MarS: a Financial Market Simulation Engine Powered by Generative Foundation Model

Junjie Li, Yang Liu, Weiqing Liu, Shikai Fang, Lewen Wang, Chang Xu, Jiang Bian

Microsoft Research Asia

ICLR 2025 (Conference Paper)

November 3, 2025

Outline

MarS (**Mar**ket **S**imulation Engine) powered by **LMM** (**L**arge **M**arket **M**odel)

- **Introduction**
- **MarS Overview**
- **MarS Detailed Design**
- **Experiments & Results**

Background

Foundation Model[1] Breakthroughs

- Generative Models in **NLP**: NLP Milestones (GPT-3 in 2020 to GPT-4 in 2023)
- Generative Models in **Vision and Simulation**: These simulation efforts primarily target the **physical world**, such as autonomous driving, robotics and games, by generating visual scenes or trajectories. Researchers treated video generation as a path to simulation.

[1] "Foundation Model" means models are trained on broad datasets and can be adapted to a wide range of downstream tasks.

Related Work

Agent-Based Market Simulators (Early Approaches)

- Multi-Agent Simulations: This tradition treats the market as an ecosystem of agents (e.g., liquidity providers, momentum traders), each following rules, to see emergent outcomes.
- ABIDES Platform (Amrouni et al., 2021))
- Pros and Cons:
  - can incorporate domain knowledge (e.g., specific trading behaviors or market rules) and produce interpretable scenarios (we can trace each agent's actions).
  - these models heavily rely on the **assumptions** made about agent behaviors.

Related Work

Generative Models for **Limit Order Books** (GANs)

- GANs could model financial time series, producing sequences that mimic real stock price movements.
- Series of GAN-based Simulators (Coletta et al.): generate LOB data, “world agent” concept, conditional LOB generators

- Pros and Cons:
  - reproduce stylized facts of markets, simulate millions of orders
  - **no interactivity** or controllability, instability or **mode collapse**

Related Work

Advanced Generative Market Models

- Incorporating microstructure data recurrent neural network model of the LOB that captures the time-evolution of the entire order book state
- State-Space Models: built on a Deep State Space Model architecture, their model processes the sequence of order messages (limit orders, cancellations, trades) and generates new sequences one event at a time
- Pros and Cons:
  - did not demonstrate the simulator's usefulness on downstream tasks
  - they generate data but typically cannot adjust to **external inputs** in the middle of a sequence.

Motivation

- **Realism vs. Complexity:** Traditional agent-based simulators often simplify behavior, and early generative models, even if statistically convincing, may miss critical market dynamics.
- **Lack of Interactivity:** No existing simulator allowed human or algorithmic interaction in the loop with full realism.
- **Controllability and Scenario Design:** Agent-based models allowed some scenario setup (via agent behaviors), but generative models so far offered no simple way to condition on scenarios
- **Validation and Utility:** stylized facts and distributional similarity were the benchmarks, stylized facts and distributional similarity were the benchmarks

Key Contribution

How MarS addresses these challenges?

- Large Market Model (LMM): A generative foundation model trained on **order-level market data** (like a "language model" but for financial orders)
- MarS Engine: A simulation platform built around LMM that produces **realistic market order streams**, integrating user inputs and scenario controls
- **Controllable** Scenario Generation: MarS introduces conditional generation features to let users shape the scenario
- Interactivity – **User-Injected Orders**: It allows the user (or a test algorithm) to inject orders into the simulation at any time, and the LMM will respond by generating the subsequent order flow conditional on those user actions

