LeBaron, and Paul Tayler. 1994. Artificial economic life: a simple model of a stockmarket. *Physica D: Nonlinear Phenomena* 75, 1 (1994), 264–274.

[26] Marco Raberto, Silvano Cincotti, Sergio M. Focardi, and Michele Marchesi. 2001. Agent-based simulation of a financial market. *Physica A: Statistical Mechanics and its Applications* 299, 1 (2001), 319–327.

[27] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. 2022. High-Resolution Image Synthesis with Latent Diffusion Models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

[28] Jascha Sohl-Dickstein, Eric A. Weiss, Niru Maheswaranathan, and Surya Ganguli. 2015. Deep Unsupervised Learning using Nonequilibrium Thermodynamics. In *Proceedings of the International Conference on Machine Learning*.

[29] Jiaming Song, Chenlin Meng, and Stefano Ermon. 2021. Denoising Diffusion Implicit Models. In *Proceedings of the International Conference on Learning Representations1*.

[30] Jiaming Song, Qinsheng Zhang, Hongxu Yin, Morteza Mardani, Ming-Yu Liu, Jan Kautz, Yongxin Chen, and Arash Vahdat. 2023. Loss-Guided Diffusion Models for Plug-and-Play Controllable Generation. In *Proceedings of the International Conference on Machine Learning*.

[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. In *Proceedings of the International Conference on Learning Representations*.

[32] Svitlana Vyetrenko, David Byrd, Nick Petosa, Mahmoud Mahfouz, Danial Dervovic, Manuela Veloso, and Tucker Balch. 2020. Get real: realism metrics for robust limit order book market simulations. In *Proceedings of the ACM International Conference on AI in Finance*.

[33] Xintong Wang and Michael P. Wellman. 2017. Spoofing the Limit Order Book: An Agent-Based Model. In *Proceedings of the Conference on Autonomous Agents and MultiAgent Systems*.

[34] Xinyu Yuan and Yan Qiao. 2024. Diffusion-TS: Interpretable Diffusion for General Time Series Generation. In *International Conference on Learning Representations*.

A Detailed description of dataset and preprocessing

We conduct the experiments on two tick-by-tick order datasets over China A-share market: *A-Main* and *ChiNext*. Both datasets are collected from Wind[1], retrieving Shenzhen Stock Exchange (SZSE) of year 2020. Table 4 summarizes the statistics of the dataset.

Table 4: Dataset statistics for DiGA.

	A-Main	ChiNext
Number of date-stock pairs	316,287	122,574
Number of unique stocks	1452	854
Number of unique trading days	237	231

For each dataset, we randomly draw 5,000 samples each for validation and test, with all of the rest samples for training.

Preprocessing includes filtering and transformation.

- **Filtering** We filtered out samples that contains incomplete records (i.e. trading suspended) or invalid orders (i.e. invalid order price).
- **Transformation** We transform tick-by-tick data into market states represented with mid-price return and order arrival rate.

  For mid-price return, we first extract minutely mid-price series from order flow samples of each trading day, where mid-price $p_t = (a_{t,1} + b_{t,1})/2$ is defined as the average of best ask price $b_{t,1}$ and best bid price $a_{t,1}$ at the end of the t-th trading minute. Then we calculate the log differences between consecutive minutes as the mid-price return rate $r_t = \log(p_t) - \log(p_{t-1})$. Finally, $\boldsymbol{r} = [r_1, r_2, ..., r_T]$. The effective T for mid-price return is 236, excluding the call auction phase at the end of each trading day.

  For order arrival rate, we first calculate the number or orders within each minutes as N_t. With the assumption that the arrival of order follows a Poisson process, we take N_t as the expected number of orders and the order arrival rate can be approximate by $\lambda_t = N_t$. Finally, $\boldsymbol{\lambda} = [\lambda_1, \lambda_2, ..., \lambda_T]$.

  During training, the inputs are transformed by z-score normalization method with mean and standard deviation calculated from the training split of data. When sampling, model outputs are inverse transformed accordingly.

