How My Simulator Mimics a Real Exchange

1. Introduction

Modern financial markets such as stock exchanges and cryptocurrency exchanges operate using an **order-driven market structure**, where buyers and sellers submit orders that are matched electronically. At the core of such markets lies the **Limit Order Book (LOB)**, which records all outstanding buy and sell orders and determines how trades are executed.

The objective of this project was to design and implement a **basic market simulator** that captures the essential mechanics of a real exchange. The simulator does not rely on historical data; instead, it generates synthetic order flow using simple agent behaviors. Despite this simplicity, the simulator reproduces many realistic market properties such as bid-ask spread formation, liquidity accumulation, price stabilization, and trade execution through price–time priority.

This report explains how the simulator mimics a real exchange in terms of **order handling, matching logic, agent interaction, time modeling, and liquidity dynamics**.

---

2. Order Book Mechanics

The simulator is built around a **Limit Order Book**, which maintains two ordered lists:

- **Bids**: Buy orders, sorted by highest price first and then by earliest arrival time.
- **Asks**: Sell orders, sorted by lowest price first and then by earliest arrival time.

Each order consists of:

- Order ID
- Side (buy or sell)
- Price
- Quantity
- Timestamp

This structure closely matches real electronic exchanges such as NSE, NASDAQ, or Binance.

Matching Logic

Whenever a new order is submitted, the matching engine checks whether a trade can occur. A trade is executed if:

best_bid ≥ best_ask

If this condition is met:

- Orders are matched at the **ask price**
- Partial fills are allowed
- Orders with zero remaining quantity are removed from the book

This rule mirrors real continuous trading systems, where trades occur whenever a buy order is willing to pay at least the lowest selling price.

---

3. Price–Time Priority (FIFO)

Within each price level, the simulator uses **First-In-First-Out (FIFO)** matching. This means:

- Orders at better prices are executed first
- Among orders at the same price, earlier orders are executed before later ones

FIFO is widely used in equity markets because it:

- Rewards patience
- Encourages traders to quote early
- Produces tighter spreads and deeper liquidity

This design choice ensures fairness and realism in execution behavior.

---

4. Agent-Based Trading

Instead of using real market data, the simulator generates orders through **agents**, each representing a simplified trader type.

Random (Noise) Traders

These agents submit buy or sell limit orders at prices drawn from a random distribution around the current mid-price. Their arrivals approximate **Poisson order flow**, similar to uninformed traders in real markets.

This behavior reflects findings from **Gode and Sunder (1993)**, who showed that even zero-intelligence traders can produce efficient market outcomes when the exchange mechanism is realistic.

Market Makers

Market makers provide liquidity by placing buy and sell orders on both sides of the book. Their presence:

- Narrows the bid-ask spread
- Increases depth near the mid-price
- Stabilizes prices

This mimics real designated market makers and liquidity providers.

Market Takers

Market takers submit aggressive orders that immediately consume liquidity. They represent traders who prioritize execution speed over price improvement.

The interaction between these three agent types recreates realistic market dynamics.

---

5. Liquidity, Spread, and Volatility

One of the most important outcomes of the simulator is the **emergence of liquidity**.

- The **bid-ask spread** forms naturally from competing buy and sell orders.
- As more limit orders accumulate, the spread shrinks.
- Market depth becomes thickest near the mid-price.
- Price volatility decreases as liquidity increases.

This behavior aligns with theoretical models such as **Kyle (1985)**, which explain how liquidity absorbs trading shocks and reduces price impact.

Depth charts and price series generated by the simulator visually demonstrate these effects.

---

6. Discrete Event Simulation

The simulator uses a **discrete event clock** rather than wall-clock time. Each simulation step corresponds to an event (order submission or trade), not elapsed seconds.

This approach is used in real exchange engines because it:

- Ensures deterministic behavior
- Allows exact replay of simulations
- Is suitable for reinforcement learning environments

Wall-clock time is avoided because it introduces nondeterminism and makes debugging and learning unstable.

---

7. Market Data Logging and Visualization

A logging module records:

- Best bid prices
- Best ask prices
- Executed trades

These logs act as a simplified **market data feed**, similar to what exchanges broadcast to participants. Visualizers then plot:

- Best bid / ask time series
- Spread convergence
- Price stability over time

This separation between execution and observation reflects real market architecture.

---

8. Limitations of the Simulator

While realistic in structure, the simulator has limitations:

- No order cancellations
- No latency modeling
- No transaction costs
- No informed traders or news shocks

These features were intentionally excluded to keep the system minimal and interpretable.

---

9. Conclusion

This project demonstrates that a relatively simple simulator can successfully reproduce many key properties of real financial exchanges. By combining a realistic limit order book, price–time priority matching, and basic agent behaviors, the system naturally generates spreads, liquidity, and stable prices.

The simulator provides a strong foundation for future extensions, including:

- Reinforcement learning agents
- Inventory and PnL tracking
- Order cancellations and latency
- More advanced market microstructure studies

Overall, the project offers both a practical implementation and a conceptual understanding of how orders become trades in modern electronic markets.