Overview

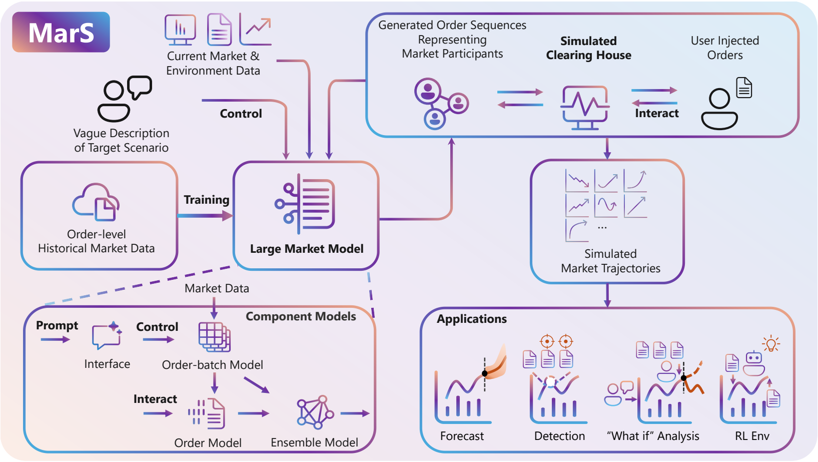
Components:

- LLM:
  - Order-Level Model
  - Order-Batch Model
  - Ensemble Model
- Simulated Clearing House

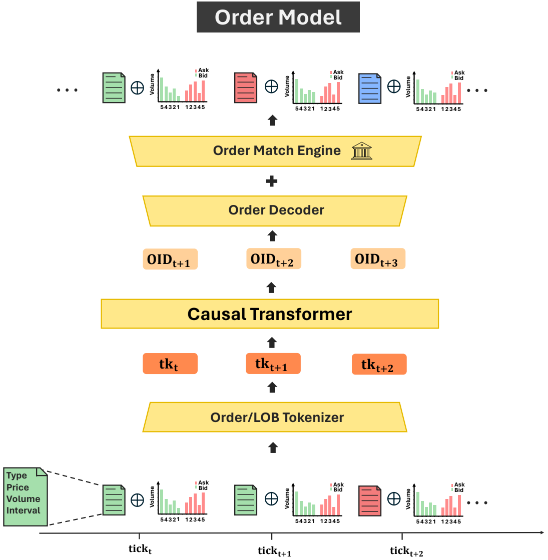
Workflow: Historical data → train LMM; At runtime → LMM generates orders → clearing house matches orders → feedback loop with updated market state for next generation

Applications Enabled: Forecasting, Risk/Anomaly detection, What-if analysis, RL trading environment

Overview


MarS
Current Market &
Environment Data
Vague Description
of Target Scenario
Control
Order-level
Historical Market Data
Training
Large Market Model
Generated Order Sequences
Representing
Market Participants
Simulated
Clearing House
User Injected
Orders
Interact
Simulated
Market Trajectories
Market Data
Prompt
Interface
Control
Order-batch Model
Component Models
Interact
Order Model
Ensemble Model
Applications
Forecast
Detection
"What if" Analysis
RL Env

Order-Level Model (Micro LMM)



Tokenization

$$Emb_i = \text{emb}(order_i) + \text{linear\_proj}\left(LOB_i^{\text{volumes}}\right) + \text{emb}\left(LOB_i^{\text{mid\_price}}\right).$$

Order Information

Input (encodes four sub-fields):

Field	Discrete Range
Type	{Ask, Bid, Cancel}
Price	$[-32, +32]$ (relative to mid-price)
Volume	$[0, 32]$
Interval	$[0, 16]$

Output: A d-dimensional vector (emb_dim)

Tokenization

$$Emb_i = \text{emb}(order_i) + \text{linear\_proj}\left(LOB_i^{\text{volumes}}\right) + \text{emb}\left(LOB_i^{\text{mid\_price}}\right).$$

LOB Depth

Input: A length-10 real-valued vector of order-book volumes

- representing the remaining order volumes of the first to fifth bid and ask levels respectively (bid1...bid5, ask1...ask5)

Output: A d-dimensional vector (emb_dim)