B Detailed description of DDPM

B.1 Brief review of DDPM model

A diffusion probabilistic model [28] learns to reverse the transitions of a Markov chain which is known as the diffusion process that gradually adds noise to data, ultimately destroying the signal.

Let $\mathbf{x}_0 \in \mathbb{R}^d \sim q(\mathbf{x}_0)$ be real data of dimension d from space $\mathcal{X}$. The diffusion process generates $\mathbf{x}_1, ..., \mathbf{x}_N$ from the same space with the same shape as $\mathbf{x}_0$, using a Markov chain that adds Gaussian noise over N time steps: $q(\mathbf{x}_1, ..., \mathbf{x}_N|\mathbf{x}_0) := \prod_{n=1}^{N} q(\mathbf{x}_n|\mathbf{x}_{n-1})$. The transition kernel is commonly defined as:

$$q(\mathbf{x}_n|\mathbf{x}_{n-1}) := \mathcal{N}(\mathbf{x}_n; \sqrt{1-\beta_n}\mathbf{x}_n, \beta_n\mathbf{I}), \tag{10}$$

where $\{\beta_n \in (0,1)\}_{n=1,...,N}$ defines the variance schedule. Note that $\mathbf{x}_n$ at any arbitrary time step n can be derived in a closed form $q(\mathbf{x}_n|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_n; \sqrt{\bar{\alpha}_n}\mathbf{x}_0, (1-\bar{\alpha}_n)\mathbf{I})$, where $\alpha_n := 1 - \beta_n$ and $\bar{\alpha}_n := \prod_{s=1}^{n} \alpha_s$. For the reverse process, the diffusion model, parameterized by θ, yields:

$$p_\theta(\mathbf{x}_0, \mathbf{x}_1, ..., \mathbf{x}_N) := p(\mathbf{x}_N) \prod_{n=1}^{N} p_\theta(\mathbf{x}_{n-1}|\mathbf{x}_n), \tag{11}$$

where $p_\theta(\mathbf{x}_{n-1}|\mathbf{x}_n) := \mathcal{N}(\mathbf{x}_{n-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_n, n), \Sigma_\theta(\mathbf{x}_n, n))$ and the transitions start at $p(\mathbf{x}_N) = \mathcal{N}(\mathbf{x}_n; \mathbf{0}, \mathbf{I})$.

While the usual optimization objective can be written as:

$$L := \mathbb{E}[-\log p_\theta(\mathbf{x}_0)] \leq \mathbb{E}_q[-\log \frac{p_\theta(\mathbf{x}_0, ..., \mathbf{x}_N)}{q(\mathbf{x}_1, ..., \mathbf{x}_N|\mathbf{x}_0)}], \tag{12}$$

a widely adopted parameterization writes:

$$\boldsymbol{\mu}_\theta(\mathbf{x}_n, n) = \frac{1}{\sqrt{\alpha_n}}(\mathbf{x}_n - \frac{\beta_n}{\sqrt{1-\bar{\alpha}_n}}\boldsymbol{\epsilon}_\theta(\mathbf{x}_n, n)), \tag{13}$$

which simplifies the objective to:

$$L_{simple} := \mathbb{E}_{\mathbf{x}_0, \boldsymbol{\epsilon}, n}[\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_n}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_n}\boldsymbol{\epsilon}, n)\|^2]. \tag{14}$$

On sampling, $\mathbf{x}_{n-1} = \frac{1}{\sqrt{\alpha_n}}(\mathbf{x}_n - \frac{1-\alpha_n}{\sqrt{1-\bar{\alpha}_n}}\boldsymbol{\epsilon}_\theta(\mathbf{x}_n, n)) + \sigma_n\mathbf{z}$, where $\sigma_n = \sqrt{\beta_n}$ and $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ [17].

