The Avellaneda-Stoikov Model for High-Frequency Trading in Limit Order Books

Siyi Lyu

January 22, 2025

Abstract

High-frequency trading (HFT) in electronic financial markets relies heavily on the structure of the Limit Order Book (LOB), where market makers provide liquidity by submitting bid and ask quotes. This paper explores the challenges faced by market makers, particularly inventory risk, and formulates a utility-based framework for optimal quote placement. By integrating exponential utility functions with stochastic models for mid-price dynamics and Poisson processes for order arrivals, the study derives closed-form expressions for reservation bid and ask prices. The trading intensity model incorporating market order size distributions and price impact is developed to enhance profitability and risk management. Numerical simulations compare inventory-adjusted strategies against symmetric quoting strategies under varying risk aversion levels. The results demonstrate that inventory-aware strategies mitigate risk and achieve more consistent profits, aligning with findings from existing high-frequency trading research.

1 Introduction

In modern financial markets, the *Limit Order Book* (LOB) serves as the backbone of electronic trading platforms, providing a transparent and efficient mechanism for matching buyers and sellers. The LOB records all outstanding buy and sell limit orders, ranked by price and time priority, allowing market participants to interact dynamically with each other. For market makers—entities traditionally responsible for providing liquidity—the LOB presents both opportunities and challenges. By quoting bid and ask prices, market makers facilitate trading, ensuring that assets remain liquid and that market participants can transact efficiently [1; 2]. As discussed by Avellaneda and Stoikov [11], the role of market makers has evolved significantly with the advent of high-frequency trading and algorithmic strategies. Historically, market makers were specialized firms, but in contemporary electronic exchanges, any participant willing to submit limit orders can effectively act as a market maker. A key challenge for market makers lies in managing their inventory, as holding a large or imbalanced position can expose them to significant price risk. This is known as *inventory risk*, which arises when adverse price movements reduce the profitability of the assets held [3; 4].

Market makers rely on dynamic models that balance profitability and risk to navigate this challenge. A key tool in this context is the use of *utility functions*, which quantify the trade-off between expected returns and risk tolerance. Exponential utility functions are particularly popular, as they encapsulate the preferences of risk-averse agents and enable the computation of optimal bid and ask quotes [6; 7]. The optimization framework for market makers builds on the reservation price model introduced by Avellaneda and Stoikov [11], where bid and ask quotes are adjusted dynamically based on inventory risk. This model, which combines utility optimization with the microstructural properties of the LOB, has become a cornerstone of high-frequency trading research. Liquidity provision, the core business model for market makers, involves dynamically setting bid and ask prices that balance profitability and inventory risk. Market makers earn profits primarily from the *bid-ask spread*, but excessive positions expose them to adverse price movements. Effective inventory management is therefore crucial for long-term profitability [8].

This paper reproduces the framework of Avellaneda and Stoikov [11] by incorporating a refined model for trading intensity that accounts for the distribution of market order sizes and their market impact. By modeling the arrival rates of buy and sell orders as functions of the distance from the mid-price, we derive optimal bid and ask quotes that balance inventory control and market dynamics [14; 10]. Additionally, we also reproduces a utility-based optimization approach that dynamically adjusts quotes to mitigate inventory risk under stochastic price dynamics. The remainder of the paper is organized as follows. Section 2 describes the modeling framework, including the dynamics of the mid-price, utility maximization objectives, and order arrival processes. Section 3 presents numerical simulations to compare our proposed strategy with benchmark strategies, highlighting the impact of inventory risk and order flow dynamics on profitability.

2 The Model

Notation. Throughout this paper, we use the following notation:

- S_t: The mid-price of the stock at time t.
- δ^b, δ^a: Bid and ask offsets from the mid-price.
- k, K: Parameters for trading intensity, with $k = \alpha K$.
- γ: Risk aversion coefficient in the utility function.

