Controllable Financial Market Generation with Diffusion Guided Meta Agent

Yu-Hao Huang[1], **Chang Xu**[2], **Yang Liu**[2], **Weiqing Liu**[2], **Wu-Jun Li**[1], **Jiang Bian**[2]
[1]National Key Laboratory for Novel Software Technology,
School of Computer Science, Nanjing University
[2]Microsoft Research Asia
huangyh@smail.nju.edu.cn, {chanx, yangliu2, weiqing.liu}@microsoft.com,
liwujun@nju.edu.cn, jiang.bian@microsoft.com

Abstract

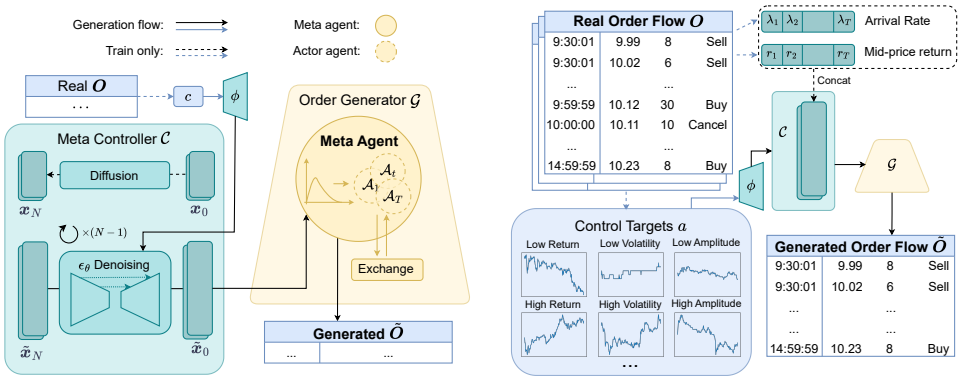
Generative modeling has transformed many fields like language and visual modeling, while its exploit in financial market is under-explored. As the minimal unit within a financial market is an order, order flow modeling represents the fundamental generative financial task. However, current approaches 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 market generation, and propose a *Diffusion Guided meta Agent* (DiGA) model. 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.

1 Introduction

Generative modeling has transformed fields such as natural language processing [1, 2, 3], media synthesis [4, 5, 6], science discovery [7, 8, 9, 10] and medical applications [11, 12]. Similar to word for language and pixel for images, order is the fundamental element representing a minimal unit of event to generate within financial market [13, 14, 15, 16, 17]. Recent works have attempted to simulate order-level financial market with agent-based methods, either using rule-based agents [18, 19, 20] or learned agents [21, 22, 23]. However, the fidelity and flexibility are limited. Ruled-based agents rely on over-simplified assumptions of the market that can only represent known types of market participants and predefined market scenario. They are not trained with real market data but are constructed with hand-crafted rules instead, resulting in constrained simulation fidelity. Learned agents are trained to predict next order given history order flow where the order flow may contain over hundreds of orders in one minute. It is challenging to directly capture the long term dependency since a trading day lasts hundreds of minutes. They stress more on local distribution of the simulated order flow, neglecting the global dynamic. More importantly, the controllability to the generated market, which enables researchers and practitioners to systematically explore market behaviors under various conditions such as extreme or rare events, is absent from the literature. This underlines a practical gap for conducting scenario-based experiments and counterfactual analysis [24, 25].

In this paper, we propose a *Diffusion Guided meta Agent* (DiGA) model to address the controllable financial market generation problem. Specifically, we utilize a conditional diffusion model to capture dynamics of market state represented by time-evolving distribution parameters about mid-price return

39th Conference on Neural Information Processing Systems (NeurIPS 2025 Workshop on GenAI in Finance).



(a) Meta controller and order generator in DiGA. (b) Data flow of DiGA.

Figure 1: 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.

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.

2 Diffusion Guided Meta Agent Model

Order flow data is highly intricate and noisy, with a huge amount 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. 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, and guided by the meta controller, to generate order through a stochastic process. Figure 1 provides the overview of DiGA model.