Tokenization

$$Emb_i = \text{emb}(order_i) + \text{linear\_proj}\left(LOB_i^{\text{volumes}}\right) + \text{emb}\left(LOB_i^{\text{mid\_price}}\right).$$

Mid-Price Background

Input: number of mid-price tick changes since market open (a discrete integer)

Output: A d-dimensional vector (emb_dim)

Tokenization - An Example

Assume the embedding dimension is $d = 4$ (for simplicity):

$\mathrm{emb}(order_i)$

- Input: order_type = Bid, price_level = 3, volume = 10, interval = 2
- Output: $[0.10,\ -0.20,\ 0.30,\ 0.40]$

$\mathrm{linear\_proj}(LOB_i^{\text{volumes}})$

- Input: LOB_volumes = [100,90,...,80](10 numbers)
- Output: $[-0.05,\ 0.02,\ 0.07,\ 0.01]$

$\mathrm{emb}(LOB_i^{\text{mid-price}})$

- Input: mid_price_index=15
- Output: $[0.03,\ 0.04,\ -0.01,\ 0.10]$

$Emb_i = [0.08,\ -0.14,\ 0.36,\ 0.51]$, is the **token embedding** at position i; it will be fed into the Causal Transformer to predict the next order.

Model Training Set

Data

- top 500 liquidity stocks in the Chinese stock market
- period from 2017 to 2023
- 16 billion order tokens

Architecture

- LLaMA2, sequence length is set at 1024 (Touvron et al., 2023)
- AdamW optimizer, fp16 precision (Loshchilov, 2017)
- DeepSpeed ZERO stage 2 (Rajbhandari et al., 2020)

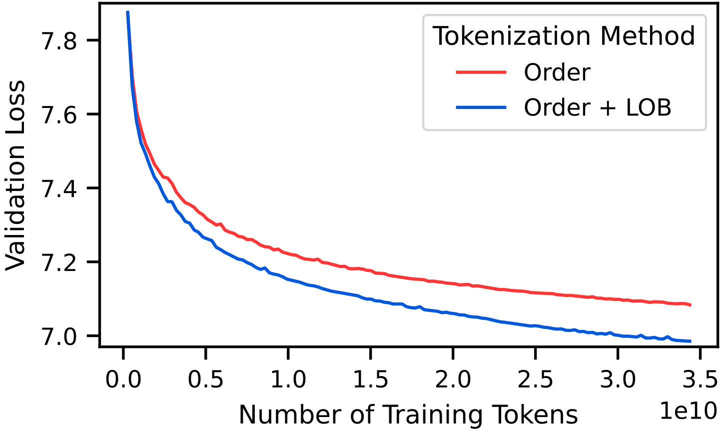
Batch size: 4096, sequence length: 1024 $\rightarrow$ 4 million tokens per optimization step

Model Training Set

Order vs. Order + LOB

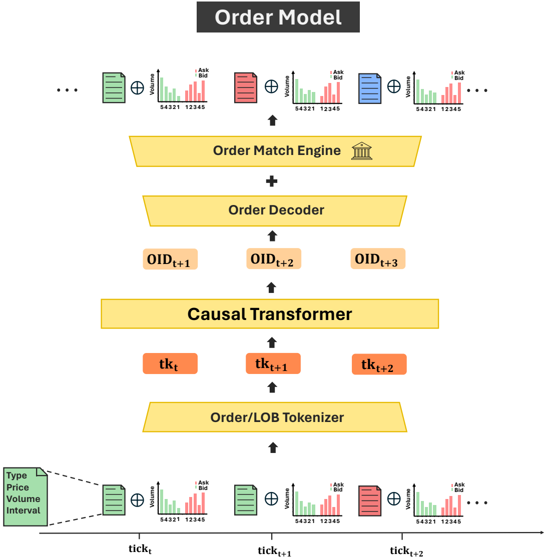
```
def forward(self, features: Tensor) -> Tensor:
    """Tokenize inputs."""
    ...
    embs = [
        self.emb_order_type(order_type),
        self.emb_price_level(price_level),
        self.emb_pred_order_volume(pred_order_volume),
        self.emb_order_interval(order_interval),
        # with LOB
        self.emb_chg_to_open(...),
        self.emb_time_to_open(...),
        self.lob_tokenizer(features[:, 5:15].to(dtype)),
    ]
    ...
    return tokens
```