2.1 Poisson Processes and Exponential Utility in the Limit Order Book

The LOB is a key structure in electronic financial markets, recording all outstanding buy and sell orders [11; 4]. Its dynamics are stochastic due to order arrivals, executions, and cancellations, which can be modeled as *Poisson processes.* The arrival rates of buy and sell orders are denoted as $\lambda_b(\delta_b)$ and $\lambda_a(\delta_a)$, where $\delta_b = s - p_b$ and $\delta_a = p_a - s$ represent the offsets from the mid-price s. These intensities are commonly modeled as:

$$\lambda_b(\delta_b) = A_b e^{-k\delta_b}, \quad \lambda_a(\delta_a) = A_a e^{-k\delta_a}, \tag{1}$$

where A_b, A_a are scaling factors, and k controls how rapidly execution probabilities decay with offset.

Market makers, as risk-averse agents, aim to maximize their expected utility rather than profits. Exponential utility functions are particularly popular, as they encapsulate the preferences of risk-averse agents and enable the computation of optimal bid and ask quotes [7; 6]. The exponential utility function:

$$U(X) = -e^{-\gamma X}, \tag{2}$$

with risk aversion coefficient $\gamma > 0$, captures their preferences. For initial wealth x and inventory q, the terminal wealth is $X_T = x + qS_T$. The objective is to maximize:

$$\mathbb{E}[U(X_T)] = \mathbb{E}\left[-e^{-\gamma(x+qS_T)}\right]. \tag{3}$$

This framework combines order arrival rates and utility optimization, forming the basis for determining optimal bid and ask quotes.

2.2 The Mid-Price of the Stock

The *mid-price* S_t, defined as the average of the best bid and ask prices:

$$S_t = \frac{p_b + p_a}{2}, \tag{4}$$

is critical for market-making strategies. Its dynamics are modeled as a stochastic process:

$$dS_t = \mu \, dt + \sigma \, dW_t, \tag{5}$$

where μ is the drift term, $\sigma > 0$ is the volatility, and W_t is a standard Brownian motion. Market makers set their quotes around S_t:

$$p_b = S_t - \delta_b, \quad p_a = S_t + \delta_a. \tag{6}$$

Offset sizes δ_b, δ_a balance execution probability with risk. Execution probabilities decrease exponentially with offsets, as described in Section 2.1. Adjustments to δ_b, δ_a depend on inventory levels and stochastic mid-price movements, ensuring profitability while controlling risk.

2.3 The Optimizing Agent with Finite Horizon

The agent's objective is to maximize the expected exponential utility of their P&L profile at a terminal time T. This choice of convex risk measure is convenient, as it allows us to define reservation (or indifference) prices that are independent of the agent's wealth.

We first model an inactive trader who does not have any limit orders in the market and simply holds an inventory of q stocks until the terminal time T. This "frozen inventory" strategy will later prove to be useful in the case when limit orders are allowed. The agent's value function is given by:

$$v(x, s, q, t) = \mathbb{E}_t \left[-\exp\big(-\gamma(x + qS_T) \big) \right], \tag{7}$$

where x is the initial wealth in dollars. This value function can be written as:

$$v(x, s, q, t) = -\exp(-\gamma x)\exp(-\gamma q s)\exp\left(\frac{\gamma^2 q^2 \sigma^2 (T-t)}{2} \right), \tag{8}$$

which shows its dependence on market parameters.

We now define the reservation bid and ask prices for the agent. The reservation bid price r^b is the price that would make the agent indifferent between holding their current portfolio and their current portfolio plus one stock. Similarly, the reservation ask price r^a is defined as the price that would make the agent indifferent between holding their portfolio and selling one stock. These are given by:

$$v(x - r^b(s, q, t), s, q + 1, t) = v(x, s, q, t), v(x + r^a(s, q, t), s, q - 1, t) = v(x, s, q, t). \tag{9}$$

Using the expressions for $v(x, s, q, t)$, a simple computation yields the closed-form solutions for the reservation prices:

$$r^a(s, q, t) = s + (1 - 2q)\frac{\gamma \sigma^2 (T-t)}{2}, r^b(s, q, t) = s + (-1 - 2q)\frac{\gamma \sigma^2 (T-t)}{2}. \tag{10}$$

In the setting where no trading is allowed, the average of these two prices is referred to as the reservation or indifference price:

$$r(s, q, t) = s - q\gamma\sigma^2(T-t). \tag{11}$$

This indifference price reflects an adjustment to the mid-price based on the agent's inventory q. If the agent is long stock ($q > 0$), the reservation price is below the mid-price, indicating a desire to sell stock to reduce inventory. Conversely, if the agent is short stock ($q < 0$), the reservation price is above the mid-price, reflecting a willingness to buy stock at a higher price.