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 day as a sample. Consequently, the training objective for fitting the distribution of the market states can be expressed as $\min \mathbb{E}_{\boldsymbol{x}} \mathcal{D}\left(p_{\mathcal{G}}(\boldsymbol{x}) \parallel q(\boldsymbol{x})\right)$. 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 $\{\beta_n \in (0,1)\}_{n=1,\dots,N}$.

Following common practice [4], 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 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 ϕ concurrently with the conditional sampler and the training objective is:

$$L_C := \mathbb{E}_{\mathcal{H}_e(\boldsymbol{O}),\boldsymbol{c},\boldsymbol{\epsilon}\sim\mathcal{N}(\mathbf{0},\boldsymbol{I}),n}[\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\boldsymbol{x}_n, n, \boldsymbol{\phi}(\boldsymbol{c}))\|^2]. \tag{1}$$

For both methods, we incorporate classifier-free guidance [26] 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:

$$\tilde{\boldsymbol{\epsilon}}_{\theta,\phi}(\boldsymbol{x}_n, n, \boldsymbol{c}) = (1-s)\boldsymbol{\epsilon}_\theta(\boldsymbol{x}_n, n) + s\boldsymbol{\epsilon}_\theta(\boldsymbol{x}_n, n, \boldsymbol{\phi}(\boldsymbol{c})), \tag{2}$$

where s is the conditioning scale that controls the strength of guidance. In practice, we adopt DDIM sampling [27] for better sampling 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 the Appendix C for the details of algorithm and architecture.

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 [17], who makes decisions following the optimization of CARA utility function given the market observations. The generated order is then recorded as $\boldsymbol{o}_i = (t_i, p_i, q_i, o_i) \sim p(\boldsymbol{o}|r_t, \lambda_t)$, where $t_i = \sum_{j=1}^{i} \delta_i$. 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: $\tilde{\boldsymbol{O}} = \{\boldsymbol{o}_1, \ldots, \boldsymbol{o}_{max}\} \sim p(\boldsymbol{O}|\tilde{\boldsymbol{x}})$, where $\tilde{\boldsymbol{x}} = \{\boldsymbol{r}, \boldsymbol{\lambda}\}$ is generated by the meta controller. The pseudo codes of the order generation procedure can be found in the Algorithm 2.

3 Experiments

3.1 Evaluation on Controlling Financial Market Generation

We train DiGA with three control targets respectively: *return, amplitude* and *volatility*, all of which are indicators that represent a vast range of market scenarios. For each indicator, 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.

Figure 2 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.

3.2 Evaluation on Generation Fidelity

We evaluate generation fidelity of DiGA and compare with market simulation baselines with both rule-based agents (*RFD* [18], *RMSC* [20]) and learning-based agents (*LOBGAN* [22]). We test the distribution discrepancy of several "stylized facts", which are statistics regarding asset returns and order book, between the real and simulated markets. These statistics are among the most representative features in the domain of the financial market micro-structure: *Minutely Log Returns* (MinR), *Return Auto-correlation* (RetAC), *Volatility Clustering* (VolC) and *Order Imbalance Ratio* (OIR).

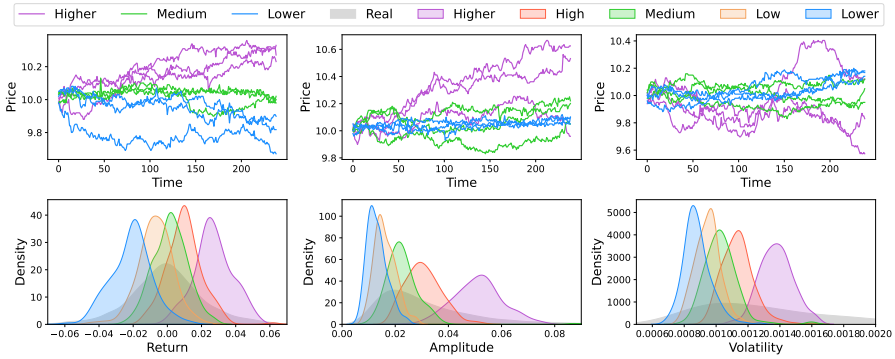


Figure 2: Aggregated price curves demos (first row) and the distribution of targeted indicators (second row). For the first row, each curve represents the order flow of one day. For the second row, each colored density represents the distribution of targeted indicator computed from generation results.