Model Training Result



A comparative analysis of the tokenization process with and without LOB

Order-Level Model



Order Match Engine

Purpose: After matching, the system uses the "new order + updated book snapshot" as the next input and repeats the entire pipeline—until you've generated as many orders as you need.

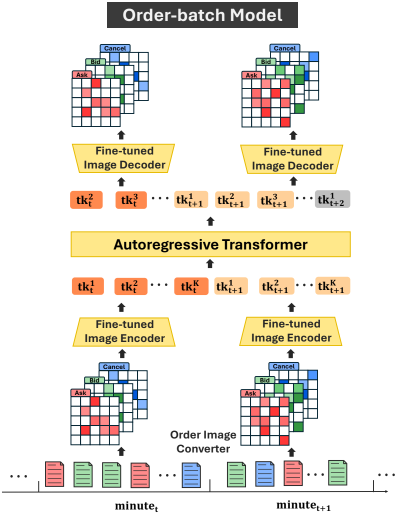
Match: Apply the exchange's matching rules to the decoded orders (decoded from logits): either execute trades against resting orders or add it as a new limit order, yielding an updated LOB state.

Record Trades or Book Updates:

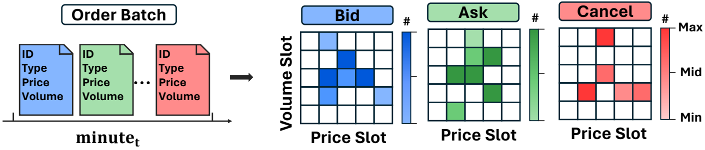
- If the order was executed, log the trade price and quantity.
- If it became a resting order, update the appropriate price-level queue in the book.

Build the next input features ...

Order-Batch Model (Macro LMM)



Order Image Converter



The order image converter transforms order data into a visual representation (**per minute for each stock**).

Input: All orders in 1 minute with

- **type:** bid, ask, cancel
- **price slot:** number of ticks above or below the mid-price
- **volume slot:** e.g. 1–10 shares=slot 1, 11–50 shares=slot 2

Output: A 32×32, 3 channels image with **pixel value** (representing the number of orders with the same attributes, with higher pixel values indicating more orders (clamped to [0–100]))

Order Image Converter - An Example

Consider the one-minute interval from 09:30:00 to 09:30:59 and observe the following eight events:

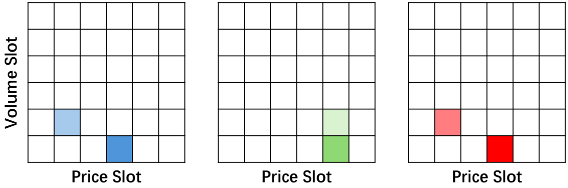
Index	Type	Price Slot (ticks from mid-price)	Volume (shares)
1	Bid	+1	5
2	Bid	+1	5
3	Bid	-2	20
4	Ask	+2	10
5	Ask	+2	10
6	Ask	+2	50
7	Cancel	+1	5
8	Cancel	-2	20

Volume slot 1 = 1–10 shares, slot 2 = 11–50 shares, and so on
X-axis (Price Slot): from 3 to +3 ticks, Y-axis (Volume Slot): slot 1, 2, . . .