2.4 Limit Orders

The infinite-horizon objective introduced in Section 2.3 involves dynamically adjusting bid and ask quotes. Limit orders are set around the mid-price S_t, with the bid price p^b and ask price p^a defined as:

$$p^b = S_t - \delta^b, \quad p^a = S_t + \delta^a, \tag{12}$$

where δ^b and δ^a represent the bid and ask offsets. These offsets capture the trade-off between execution probability and inventory control. Execution priority is determined by the relative position of quotes in the limit order book, with quotes closer to S_t being executed first. The inventory q_t and wealth X_t evolve dynamically with executed orders, modeled as:

$$q_t = N_t^b - N_t^a, \quad dX_t = p^a dN_t^a - p^b dN_t^b, \tag{13}$$

where N_t^b and N_t^a are cumulative counts of executed buy and sell orders. By dynamically adjusting δ^b and δ^a in response to market conditions and inventory levels, the agent ensures an optimal balance between execution probability and profitability.

2.5 The Trading Intensity

The execution rate of limit orders depends on the distance δ from the mid-price S_t, modeled as an exponential decay function:

$$\lambda(\delta) = A\exp(-k\delta), \tag{14}$$

where A is the baseline execution rate and k determines sensitivity to δ. This behavior can be derived by combining the power-law distribution of order sizes with market impact models. Empirical studies show that market order sizes often follow a power-law distribution:

$$f^Q(x) \propto x^{-1-\alpha}, \tag{15}$$

where α reflects the frequency of large orders. The price impact of an order of size Q is given by:

$$\Delta p \propto \ln(Q), \tag{16}$$

depending on the market model. Substituting these relationships, the execution probability beyond δ becomes:

$$\begin{aligned} \lambda(\delta) &= \Lambda P(\Delta p > \delta) \\ &= \Lambda P(\ln(Q) > K\delta) \\ &= \Lambda P(Q > \exp(K\delta)) \\ &= \Lambda \int_{\exp(K\delta)}^{\infty} x^{-1-\alpha}\, dx \\ &= A\exp(-k\delta) \end{aligned} \tag{17}$$

where $A = \frac{\Lambda}{\alpha}$ and $k = \alpha K$.

The parameter k encapsulates both order size distribution and the impact on the market, linking microstructural market properties to execution dynamics. Larger k corresponds to tighter liquidity conditions, while smaller k reflects broader spreads and more relaxed market dynamics.

3 The Solution

In this section, we detail the methodology for deriving optimal quoting strategies under the framework outlined in last section. The solution involves three main components: (1) solving the utility optimization problem to determine optimal bid and ask offsets; (2) incorporating trading intensity functions to account for market dynamics; and (3) implementing a feedback mechanism for real-time adjustment of quoting strategies, which aligns with the work of

3.1 Optimal Bid and Ask Quotes

Recall that the agent's objective is to maximize their utility by determining optimal bid and ask quotes. This can be expressed in terms of the value function:

$$u(s,x,q,t) = \max_{\delta^a,\delta^b} \mathbb{E}\left[-\exp\left(-\gamma(X_T + q_T S_T)\right)\right], \tag{18}$$

where δ^a and δ^b represent the ask and bid quotes, respectively, and γ is the agent's risk aversion parameter. The goal is to find state-dependent, time-varying quotes that maximize expected utility. Using dynamic programming, the value function $u(s,x,q,t)$ satisfies the following Hamilton-Jacobi-Bellman (HJB) equation:

$$\begin{aligned} u_t + \frac{1}{2}\sigma^2 u_{ss} &+ \max_{\delta^b} \lambda^b(\delta^b)\left[u(s,x-s+\delta^b,q+1,t) - u(s,x,q,t)\right] \\ &+ \max_{\delta^a} \lambda^a(\delta^a)\left[u(s,x+s+\delta^a,q-1,t) - u(s,x,q,t)\right] = 0, \end{aligned} \tag{19}$$

with the terminal condition:

$$u(s,x,q,T) = -\exp\left(-\gamma(x+qs)\right). \tag{20}$$

Thanks to the property of the exponential utility function, we have:

$$u(s,x,q,t) = -\exp(-\gamma x)\exp(-\gamma\theta(s,q,t)), \tag{21}$$

where $\theta(s,q,t)$ represents the cost function related to the inventory q. Substituting this ansatz into the HJB equation gives the simplified PDE for $\theta(s,q,t)$:

$$\begin{aligned} &\theta_t + \frac{1}{2}\sigma^2\theta_{ss} - \frac{1}{2}\sigma^2\gamma\theta_s^2 \\ &\quad + \max_{\delta^b} \frac{\lambda^b(\delta^b)}{\gamma}\left[1 - e^{-\gamma(s-\delta^b-r^b)}\right] \\ &\quad + \max_{\delta^a} \frac{\lambda^a(\delta^a)}{\gamma}\left[1 - e^{-\gamma(s+\delta^a-r^a)}\right] = 0. \end{aligned} \tag{22}$$

The terminal condition becomes $\theta(s,q,T) = qs$.

Using the function $\theta(s,q,t)$, we define the reservation bid price r^b and ask price r^a:

$$r^b(s,q,t) = \theta(s,q+1,t) - \theta(s,q,t), \tag{23}$$

$$r^a(s,q,t) = \theta(s,q,t) - \theta(s,q-1,t). \tag{24}$$

The optimal distances δ^b and δ^a from the mid-price are then determined implicitly by:

$$s - r^b(s,q,t) = \delta^b(s,q,t) - \frac{1}{\gamma}\ln\left(1 - \gamma\frac{\lambda^b(\delta^b)}{\partial\lambda^b/\partial\delta^b}\right), \tag{25}$$

$$r^a(s,q,t) - s = \delta^a(s,q,t) - \frac{1}{\gamma}\ln\left(1 - \gamma\frac{\lambda^a(\delta^a)}{\partial\lambda^a/\partial\delta^a}\right). \tag{26}$$

3.2 Incorporating Trading Intensity

The main computational challenge in solving (19) lies in the nonlinear nature of the order arrival terms, which depend on the inventory and must be maximized. To address this, we propose an asymptotic expansion of θ in terms of the inventory variable q and a linear approximation for the order arrival rates. Assuming symmetric exponential arrival rates, we define:

$$\lambda^a(\delta) = \lambda^b(\delta) = Ae^{-k\delta}. \tag{27}$$

Under this assumption, the reservation prices $r^a(s,q,t)$ and $r^b(s,q,t)$ approximate their values using the "frozen inventory" framework as introduced in Sec 2.3. Substituting these into (19), we derive:

$$\theta_t + \frac{1}{2}\sigma^2\theta_{ss} - \frac{1}{2}\sigma^2\gamma\theta_s^2 + \frac{A}{k+\gamma}\left(e^{-k\delta^a} + e^{-k\delta^b}\right) = 0, \quad \theta(s,q,T) = qs. \tag{28}$$

To simplify, we expand θ as a power series in q, then the resulting expressions for the reservation bid and ask prices are:

$$r^b(s,q,t) = \theta^1(s,t) + (1+2q)\theta^2(s,t) + \cdots, \tag{29}$$

$$r^a(s,q,t) = \theta^1(s,t) + (-1+2q)\theta^2(s,t) + \cdots. \tag{30}$$

Substituting these into the optimality conditions, finally, for this linear approximation of the order arrival term, the sensitivity term simplifies to:

$$\theta^2(s,t) = -\frac{1}{2}\sigma^2\gamma(T-t). \tag{31}$$

This leads to the indifference price:

$$r(s,q,t) = s - q\gamma\sigma^2(T-t), \tag{32}$$

and the bid-ask spread:

$$\delta^a + \delta^b = \gamma\sigma^2(T-t) + \frac{2}{\gamma}\ln\left(1+\frac{\gamma}{k}\right). \tag{33}$$

Equations (32) and (33) provide simplified expressions for the bid and ask prices in terms of the model parameters, facilitating the simulations discussed in the next section.

3.3 Strategies Used

In this study, we evaluate the performance of two distinct market-making strategies: the **Inventory Strategy** and the **Symmetric Strategy**. These strategies differ in their approach to inventory management and the placement of bid and ask quotes.