Figure 3 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. For numerical results, please refer to Appendix E.

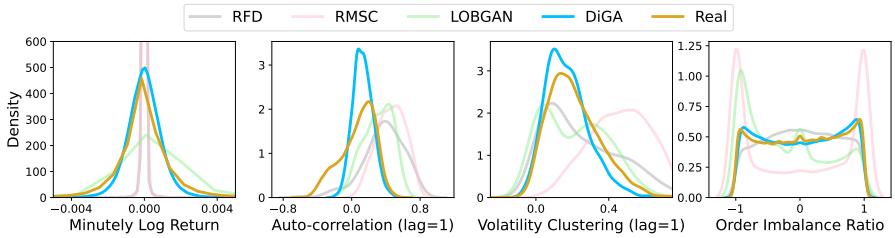


Figure 3: Comparison of stylized facts distribution across baselines. The x-axis is the stylized fact values and the y-axis is the density.

3.3 Evaluation for High-frequency Trading Task with Reinforcement Learning

We evaluate the helpfulness of DiGA as the environment for training reinforcement learning (RL) algorithms to perform high-frequency trading. We train an high-frequency trading agent with A2C algorithm in the environment generated with Replay, RFU, DiGA and the unconditional version of DiGA (DiGA-c). The agent are then tested with out-of-sample interactive replay and evaluated by *Daily return* (Ret), *Daily volatility* (Vol), *Sharpe ratio* (SR) and *Maximum drawdown* (MDD). Results are shown in Table 1, which demonstrate potential for applying DiGA in downstream task.

Table 1: Average out-of-sample test results (in percentage). Best results are highlighted in bold.

Environment	**Ret(%)(↑)**	**Vol(↓)**	**SR(↑)**	**MDD(%)(↑)**
Replay	$0.009_{\pm 0.043}$	$0.413_{\pm 0.090}$	$0.014_{\pm 0.008}$	$-1.133_{\pm 0.173}$
RFD	$0.000_{\pm 0.008}$	$0.159_{\pm 0.094}$	$0.011_{\pm 0.029}$	$-0.803_{\pm 0.327}$
DiGA-c	$0.015_{\pm 0.023}$	$\mathbf{0.147_{\pm 0.151}}$	$0.006_{\pm 0.066}$	$\mathbf{-0.715_{\pm 0.464}}$
DiGA	$\mathbf{0.029_{\pm 0.019}}$	$0.411_{\pm 121}$	$\mathbf{0.049_{\pm 0.031}}$	$-1.313_{\pm 0.156}$

4 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. 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.

References

[1] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, "Language models are few-shot learners," in *Advances in Neural Information Processing Systems*, 2020.

[2] A. Chowdhery, S. Narang, J. Devlin, M. Bosma, G. Mishra, A. Roberts, P. Barham, H. W. Chung, C. Sutton, S. Gehrmann *et al.*, "Palm: Scaling language modeling with pathways," *Journal of Machine Learning Research*, vol. 24, no. 240, pp. 1–113, 2023.

[3] K. Singhal, S. Azizi, T. Tu, S. S. Mahdavi, J. Wei, H. W. Chung, N. Scales, A. Tanwani, H. Cole-Lewis, S. Pfohl *et al.*, "Large language models encode clinical knowledge," *Nature*, vol. 620, no. 7972, pp. 172–180, 2023.

[4] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, "High-resolution image synthesis with latent diffusion models," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022.