To obtain a conditional DDPM model using classifier-free guidance [18], we modify the noise estimator to receive condition embedding $\boldsymbol{\phi}(\boldsymbol{c})$ as input, forming $\boldsymbol{\epsilon}_\theta(\mathbf{x}_n, n, \boldsymbol{\phi}(\boldsymbol{c}))$. During training, condition c is replaced by an unconditional identifier $\boldsymbol{c}_0$ by a probability p_{uncond} to obtain unconditional prediction $\boldsymbol{\epsilon}_\theta(\mathbf{x}_n, n) = \boldsymbol{\epsilon}_\theta(\mathbf{x}_n, n, \boldsymbol{c}_0)$. During sampling, a conditioning scale s is set to control the strength of guidance, replacing the noise prediction with

$$\tilde{\boldsymbol{\epsilon}}_{\theta,\phi}(\boldsymbol{x}_n, n, \boldsymbol{c})) = (1-s)\boldsymbol{\epsilon}_\theta(\boldsymbol{x}_n, n) + s\boldsymbol{\epsilon}_\theta(\boldsymbol{x}_n, n, \boldsymbol{\phi}(\boldsymbol{c})). \tag{15}$$

Both conditional and unconditional sampling can be accelerated by DDIM [29].

B.2 Algorithmic procedure for training meta controller

Algorithm 1: Training meta controller of DiGA

Data: Order flow $\boldsymbol{O}$ processed into market states $\boldsymbol{x}$;
Result: Network parameters θ for meta controller
repeat
  Sample $\boldsymbol{x}_0 \sim q(\boldsymbol{x})$;
  Calculate target indicator $c = \mathcal{F}(\boldsymbol{x})$;
  Randomly set c as unconditional identifier c_u;
  Randomly sample time step $n \sim \mathcal{U}(1, N)$;
  Randomly sample noise $\boldsymbol{\epsilon} \sim \mathcal{N} \sim (\boldsymbol{0}, \boldsymbol{I})$;
  Corrupt data $\boldsymbol{x}_n = \sqrt{\bar{\alpha}_n}\boldsymbol{x} + \sqrt{1-\bar{\alpha}_n}\boldsymbol{\epsilon}$;
  Take gradient descent step on: $\nabla_{\theta,\phi}\|\boldsymbol{\epsilon} - \tilde{\boldsymbol{\epsilon}}_{\theta,\phi}(\boldsymbol{x}_n, n, \boldsymbol{c}))\|$;
until *reach max epochs*;

[1] https://www.wind.com.cn/

B.3 Detailed parameters of training meta controller

The denoising model ϵ_θ used in meta controller is adapted from [17]: The input of denoising model is shaped as (B, C, T), where B is the batch size, C is the number of channels and T is the number of trading minutes in a day. In our case, $C = 2$ and $T = 236$.

The denoising model is structured as a U-Net mainly with 3 down-sampling blocks, 1 middle blocks, 3 up-sampling blocks and 1 output block. Each up/down-sampling block contains 2 ResNet blocks, 1 self-attention layer and 1 up/down sample layer. Each ResNet block contains 2 convolution layers of size 15, with SiLU activation, with residual connection and layer normalization. Each up/down sampling layer use a factor of 2. The number of channels after each up/down sampling starts from 64 and is then scaled by a factor of 4. The middle block contains 2 ResNet block with a self-attention layer in between. The output block is another ResNet block followed by an 1*1 convolution layer. In addition, the conditioning embedding is extracted by a fully connected network with 2 layers of dimension 64. Table 5 lists the required parameters along with the above description.

The model is trained on 1 NVIDIA Tesla V100 GPU. Training 10 epochs takes approximately 2 hours for the A-Main dataset and 1 hour for the ChiNext dataset.

Table 5: Detailed parameters used for meta controller training.

Parameter name	Parameter value
Denoising shape	(2, 236)
Diffusion steps	200
Residual blocks	2
First layer hidden dimension	64
U-Net dimension multipliers	(1, 4, 16)
Embedding dimensions	256
Convolution kernel size	15
Convolution padding length	7
Unconditional probability p_{uncond}	0.5
Batch size	256
Learning rate	1e-5

C Details of generating with DiGA

C.1 Algorithmic procedure for generating financial market with DiGA.

We discuss the input parameters in C.2 and C.3.

