Analysis of Stochastic Processes with Neural Networks

Molnár András

Individual Project II.

03-06-2026

Table of Contents

Introduction

Model overview
- The uncontrolled model
- Controlled trader

Implementation and HJB
- Simulation framework
- HJBQVI solver

Validation and results
- The value surface
- Monte Carlo validation
- Policy comparison

Future work
- Reinforcement learning
- AI tools

Introduction and Motivation

- Classical asset price models often describe prices with **continuous-time processes**
  - for example **Brownian motion** or diffusion-based models
  - mathematically convenient
  - but they abstract away from individual trading events

- In real markets, short-term price changes are strongly connected to **order flow**
  - A buy market order can push the price up
  - A sell market order can push the price down
  - The size of the move depends on how much **liquidity** is available

Microstructure-based modelling

- The project is based on a microstructure model by Bank, Cartea and Körber
- Instead, **price changes are generated by discrete events in the order book**
- Market orders move the price, while limit orders and cancellations change liquidity
- The **size of price impact depends on the current liquidity** level

What is new this semester?

- Last semester: simulation of the uncontrolled market dynamics
  - simulation of order flow, liqudity and price paths
  - exploratory analysis of simulated market dynamics
- This semester: extension to the full controlled trader problem
  - trader inventory and cash
  - trade signals and signal-based trades
  - transaction costs and price impact of the trader
  - trading halt mechanism
  - terminal liquidation
- Main numerical goal: compute a benchmark strategy that tells the trader what to do in each market state

Order flow and liquidity

- The model is event-driven: market activity is generated by random order arrivals
- The key market variable is **liquidity** λ_t

$$d\lambda_t = dL_t - |dM_t|$$

- L_t: limit order and cancellation flow, M_t: market order flow
- Market orders **reduce** liquidity, while limit orders can **add** liquidity
- Order arrivals are modeled by a marked **Poisson** point **process**

$$dL_t = \int_{E\times\mathbb{R}_+} \mathbf{1}_{\{y\le f(\lambda_{t-})\}}\rho(e)\,N(dt,de,dy),$$

$$dM_t = \int_{E\times\mathbb{R}_+} \mathbf{1}_{\{y\le g(\lambda_{t-})\}}\eta(e)\,N(dt,de,dy).$$

Price impact

- **Market orders move the price**; limit orders and cancellations only change liquidity
- Price dynamics:

$$dP_t = I(\Delta_t M, \lambda_{t-})$$

- Price impact of an order of size Δ:

$$I(\Delta, \lambda) = \operatorname{sgn}(\Delta) \int_0^{|\Delta|} \iota(\lambda - z)\, dz$$

- Interpretation:
  - **buy** market orders **push the price up**
  - **sell** market orders **push the price down**
  - lower liquidity means larger price impact

The trader in the model

- After the market model is defined, we introduce an **active trader**
- The trader can send **market orders** and therefore also affects liquidity and price
- The state of the trader and the market is

$$S_t^C = (\lambda_t^C, Q_t^C, P_t^C, X_t^C)$$

  - λ_t^C: liquidity in the book
  - Q_t^C: trader's inventory
  - P_t^C: asset price
  - X_t^C: cash position

- The trader's problem is to decide when to buy, sell, or wait
- Here C denotes the trader's strategy, i.e. when and how much to trade

Signals and event timing

- Before an external order arrives, the trader may observe a short-term signal
- The benchmark example in the paper:

$$Z_t \in \{-1, 1\}$$

  - $Z_t = -1$: **liquidity-taking** event is coming
  - $Z_t = 1$: **liquidity-providing** event is coming

- At an event time, the model follows the order:

$$\textit{signal} \to \textit{signal-based trade} \to \textit{external order} \to \textit{state-based trade}$$

- This is what makes the signal valuable: the trader can react before the external order changes the market