[5] J. Lu, C. Clark, S. Lee, Z. Zhang, S. Khosla, R. Marten, D. Hoiem, and A. Kembhavi, "Unified-io 2: Scaling autoregressive multimodal models with vision language audio and action," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

[6] J. Copet, F. Kreuk, I. Gat, T. Remez, D. Kant, G. Synnaeve, Y. Adi, and A. Défossez, "Simple and controllable music generation," *Advances in Neural Information Processing Systems*, 2024.

[7] J. Wang, C.-Y. Hsieh, M. Wang, X. Wang, Z. Wu, D. Jiang, B. Liao, X. Zhang, B. Yang, Q. He *et al.*, "Multi-constraint molecular generation based on conditional transformer, knowledge distillation and reinforcement learning," *Nature Machine Intelligence*, vol. 3, no. 10, pp. 914–922, 2021.

[8] M. A. Skinnider, F. Wang, D. Pasin, R. Greiner, L. J. Foster, P. W. Dalsgaard, and D. S. Wishart, "A deep generative model enables automated structure elucidation of novel psychoactive substances," *Nature Machine Intelligence*, vol. 3, no. 11, pp. 973–984, 2021.

[9] A. M. Bran, S. Cox, O. Schilter, C. Baldassari, A. D. White, and P. Schwaller, "Augmenting large language models with chemistry tools," *Nature Machine Intelligence*, pp. 1–11, 2024.

[10] F. Fürrutter, G. Muñoz-Gil, and H. J. Briegel, "Quantum circuit synthesis with diffusion models," *Nature Machine Intelligence*, pp. 1–10, 2024.

[11] M. Aversa, G. Nobis, M. Hägele, K. Standvoss, M. Chirica, R. Murray-Smith, A. M. Alaa, L. Ruff, D. Ivanova, W. Samek, F. Klauschen, B. Sanguinetti, and L. Oala, "Diffinfinite: Large mask-image synthesis via parallel random patch diffusion in histopathology," in *Advances in Neural Information Processing Systems*, 2023.

[12] T. Tu, S. Azizi, D. Driess, M. Schaekermann, M. Amin, P.-C. Chang, A. Carroll, C. Lau, R. Tanno, I. Ktena *et al.*, "Towards generalist biomedical ai," *NEJM AI*, vol. 1, no. 3, p. AIoa2300138, 2024.

[13] R. Palmer, W. Brian Arthur, J. H. Holland, B. LeBaron, and P. Tayler, "Artificial economic life: a simple model of a stockmarket," *Physica D: Nonlinear Phenomena*, vol. 75, no. 1, pp. 264–274, 1994.

[14] T. Lux and M. Marchesi, "Scaling and criticality in a stochastic multi-agent model of a financial market," *Nature*, vol. 397, no. 6719, pp. 498–500, 1999.

[15] M. Raberto, S. Cincotti, S. M. Focardi, and M. Marchesi, "Agent-based simulation of a financial market," *Physica A: Statistical Mechanics and its Applications*, vol. 299, no. 1, pp. 319–327, 2001.

[16] C. Chiarella and G. Iori, "A simulation analysis of the microstructure of double auction markets," *Quantitative finance*, vol. 2, no. 5, p. 346, 2002.

[17] C. Chiarella, G. Iori, and J. Perello, "The Impact of Heterogeneous Trading Rules on the Limit Order Book and Order Flows," *Journal of Economic Dynamics and Control*, vol. 33, no. 3, pp. 525–537, 2009.

[18] S. Vyetrenko, D. Byrd, N. Petosa, M. Mahfouz, D. Dervovic, M. Veloso, and T. Balch, "Get real: realism metrics for robust limit order book market simulations," in *Proceedings of the ACM International Conference on AI in Finance*, 2020.

[19] D. Byrd, M. Hybinette, and T. H. Balch, "Abides: Towards high-fidelity multi-agent market simulation," in *Proceedings of the 2020 ACM SIGSIM Conference on Principles of Advanced Discrete Simulation*, 2020.