C.2 Discussion on meta controller sampling parameters

When sampling with the meta controller, we apply DDIM [29] to reduce sampling steps to 20. For obtaining best generation result, the classifier guidance scale s should be properly selected. In our experiments, we select the best s from the range of 1, 2, 4, 6, 8 for each target scenario. The selection is based on the discrepancy between control target and the generated scenario during training, and we take the s that shows the lowest discrepancy.

Algorithm 2: Generating market order flow with DiGA

Input: Meta controller parameter θ, control target c, conditioning scale s, max trading time T, initial price p_0, meta agent parameters $\lambda_f, \lambda_c, \lambda_n, \tau_0, \alpha_0, \sigma_n$, simulated exchange
Output: Order flow $\boldsymbol{O}$
(Phase 1: sampling market states with meta controller)
Random sample $\boldsymbol{x}_n \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$;
for $n = N$ **to** 1 **do**
  Sample $\boldsymbol{z} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$;
  $\hat{\boldsymbol{\epsilon}} = \tilde{\boldsymbol{\epsilon}}_{\theta,\phi}(\boldsymbol{x}_n, n, \boldsymbol{c})) = (1-s)\boldsymbol{\epsilon}_\theta(\boldsymbol{x}_n, n) + s\boldsymbol{\epsilon}_\theta(\boldsymbol{x}_n, n, \boldsymbol{\phi}(\boldsymbol{c})))$;
  $\boldsymbol{x}_{n-1} = \frac{1}{\sqrt{\alpha_n}}(\boldsymbol{x}_n - \frac{1-\alpha_n}{\sqrt{1-\bar{\alpha}_n}}\hat{\boldsymbol{\epsilon}}) + \sigma_n \boldsymbol{z}$;
end
(Phase 2: generate order flow with meta agent)
Initilize $t = 0$, $\boldsymbol{O} = \emptyset$, $p_t = p_0$;
repeat
  Extract r_t and λ_t from $\boldsymbol{x}$;
  Initialize asset for actor agent $\mathcal{A}_i$: $S_i \sim \text{exponential}(S_0)$, $C_i \sim \text{exponential}(C_0)$,
  $g_f \sim \text{Laplace}(\lambda_f)$, $g_c \sim \text{Laplace}(\lambda_c)$, $g_n \sim \text{Laplace}(\lambda_n)$, $\tau_i = \tau_0 \frac{1+g_f}{1+g_c}$, $\alpha_i = \alpha_0 \frac{1+g_f}{1+g_c}$;
  Sample time interval $\delta_i \sim \text{exponential}(\lambda)$, set $t_i = t + \delta_i$;
  Observe $\bar{r}$ from exchange and sample $r_\sigma \sim \mathcal{N}(0, \sigma_n)$;
  Estimate future return $\hat{r} = \frac{g_f r_t + g_c \bar{r} + g_n r_\sigma}{g_f + g_c + g_n}$;
  Estimate future price $\hat{p}_t = p_t \exp(\hat{r})$;
  Solve p_l from $p_l(u(p_l) - S_i) = C_i$, where $u(p) = \frac{\ln(\hat{p}_t/p)}{\alpha_i V p}$;
  Sample price $p_i \sim \mathcal{U}(p_l, \hat{p})$;
  Calculate volume $q_i = u(p_i) - S_i$;
  Obtain order type $o_i = \text{sign}(q_i)$;
  Compose order $\boldsymbol{o}_i = (t_i, p_i, q_i, o_i)$;
  Simulated exchange update current price $p_t = \text{Exchange}(\boldsymbol{o}_i)$;
  Update $t = t_i$, $\boldsymbol{O} = \boldsymbol{O} \cup \{o_i\}$;
until $t > T$;

C.3 Discussion on meta agent parameters

Following the heterogeneous agent settings [8], the meta agent employs several probabilistic parameters to ensure heterogeneity. The parameters and their usage are as follows: λ_f for fundamental weight, λ_c for chartist weight, λ_n for noise weight, τ_0 for estimation horizon, α_0 for risk aversion, σ_n for noisy return. All the parameters are for mimicking human preference in real stock market.