Objective and value function

- The trader evaluates the final result through **terminal wealth** W_T^C
- Terminal wealth depends on cash, inventory value and liquidation costs

$$W_T^C \approx X_T^C + Q_T^C P_T^C - \text{liquidation costs}$$

- The trader's **utility function** is

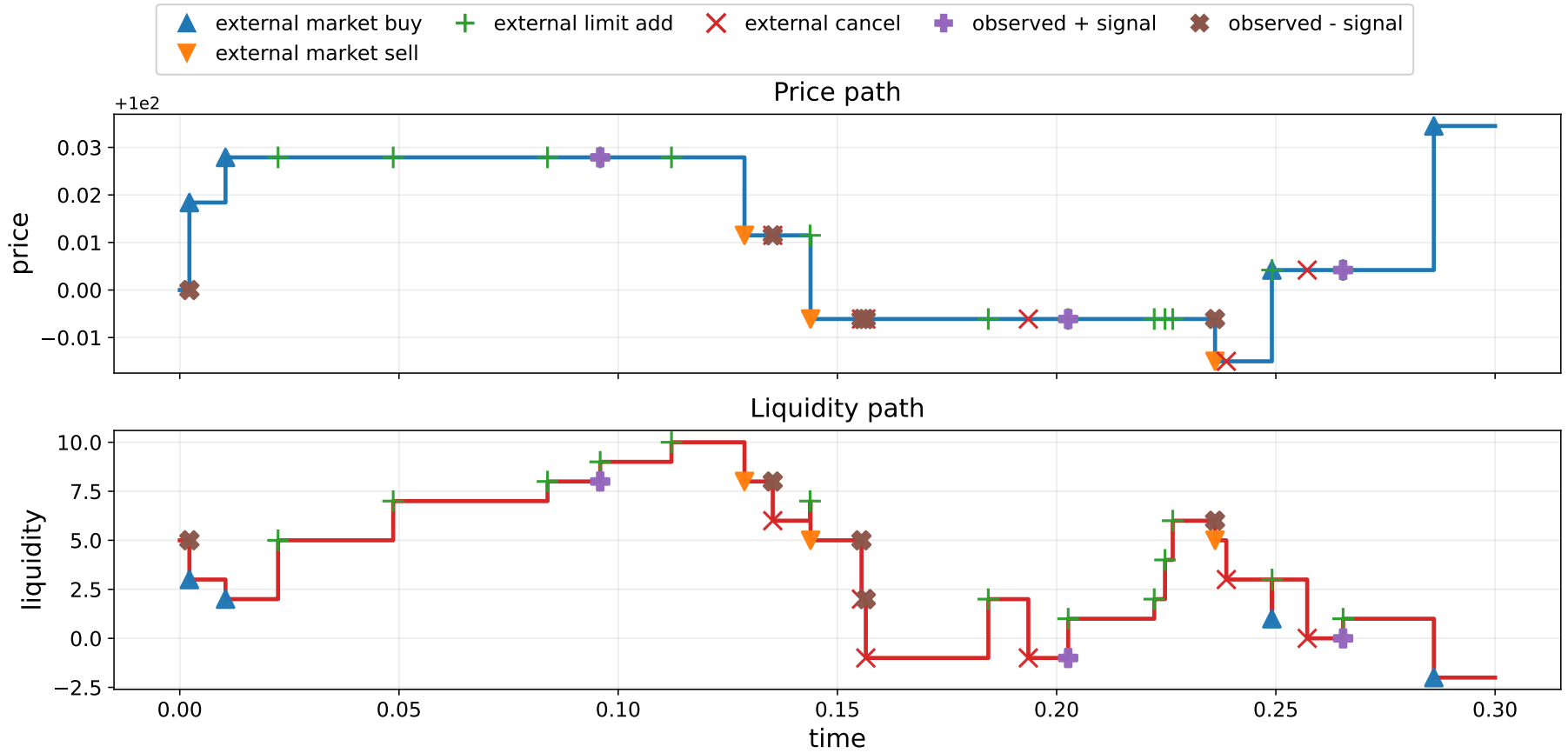
$$U(W_T^C) = -\exp(-\alpha W_T^C)$$

- The **value function** answers the question:

What is the best expected utility starting from this state?

$$v(\tau, \lambda, q, p, x) = \sup_C \mathbb{E}\left[U(W_T^C)\right] \qquad \tau = \text{time-to-go}$$