Inventory Strategy. The inventory strategy dynamically adjusts the reservation prices (bid and ask prices) based on the agent's current inventory levels. This approach aims to mitigate inventory risk by modifying the bid-ask spread asymmetrically, depending on whether the agent holds a long or short position. For instance:

- If the agent's inventory is long (i.e., holding excess stocks), the bid price is lowered, and the ask price is raised to encourage selling and discourage further buying.
- Conversely, if the agent's inventory is short (i.e., deficit in stocks), the bid price is raised, and the ask price is lowered to encourage buying and discourage further selling.

This strategy integrates the risk-aversion coefficient γ, the time to terminal horizon $T - t$, and the inventory level q into the quote adjustments, resulting in asymmetrical spreads around the mid-price.

Symmetric Strategy. The symmetric strategy, on the other hand, employs fixed bid and ask quotes centered symmetrically around the mid-price, regardless of inventory levels. This approach assumes that inventory risk can be neglected, and quotes are set purely based on market dynamics. While simpler to implement, the symmetric strategy may lead to higher inventory variance and increased exposure to adverse price movements.

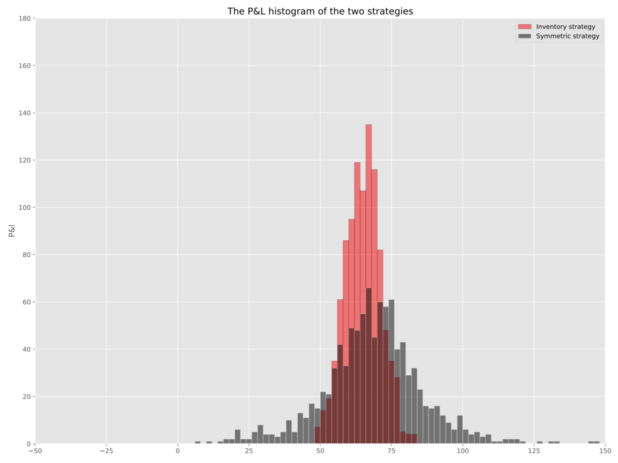
Comparison. The inventory strategy is designed to balance profitability and risk management by actively controlling inventory levels, resulting in more consistent outcomes under varying market conditions. The symmetric strategy, though straightforward, may achieve higher profits in certain scenarios but at the cost of greater variance and potentially higher risk. The numerical simulations in the following section illustrate the differences between these strategies under various levels of risk aversion.

3.4 Numerical Simulations

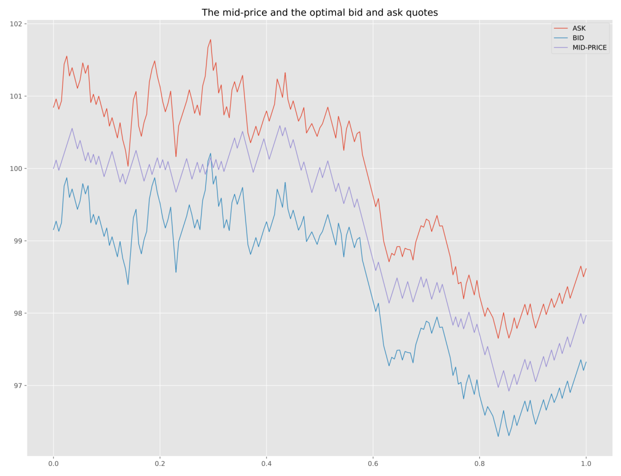
In this section, we evaluate the performance of two market-making strategies: the "inventory strategy" and the "symmetric strategy." The inventory strategy incorporates adjustments based on the agent's current inventory level, while the symmetric strategy keeps its quotes centered around the mid-price regardless of inventory. We analyze their P&L profiles and final inventory distributions under varying market dynamics. The choice of the time step Δt plays a crucial role in the simulation. On the one hand, Δt must be sufficiently small to prevent multiple orders from being executed simultaneously. On the other hand, it must also be large enough to avoid excessive updates to the agent's quotes that would exceed market order arrival rates.