[20] S. Amrouni, A. Moulin, J. Vann, S. Vyetrenko, T. Balch, and M. Veloso, "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*, 2021.

[21] J. Li, X. Wang, Y. Lin, A. Sinha, and M. P. Wellman, "Generating realistic stock market order streams," in *Proceedings of the AAAI Conference on Artificial Intelligence, AAAI*, 2020.

[22] A. Coletta, J. Jerome, R. Savani, and S. Vyetrenko, "Conditional generators for limit order book environments: Explainability, challenges, and robustness," *CoRR*, vol. abs/2306.12806, 2023.

[23] J. Li, Y. Liu, W. Liu, S. Fang, L. Wang, C. Xu, and J. Bian, "Mars: a financial market simulation engine powered by generative foundation model," in *The Thirteenth International Conference on Learning Representations*, 2025.

[24] J. Guo, S. Wang, L. M. Ni, and H. Shum, "Quant 4.0: Engineering quantitative investment with automated, explainable and knowledge-driven artificial intelligence," *CoRR*, vol. abs/2301.04020, 2023.

[25] T. Mizuta, "An agent-based model for designing a financial market that works well," in *Proceedings of the IEEE Symposium Series on Computational Intelligence*, 2020.

[26] J. Ho and T. Salimans, "Classifier-free diffusion guidance," *CoRR*, vol. abs/2207.12598, 2022.

[27] J. Song, C. Meng, and S. Ermon, "Denoising diffusion implicit models," in *Proceedings of the International Conference on Learning Representations1*, 2021.

[28] J. Sohl-Dickstein, E. A. Weiss, N. Maheswaranathan, and S. Ganguli, "Deep unsupervised learning using nonequilibrium thermodynamics," in *Proceedings of the International Conference on Machine Learning*, 2015.

[29] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," in *Advances in Neural Information Processing Systems*, 2020.

[30] R. Cont, "Empirical properties of asset returns: stylized facts and statistical issues," *Quantitative finance*, vol. 1, no. 2, p. 223, 2001.

[31] X. Wang and M. P. Wellman, "Spoofing the limit order book: An agent-based model," in *Proceedings of the Conference on Autonomous Agents and MultiAgent Systems*, 2017.

[32] A. Coletta, M. Prata, M. Conti, E. Mercanti, N. Bartolini, A. Moulin, S. Vyetrenko, and T. Balch, "Towards realistic market simulations: a generative adversarial networks approach," in *Proceedings of the ACM International Conference on AI in Finance*, 2021.

[33] A. Coletta, A. Moulin, S. Vyetrenko, and T. Balch, "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*, 2022.

[34] Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole, "Score-based generative modeling through stochastic differential equations," in *Proceedings of the International Conference on Learning Representations*, 2021.

[35] P. Dhariwal and A. Q. Nichol, "Diffusion models beat gans on image synthesis," in *Advances in Neural Information Processing Systems*, 2021.

[36] J. Song, Q. Zhang, H. Yin, M. Mardani, M. Liu, J. Kautz, Y. Chen, and A. Vahdat, "Loss-guided diffusion models for plug-and-play controllable generation," in *Proceedings of the International Conference on Machine Learning*, 2023.

[37] Z. Kong, W. Ping, J. Huang, K. Zhao, and B. Catanzaro, "Diffwave: A versatile diffusion model for audio synthesis," in *Proceedings of the International Conference on Learning Representations*, 2021.

[38] H. Liu, Z. Chen, Y. Yuan, X. Mei, X. Liu, D. Mandic, W. Wang, and M. D. Plumbley, "AudioLDM: Text-to-audio generation with latent diffusion models," in *Proceedings of the International Conference on Machine Learning*, 2023.