One simulated path



- The upper panel shows the price path, the lower panel shows liquidity

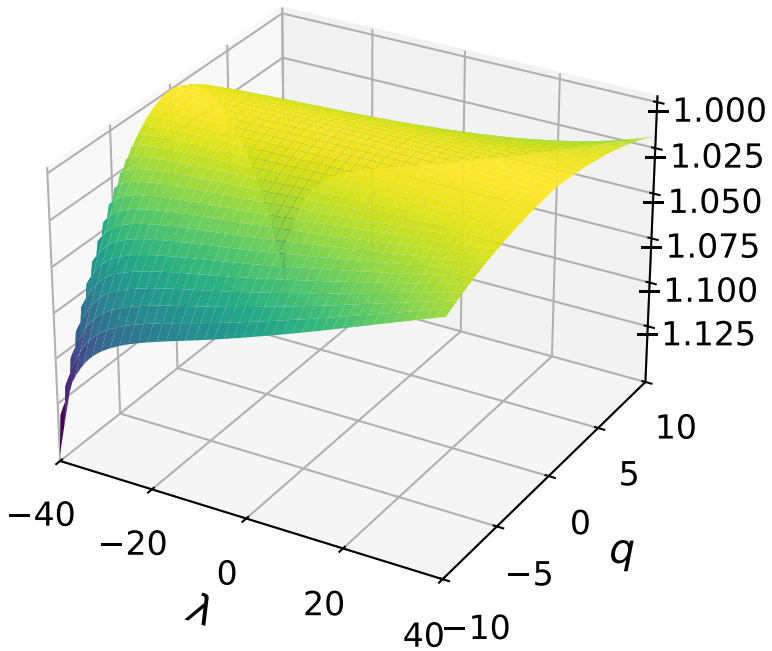
From value function to benchmark strategy

- The goal is to find the **optimal trading strategy**:

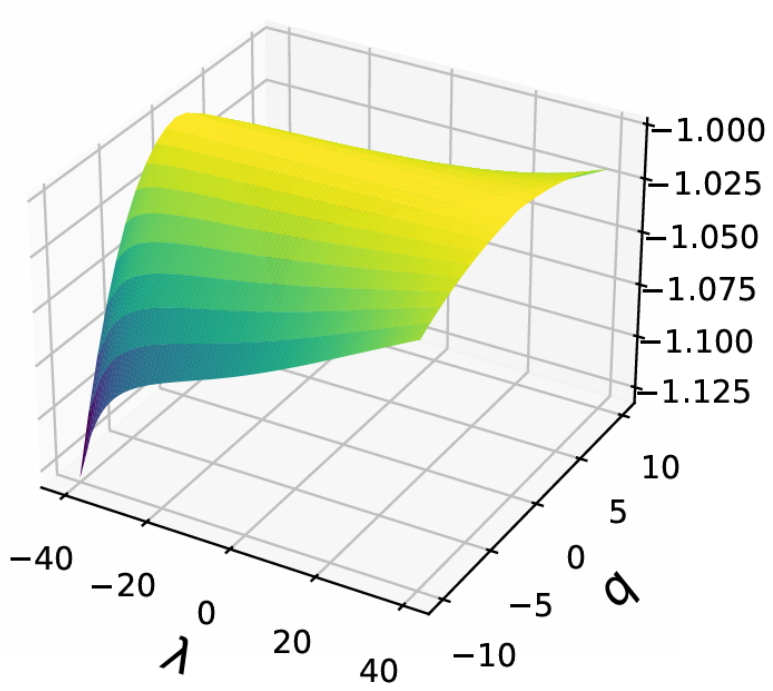
  in each market state, should the trader buy, sell, or wait?