Order Image Converter - An Example

Three "heatmaps" $\rightarrow$ a "snapshot":

- Blue channel (Bids):
  - (+1,slot 1): 2 orders (event 1 & 2) $\rightarrow$ value = 2
  - (-2,slot 2): 1 orders (event 3) $\rightarrow$ value = 1
- Green channel (Asks):
  - (+2,slot 1): 2 orders (event 4 & 5) $\rightarrow$ value = 2
  - (+2,slot 2): 1 orders (event 6) $\rightarrow$ value = 1
- Red channel (Cancels):
  - (+1,slot 1): 1 orders (event 7) $\rightarrow$ value = 1
  - (-2,slot 2): 1 orders (event 8) $\rightarrow$ value = 1



Tokenization

Purpose: convert each "order-batch image" into a discrete token sequence for autoregressive modeling

Model: a pre-trained VQGAN from LDM (Rombach et al., 2022), which was trained on the LAION-400M database (Schuhmann et al., 2021)

Input: A 32×32, 3 channels image

- Down-sampling factor $f = 4$: 32×32 → 8×8 latent grid
- Codebook size $Z = 8192$: number of discrete embeddings
- Code dimension $d = 3$: each token reconstructs an RGB patch

Output: Each 32×32×3 image → 8×8 = 64 tokens (indices in [0,8191])

Fine-tuning: from natural images to order images

Autoregressive Transformer

Purpose: Concatenate the tokens of N consecutive minutes into **one long sequence** and train an autoregressive transformer to **predict the next token** given all previous tokens

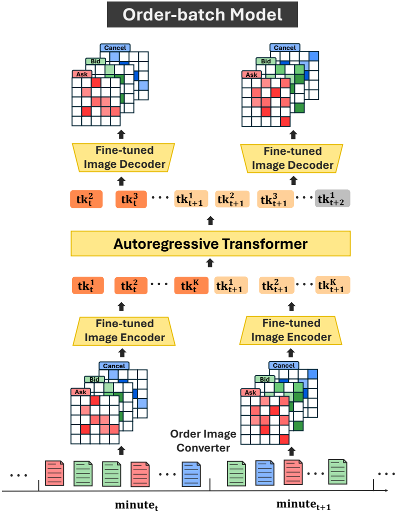
Model: LLaMA 2 (Touvron et al., 2023)

Input: 1024 tokens

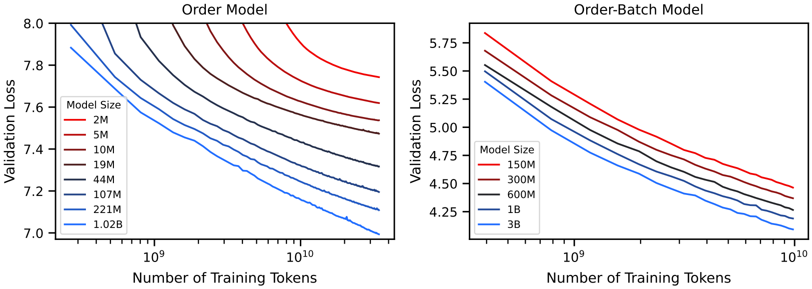
- 16 minutes$\times$64 tokens/minute $\rightarrow$ total sequence length $=$ 16$\times$64 $=$ 1024 (well below LLaMA2's 4096 context limit)