[39] T. Brooks, B. Peebles, C. Holmes, W. DePue, Y. Guo, L. Jing, D. Schnurr, J. Taylor, T. Luhman, E. Luhman, C. Ng, R. Wang, and A. Ramesh, "Video generation models as world simulators," 2024. [Online]. Available: https://openai.com/research/video-generation-models-as-world-simulators

[40] Y. Guo, C. Yang, A. Rao, Z. Liang, Y. Wang, Y. Qiao, M. Agrawala, D. Lin, and B. Dai, "Animatediff: Animate your personalized text-to-image diffusion models without specific tuning," *International Conference on Learning Representations*, 2024.

[41] M. Kollovieh, A. F. Ansari, M. Bohlke-Schneider, J. Zschiegner, H. Wang, and Y. B. Wang, "Predict, refine, synthesize: Self-guiding diffusion models for probabilistic time series forecasting," in *Advances in Neural Information Processing Systems*, 2023.

[42] A. Coletta, S. Gopalakrishnan, D. Borrajo, and S. Vyetrenko, "On the constrained time-series generation problem," in *Advances in Neural Information Processing Systems*, 2023.

[43] X. Yuan and Y. Qiao, "Diffusion-ts: Interpretable diffusion for general time series generation," in *International Conference on Learning Representations*, 2024.

A Codes and Extended Version

Codes available in `https://github.com/microsoft/TimeCraft/tree/main/DiGA`.

Extended version available in `https://arxiv.org/pdf/2408.12991`.

B 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 2 summarizes the statistics of the dataset.

Table 2: 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.

C Detailed Description of DDPM

C.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:

$$q(\mathbf{x}_n|\mathbf{x}_{n-1}) := \mathcal{N}(\mathbf{x}_n; \sqrt{1-\beta_n}\mathbf{x}_n, \beta_n\mathbf{I}), \tag{3}$$

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

[1] https://www.wind.com.cn/. According to the data license, we are not able to share the full original data.

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 $\boldsymbol{\epsilon} \sim \mathcal{N} \sim (\mathbf{0}, \boldsymbol{I})$;
  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*;

$\bar{\alpha}_n := \prod_{s=1}^{n} \alpha_s$. For the reverse process, the diffusion model, parameterized by θ, yields:

$$p_\theta(\mathbf{x}_0, \mathbf{x}_1, ..., \mathbf{x}_N) := p(\mathbf{x}_N) \prod_{n=1}^{N} p_\theta(\mathbf{x}_{n-1}|\mathbf{x}_n), \tag{4}$$

where $p_\theta(\mathbf{x}_{n-1}|\mathbf{x}_n) := \mathcal{N}(\mathbf{x}_{n-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_n, n), \boldsymbol{\Sigma}_\theta(\mathbf{x}_n, n))$ 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:

$$L := \mathbb{E}[-\log p_\theta(\mathbf{x}_0)] \leq \mathbb{E}_q[-\log \frac{p_\theta(\mathbf{x}_0, ..., \mathbf{x}_N)}{q(\mathbf{x}_1, ..., \mathbf{x}_N|\mathbf{x}_0)}], \tag{5}$$

a widely adopted parameterization writes:

$$\boldsymbol{\mu}_\theta(\mathbf{x}_n, n) = \frac{1}{\sqrt{\alpha_n}}(\mathbf{x}_n - \frac{\beta_n}{\sqrt{1-\bar{\alpha}_n}}\boldsymbol{\epsilon}_\theta(\mathbf{x}_n, n)), \tag{6}$$

which simplifies the objective to:

$$L_{simple} := \mathbb{E}_{\mathbf{x}_0,\boldsymbol{\epsilon},n}[\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_n}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_n}\boldsymbol{\epsilon}, n)\|^2]. \tag{7}$$

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 $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ [29].

To obtain a conditional DDPM model using classifier-free guidance [26], 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

$$\tilde{\boldsymbol{\epsilon}}_{\theta,\phi}(\boldsymbol{x}_n, n, \boldsymbol{c})) = (1-s)\boldsymbol{\epsilon}_\theta(\boldsymbol{x}_n, n) + s\boldsymbol{\epsilon}_\theta(\boldsymbol{x}_n, n, \boldsymbol{\phi}(\boldsymbol{c})). \tag{8}$$

Both conditional and unconditional sampling can be accelerated by DDIM [27].