- This is done through the value function $v(\tau, \lambda, q, p, x)$
- The paper derives a Hamilton–Jacobi–Bellman Quasi-Variational Inequality (HJBQVI) for this value function
- This equation cannot be solved analytically in this model, so I implemented the numerical scheme from the paper in Python
- The numerical solution gives:
  - the **value function**
  - and the **optimal strategy**

Value function comparison



(a) My HJBQVI solver



(b) Reference figure from the paper

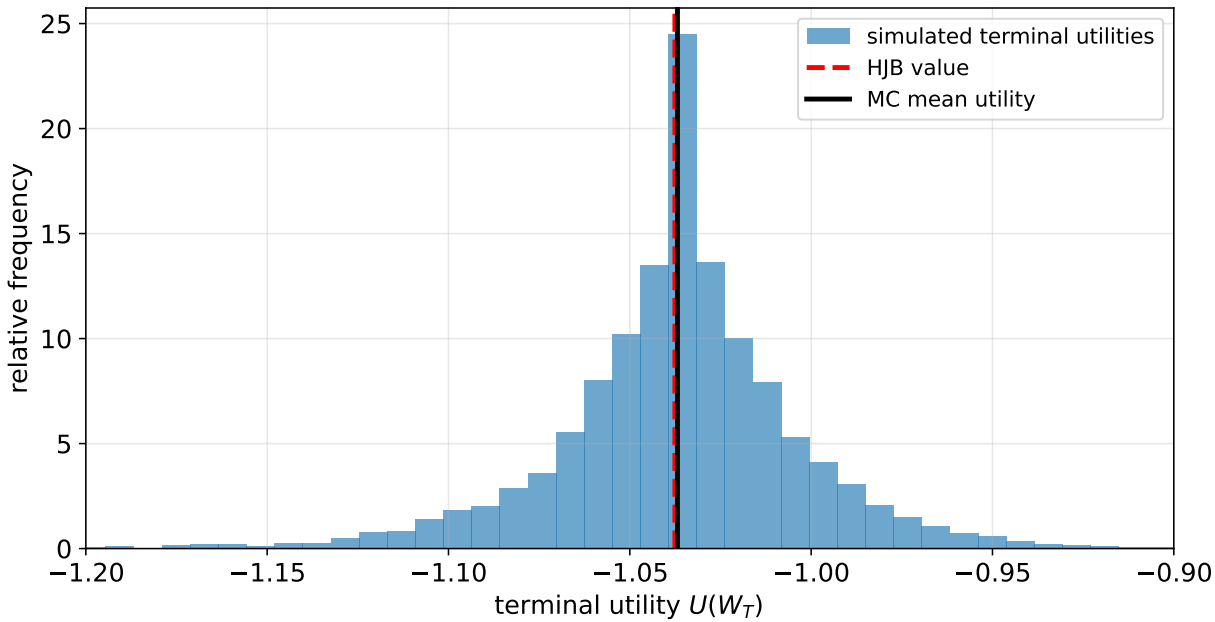
- Qualitative check of the numerical HJBQVI implementation
- The value function surface is consistent with the reference figure

Monte Carlo validation idea

- The HJBQVI solver gives two outputs:
  - a numerical value function
  - a benchmark trading policy
- The benchmark policy is the strategy obtained from the HJB solution
- Validation idea:
  - simulate many paths using this benchmark policy
  - compute the average terminal utility
  - compare it with the HJB value

$$v(T, \lambda_0, q_0, p_0, x_0) \approx \frac{1}{N} \sum_{i=1}^{N} U(W_T^{(i)})$$

Monte Carlo validation result



- Initial state:

$$\lambda_0 = 0, \qquad q_0 = 8, \qquad p_0 = 100, \qquad x_0 = -800$$

- Number of simulated paths: $N = 5000$
- The HJB value lies inside the 95% Monte Carlo confidence interval