Throughout our experiments, we fix $\lambda_f = 10, \lambda_c = 1.5, \lambda_n = 1, \tau_0 = 30, \alpha_0 = 0.1, \sigma_n = 1e^{-4}, p_0 = 10$ for all runs to avoid overfitting these parameters. Nevertheless, they can be calibrated to further improve fidelity.

D Additional results

We provide the full result table with both mean and std across 3 independent run with different random seeds for main tables as below. These results could confirm that the advantage of DiGA is significant.

Table 6: MSE between the targeted indicator and the generated aggregative statistics of generated order flow. Best results are highlighted with bold face.

			A-Main					ChiNext				
Target	**Method**		**Lower**	**Low**	**Medium**	**High**	**Higher**	**Lower**	**Low**	**Medium**	**High**	**Higher**
Return	No Control	mean	1.443	0.583	0.529	0.813	2.337	0.979	0.684	0.992	1.718	3.923
		std	0.211	0.096	0.048	0.072	0.201	0.134	0.095	0.103	0.133	0.209
	Discrete	mean	1.055	0.494	0.228	0.429	0.664	1.285	0.807	0.243	**0.413**	0.869
		std	0.159	0.040	0.003	0.027	0.031	0.008	0.055	0.023	0.029	0.110
	Continuous	mean	**0.206**	**0.178**	**0.161**	**0.184**	**0.212**	**0.584**	**0.539**	**0.342**	0.449	**0.840**
		std	0.080	0.023	0.022	0.017	0.033	0.211	0.267	0.389	0.422	0.522
Amplitude	No Control	mean	0.521	0.268	0.268	0.699	3.298	1.130	0.638	0.427	0.608	2.763
		std	0.071	0.044	0.017	0.024	0.105	0.074	0.077	0.081	0.088	0.107
	Discrete	mean	**0.049**	0.088	0.309	0.502	0.930	**0.057**	0.134	0.346	0.523	**0.963**
		std	0.005	0.004	0.016	0.016	0.053	0.003	0.008	0.002	0.031	0.015
	Continuous	mean	0.054	**0.076**	**0.149**	**0.247**	**0.348**	0.110	**0.116**	**0.255**	**0.437**	0.973
		std	0.017	0.016	0.017	0.080	0.011	0.007	0.011	0.045	0.065	0.113
Volatility	No Control	mean	0.021	0.115	0.431	1.209	4.288	0.029	0.246	0.713	1.737	5.221
		std	0.003	0.018	0.038	0.065	0.123	0.006	0.015	0.034	0.061	0.116
	Discrete	mean	0.016	0.123	0.383	0.890	2.393	0.029	0.188	0.481	**0.948**	**2.257**
		std	0.001	0.017	0.008	0.021	0.008	0.003	0.007	0.014	0.016	0.019
	Continuous	mean	**0.011**	**0.104**	**0.318**	**0.774**	**2.389**	**0.028**	**0.178**	**0.473**	1.016	2.631
		std	0.001	0.010	0.030	0.057	0.098	0.005	0.008	0.020	0.078	0.260

Table 7: K-L divergence of stylized facts distribution between real and simulated order flow.

		A-Main				ChiNext			
Model		**MinR**	**RetAC**	**VolC**	**OIR**	**MinR**	**RetAC**	**VolC**	**OIR**
RFD	mean	1.198	5.010	0.839	0.015	0.272	2.987	0.691	0.022
	std	0.083	1.335	0.322	0.001	0.037	0.937	0.316	0.001
RMSC	mean	2.640	10.170	1.237	0.563	1.371	7.461	0.668	0.588
	std	0.051	0.204	0.959	0.016	0.001	0.195	0.243	0.016
LOBGAN	mean	0.151	**1.903**	1.101	0.309	0.135	**1.711**	0.507	0.282
	std	0.007	1.045	0.639	0.011	0.002	1.043	0.108	0.010
DiGA	mean	**0.084**	2.781	**0.273**	**0.009**	**0.079**	1.997	**0.218**	**0.009**
	std	0.006	0.417	0.020	0.001	0.002	0.243	0.049	0.001