Output: autoregressively generate 64 new tokens

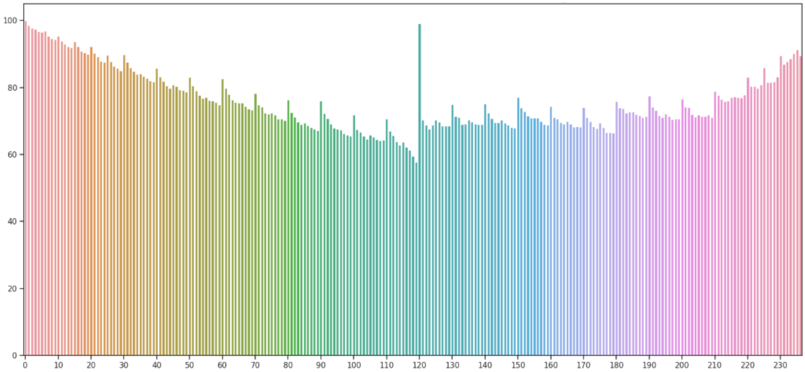
Order-Batch Model



Training Loss: Order-level and Order-Batch Model



Intraday Distribution of the Average Number of Orders



X-axis: Ranges from 0 to 239, representing the 240 consecutive minutes of a trading day (0–119: Morning session, 09:30–11:29 (120 minutes), 120–239: Afternoon session, 13:00–14:59 (120 minutes))

Y-axis: Average number of orders per minute (averaged across all sample stocks)

Pros and Cons of Order-level and Order-Batch Model

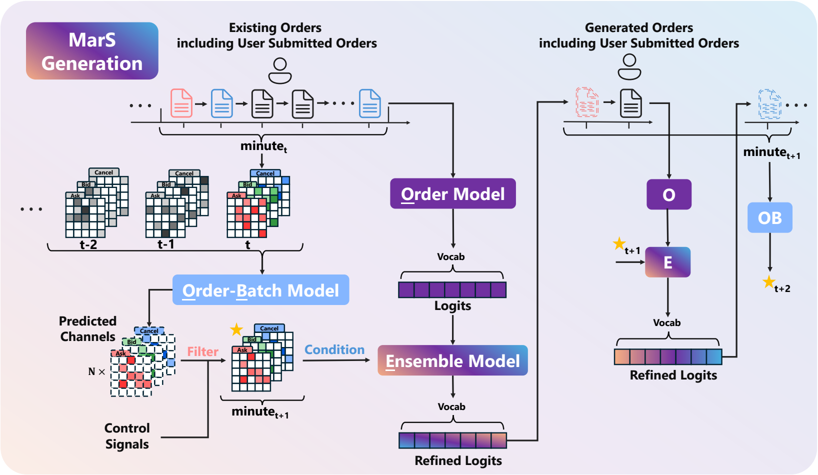
Order-level model:

- This model generates orders individually and is designed to reflect **short-term market impacts** rapidly.
- However, it lacks the ability to generate **target scenarios** over the long run.

Order-batch model:

- This model generates **order channels**, representing the macro behavior of the market, and can be used to follow control signals.
- However, it lacks the ability for **interactive** market simulation.

Ensemble Model Overview



Workflow

Ensemble Model - Refinement:

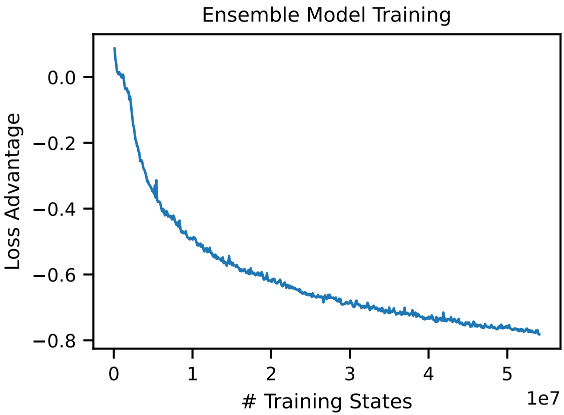
Input:

- raw logits from the Order Model
- the target channel distribution (condition)

Process: Apply **cross-attention** so that macro-level channel information refines the micro-order logits.