For our experiments, we used the following parameters, the same as these figures in the work of Avellaneda, M., & Stoikov, S.[11]: $s_0 = 100$, $T = 1$, $\sigma = 2$, $\Delta t = 0.005$, $q_0 = 0$, $\gamma = 0.1$, $k = 1.5$, and $A = 140$. The simulation evolves over discrete time steps, where the mid-price is updated by a random increment $\pm\sigma\sqrt{\Delta t}$. For each step, the agent computes optimal quotes based on its strategy, and transactions occur probabilistically based on bid/ask spreads. For the inventory strategy, the reservation price adjusts dynamically to account for the agent's inventory risk, resulting in bid/ask spreads that are often asymmetric. In contrast, the symmetric strategy employs fixed spreads centered around the mid-price, ignoring inventory considerations.

1000 **simulations** ($\gamma = 0.1$) With moderate risk aversion ($\gamma = 0.1$), the "inventory" strategy demonstrates more cautious adjustments to the inventory, resulting in reduced variance compared to the "symmetric" strategy.



(a) P-L histogram ($\gamma = 0.1$)

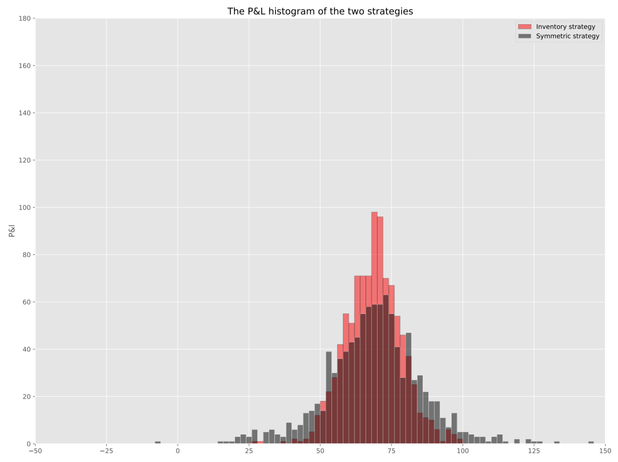


(b) Mid-price and optimal bid and ask quotes ($\gamma = 0.1$)

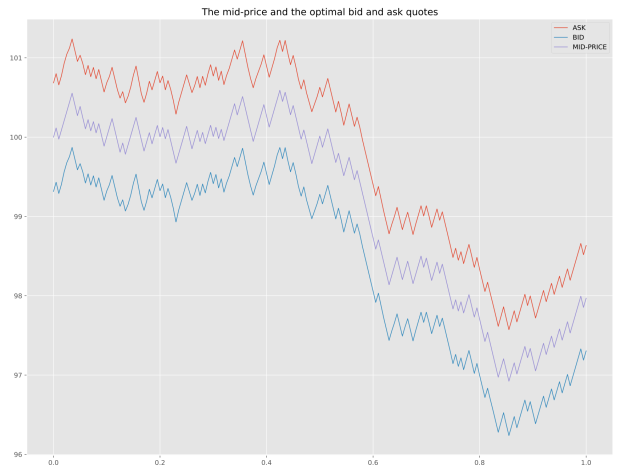
Table 1: Performance Metrics of Inventory and Symmetric Strategies

Metric	**Inventory Strategy**	**Symmetric Strategy**
P&L - Mean	65.005	68.691
P&L - Standard Deviation	6.214	17.995
Inventory - Mean	0.043	0.040
Inventory - Standard Deviation	3.159	8.438
Average Spread	1.498	

1000 **simulations ($\gamma = 0.01$)** For a risk-neutral agent ($\gamma = 0.01$), the inventory effect is minimal. The "inventory" strategy closely resembles the "symmetric" strategy, with minor differences in the P&L and inventory distribution.



(a) P-L histogram ($\gamma = 0.01$)

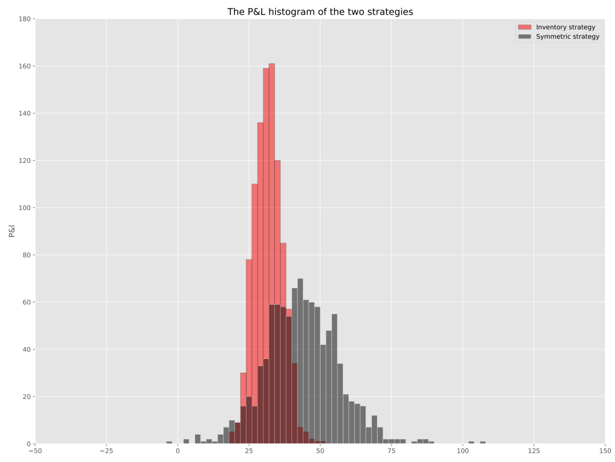


(b) Mid-price and optimal bid and ask quotes ($\gamma = 0.01$)