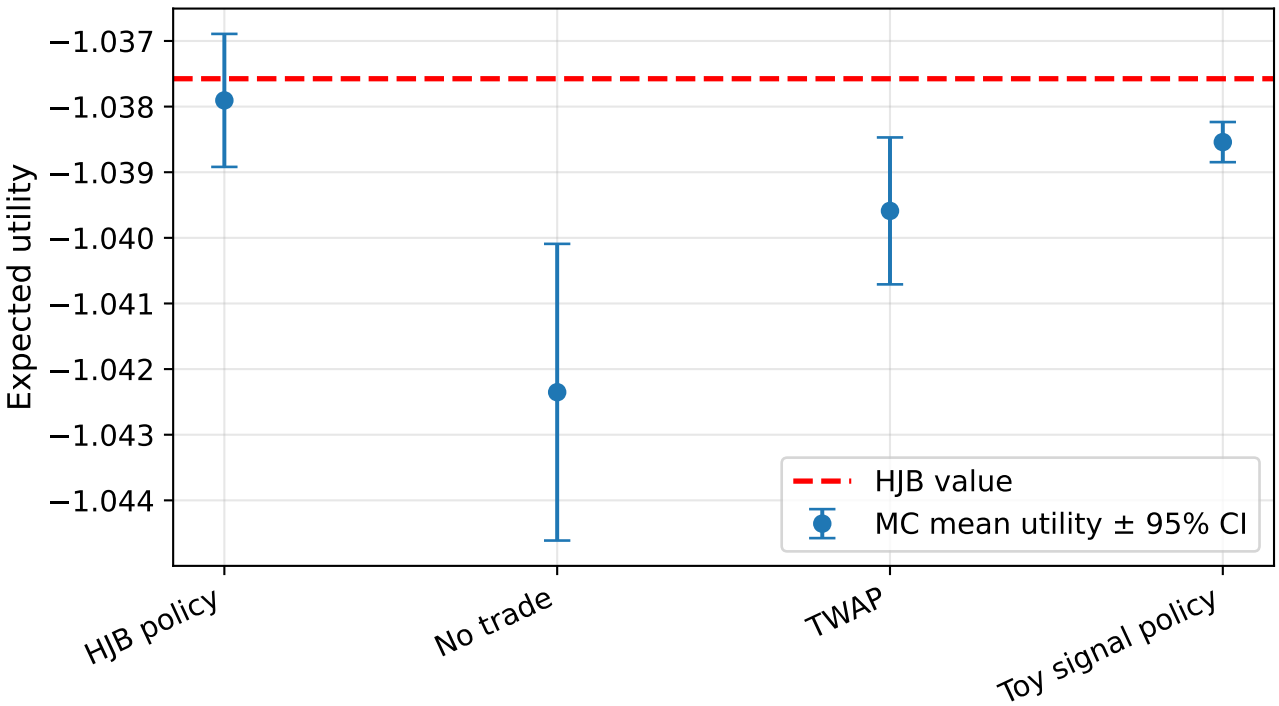
Policies used for comparison

- After validating the HJB benchmark, I compared it with simpler strategies
- The tested policies were:
  - HJB benchmark policy
  - no-trade policy
  - Time-Weighted Average Price (TWAP) policy
  - simple signal-aware toy policy
- For each policy, I simulated many paths and estimated

$$\mathbb{E}[U(W_T)]$$

- This shows how close simple strategies are to the HJB benchmark

Policy comparison result



- Higher expected utility is better, so values closer to zero are preferred
- The HJB benchmark policy is closest to the HJB value
- The no-trade policy performs clearly worse
- TWAP and the toy policy are closer, but still below the HJB benchmark

Next step: reinforcement learning

- The simulator can be used as a **Reinforcement Learning** (RL) environment
- The main goal would be to **train an agent** that learns trading decisions from simulation
- The agent would observe the current market and trader state:
  - time-to-go, liquidity, inventory
  - price, cash and observed signal
- At each decision time, the **agent chooses a trade size**
- The performance can be measured by terminal utility

$$U(W_T) = -\exp(-\alpha W_T)$$

- The learned policy can then be compared to the HJB benchmark policy

Thank you for your attention!

References

[1] Peter Bank, Álvaro Cartea, and Laura Körber. Optimal execution and speculation with trade signals. *arXiv preprint arXiv:2306.00621*, 2024. URL `https://arxiv.org/abs/2306.00621`.

Use of Artificial Intelligence Tools

- During the project, I used ChatGPT and Gemini as supporting tools
- They were used to help with:
  - understanding parts of the mathematical model
  - structuring and debugging the Python implementation
  - improving the wording of the written report and presentation