C.2 Algorithmic Procedure for Training Meta Controller

The pesudo code of training meta controller can be found in Algorithm 1.

C.3 Detailed Parameters of Training Meta Controller

The denoising model ϵ_θ used in meta controller is adapted from [29]: 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 3 lists the required parameters along with the above description.

We train the diffusion model 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. Training is operated on 1 NVIDIA Tesla V100 GPU for 10 epochs, which takes approximately 2 hours for the A-Main dataset and 1 hour for the ChiNext dataset. We set a pseudo initial stock price at 10 for every generation run.

D Details of Generating with DiGA

D.1 Algorithmic Procedure for Generating Financial Market with DiGA.

Upon each wake-up, the meta agent generate an actor agent $\mathcal{A}_i$ within the family of heterogeneous agents [17], who makes decisions following the optimization of CARA utility function given the market observations. The order generation procedures are described as follows:

- The actor agent is initialized with random holding positions S and the corresponding amount of cash C, as well as the 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 distributions, configured such that the expected values of the fundamental, chartist, and noise weights follow a ratio of 10:1.5:1.
- 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 agent estimate future price $\hat{p}_t = p_t \exp(\hat{r})$, the actor agent is able to obtain its specific demand function $u(p) = \frac{\ln(\hat{p}_t/p)}{aVp}$ by deriving from CARA utility on future wealth [17], where a is risk averse coefficient and V is history price volatility. The actor agent also estimate the price p_l such that $p_l(u(p_l) - S) = C$ is satisfied, as the lowest order price.
- 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 $o_i = \mathrm{sign}(q_i)$ where $o_i = 1$ indicates buy order and $o_i = 0$ indicates sell order.

The generated order is then recorded as $\boldsymbol{o}_i = (t_i, p_i, q_i, o_i) \sim p(\boldsymbol{o}|r_t, \lambda_t)$, where $t_i = \sum_{j=1}^{i} \delta_i$. The pseudo codes of the algorithm can be found in Algorithm 2. We discuss the input parameters in D.2 and D.3.

D.2 Discussion on Meta Controller Sampling Parameters

When sampling with the meta controller, we apply DDIM [27] 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.

D.3 Discussion on Meta Agent Parameters

Following the heterogeneous agent settings [17], the meta agent employs several probabilistic parameters to ensure heterogeneity. The parameters and their usage are as follows: 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 $\tau_0 = 30, \alpha_0 = 0.1, \sigma_n = 1e^{-4}, p_0 = 10$ for all runs to avoid overfitting these parameters. Nevertheless, they can be calibrated to further improve fidelity.

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 $\boldsymbol{x}_n \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$;

for $n = N$ **to** 1 **do**

  Sample $\boldsymbol{z} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$;

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

  $\boldsymbol{x}_{n-1} = \frac{1}{\sqrt{\alpha_n}}(\boldsymbol{x}_n - \frac{1-\alpha_n}{\sqrt{1-\bar{\alpha}_n}}\hat{\boldsymbol{\epsilon}}) + \sigma_n \boldsymbol{z}$;

end

(Phase 2: generate order flow with meta agent)

Initialize $t = 0$, $\boldsymbol{O} = \emptyset$, $p_t = p_0$;

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 $t > T$;

Table 3: 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

E Additional Experiment Description and Results

We provide detailed descriptions of the experiments and the full result table with both mean and std across 3 independent run with different random seeds for main tables in Table 5, Table 6, Table 7.

E.1 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 5 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. "No Control" presents results from a variant of DiGA with the condition mechanism in meta controller removed, whose generations are independent to the control target. From Table 5, DiGA demonstrates the ability of control by keeping a relatively small error between realized indicator value and control target, while the outcome without control is either random or repeating some particular scenarios.

Figure 2 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.