Output: **Refined logits**

Ensemble Model Training Result



X-axis represents the number of training samples

Y-axis represents the loss advantage over the order model

Simulated Clearing House (Order Match Engine)

Core Function:

- **Real-time matching:** when a new order is input, the simulated clearing house immediately matches buys against sells according to the chosen matching rules (MTCH_R)
- **Order book updates:** Matched quantities are removed from the LOB; any remainder stays in the book as an unfilled order

Output: current LOB snapshot

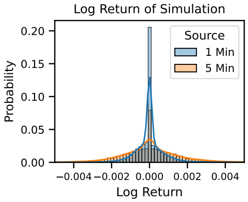
Continuous Loop: a dynamic sandbox

Training vs. Simulation

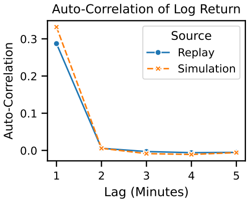
- **During training**, real historical LOB snapshots (replay data) are used as conditioning inputs so the model learns **accurate responses** without compounding batch-model noise.
- **During live simulation**, the model uses the clearing-house's own live LOB outputs as its macro input, closing the loop end-to-end.

Realistic Simulations

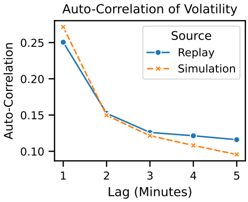
Three stylized facts:



(a) Aggregational Gaussianity

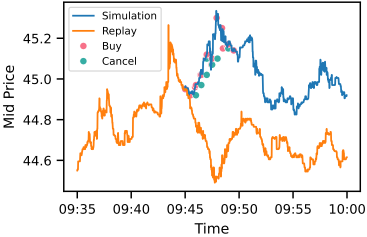


(b) Absence of Autocorrelations

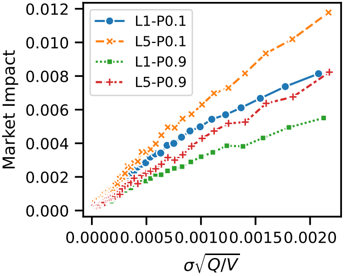


(c) Volatility Clustering

Interactive Simulations



(a) Synthetic market interaction

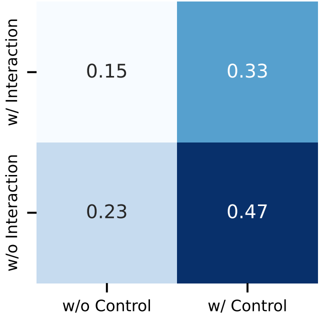


(b) Square-Root-Law Validation

Time-Weighted Average Price (TWAP) Strategy

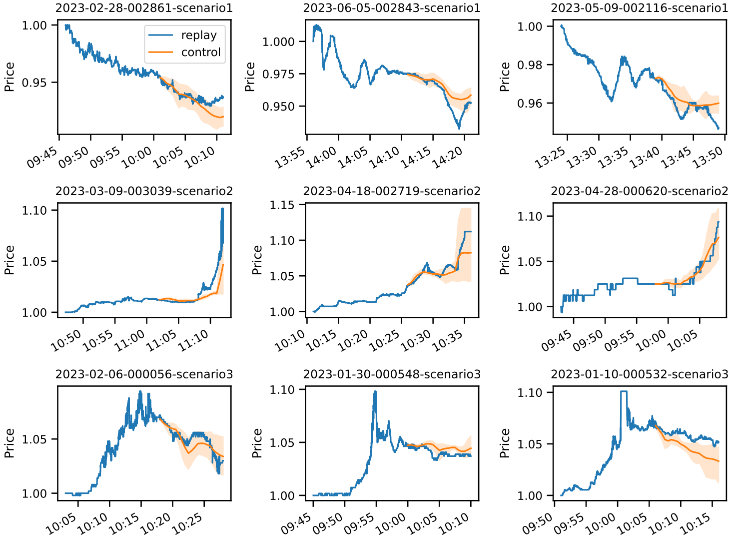
- Maximum Passive Volume Ratio (PVR) $\rightarrow$ P
- Aggressive Price (AP) $\rightarrow$ L

Controllable Simulations

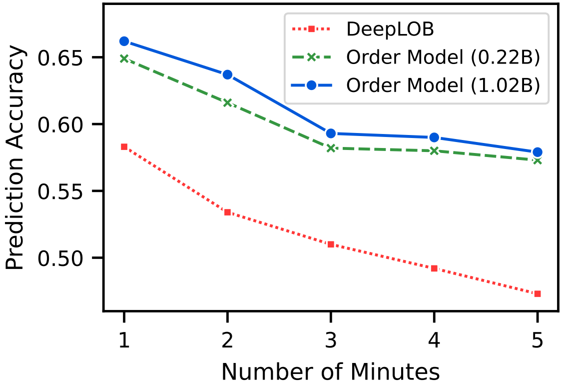


Configurations **with control but no interaction** achieve the highest correlation scores, while introducing interaction reduces control precision ($0.47 \rightarrow 0.33$).

Different Scenario Generation

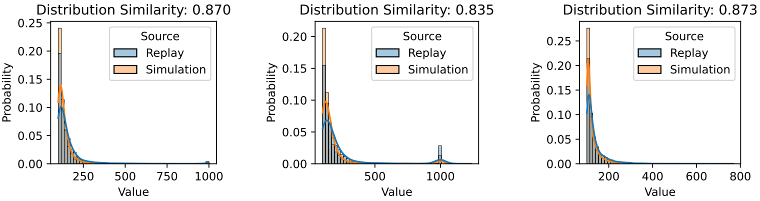


Forecasting



$$l = \frac{\frac{1}{n}\sum_{i=1}^{n} m_i - m_0}{m_0}.$$

Detection

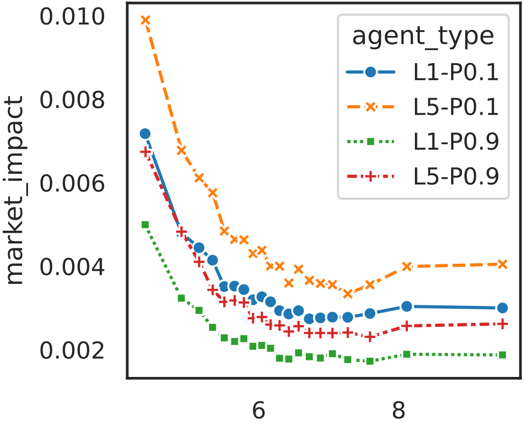


(a) Pre-manipulation (b) Manipulation period (c) Post-manipulation

Three periods, each 522 trading days

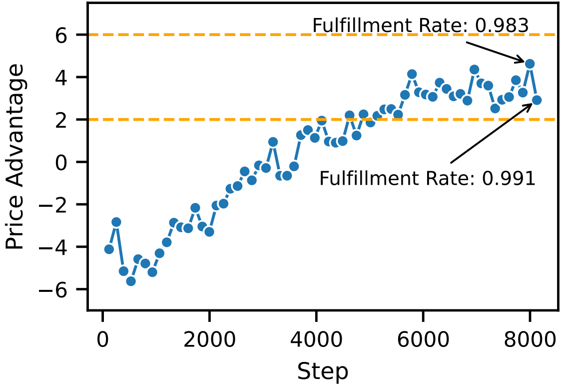
x: Spread value is defined as **best ask price - best bid price**

'What if' Analysis on Market Impact



New factor: Resiliency (From symbolic regression and genetic algorithms)

RL Environment



Performance of the trading agent (Baseline: best-configured TWAP agent)

Conclusion

MarS is

a comprehensive financial system for orders with these capabilities
- Realism, High-Resolution, Controllability, Interactivity

a powerful tool for a wide range of downstream applications:

- Forecast Tool: based on recent orders and LOB, simulating future market trajectories
- Detection System: identifies potential risks not apparent from current observations
- Analysis Platform: answers a wide range of "what if" questions by providing a realistic simulation environment
- Agent Training Environment: the realistic and responsive nature of MarS makes it ideal for training reinforcement learning agents