E.2 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** [18] is a multi-agent based market simulation configuration with random fundamental and diverse agent types, consisting of 1 market maker, 25 momentum, 100 value and 5,000 noise agents.
- **RMSC** [20] is the reference market simulation configuration introduced by ABIDES-gym, which contains all RFD agents and 1 extra percentage-of-volume (POV) agent who provide extra liquidity to the simulated market.
- **LOBGAN** [22] trains a conditional Wasserstein GAN with gradient penalty 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. These selected statistics 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 volume difference between best bid and best ask. It represents the tendency for trading of market participants.

We demonstrate results sampled with the unconditional version of DiGA in the experiment. Figure 3 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 6 provides the quantitative 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.

Table 4: Comparison on computational efficiency.

Model	Time(ms)/Order
RFD	0.049
RMSC	0.075
LOBGAN	1.710
DiGA	**0.017**

E.3 Evaluation for High-frequency Trading Task with Reinforcement Learning

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

E.3.1 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 1440 decision steps in 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.

E.3.2 Results

Table 7 displays the average numerical results for the high-frequency trading task. The trading agent trained with DiGA generated environment has earned the best daily return and Sharpe ratio against all baselines. Agent trained with DiGA-c obtains the best daily volatility and maximum drawdown, which indicates a more conservative strategy. These results show that the DiGA generated environment helps the trading agent learn a better policy. Moreover, the result difference between DiGA and DiGA-c indicates that performing control on the training environment is meaningful for establishing decision preference of a trading agent.

E.4 Analysis on Computational Efficiency

In this section, we compare the computational efficiency of between DiGA and all baseline models. We use the speed of financial market generation which is measured by the average time used for generating an order as the benchmark for computation efficiency. The results are shown in Table 4.

From Table 4, DiGA can generate orders at the fastest speed among baselines, taking approximately 0.017 milliseconds for generating each order and providing support for real-time, latency-critical applications. RFD and RMSC methods generate orders by a slightly slower rate, which are the results of their specific agent design that requires quoting from the virtual exchange for each decision time. LOBGAN generates orders about 100 times slower than DiGA, due to the recurrent neural network architecture and the autoregressive nature of its generator module. Considering the best performance on fidelity, we can conclude that DiGA achieves the best efficiency on financial market generation.

Table 5: 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 6: 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

Table 7: Detailed 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	$\mathbf{0.029_{\pm 0.019}}$	$0.411_{\pm 121}$	$\mathbf{0.049_{\pm 0.031}}$	$-1.313_{\pm 0.156}$

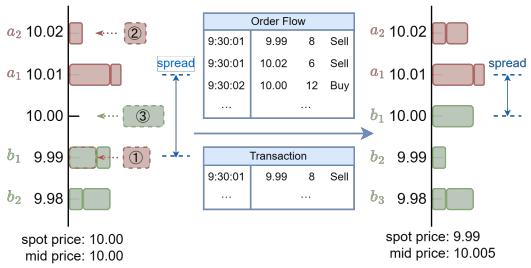


Figure 4: Limit order book and order flow

F 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.

F.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 [17].

The output of market simulation model is a series of orders $\boldsymbol{O} = \{(t_1, p_1, q_1, o_1), (t_2, p_2, q_2, o_2), ...\}$, 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 4.

F.2 Financial Market Simulation

Early works on market simulation has followed a multi-agent approach [15, 13, 14], using rule-based agents under simplified trading protocols to replicate "stylized facts" [30] such as volatility clustering. [16, 17] 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 [18, 19, 31]. With the recent success of machine learning, researcher has also employed neural networks as world agent to directly predict orders given history [32, 21, 22, 33]. In contrast, our model employs conditional diffusion model for controlling agent based models to generate orders.

F.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 [28, 29, 34]. Diffusion models have been built to generate data in different modalities, such as image [4, 11, 35, 36], audios [37, 38], videos [39, 40] and general time series [41, 42, 43]. Different from existing works, we are the first to apply diffusion model for generating financial market.