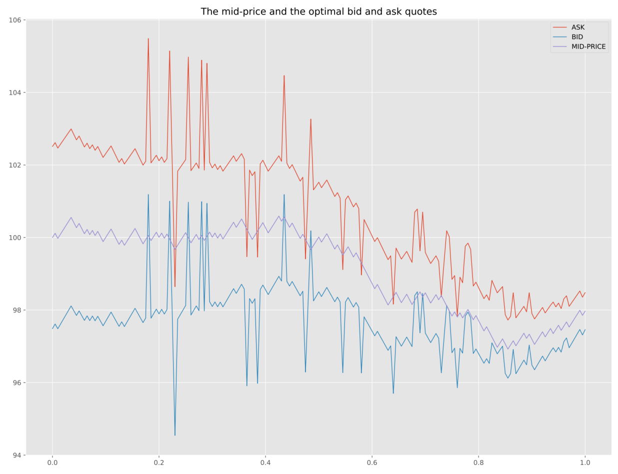
1000 **simulations ($\gamma = 1$)** For a highly risk-averse agent ($\gamma = 1$), the "inventory" strategy prioritizes minimizing inventory exposure. This leads to lower P&L variance but also lower profits compared to the "symmetric" strategy.

Table 2: Performance Metrics of Inventory and Symmetric Strategies

Metric	**Inventory Strategy**	**Symmetric Strategy**
P&L - Mean	68.678	69.109
P&L - Standard Deviation	9.769	17.211
Inventory - Mean	0.066	0.008
Inventory - Standard Deviation	5.363	8.432
Average Spread	1.356	



(a) P-L histogram ($\gamma = 1$)



(b) Mid-price and optimal bid and ask quotes ($\gamma = 1$)

Table 3: Performance Metrics of Inventory and Symmetric Strategies

Metric	**Inventory Strategy**	**Symmetric Strategy**
P&L - Mean	31.794	43.624
P&L - Standard Deviation	4.891	13.034
Inventory - Mean	0.012	0.000
Inventory - Standard Deviation	1.773	5.187
Average Spread	3.037	

3.5 Results Analysis

The results of 1000 simulations for $\gamma = 0.1$ highlight key differences between these strategies, which aligns with previous work [8] [11]. For example, the symmetric strategy typically achieves higher average returns due to its centering around the mid-price, which increases the likelihood of order executions. However, this approach leads to greater variance in P&L outcomes, as shown in Figure1a. By contrast, the inventory strategy achieves more consistent P&L profiles (lower variance), as illustrated in Figures 1b, albeit with slightly lower average returns. Finally, when simulating for $\gamma = 0.01$ and $\gamma = 1$, we observe distinct effects. With $\gamma = 0.01$, the inventory strategy behaves almost identically to the symmetric strategy, as the inventory risk aversion is minimal, showed in Figures 2a and 2b. Conversely, for $\gamma = 1$, the inventory strategy becomes highly risk-averse, leading to narrower spreads and reduced inventory fluctuations at the cost of lower profitability, showed in Figures 3a and 3b. These effects are summarized in the accompanying figures and tables for each scenario, showed in Tables 1, 2, and 3b. We performed 1000 simulations to compare the performance of the "inventory" strategy and the "symmetric" strategy under varying risk aversion levels ($\gamma = 0.01, 0.1, 1$). This result aligns with the work of Avellaneda, M., & Stoikov, S.[11].

References

[1] Biais, B., Hillion, P., & Spatt, C. (1995). An empirical analysis of the limit order book and the order flow in the Paris Bourse. *The Journal of Finance*, 50(5), 1655–1689. DOI:10.1111/j.1540-6261.1995.tb05192.x.

[2] O'Hara, M. (1995). *Market microstructure theory.* Blackwell Publishers.

[3] Foucault, T., Kadan, O., & Kandel, E. (2005). Limit order book as a market for liquidity. *The Review of Financial Studies*, 18(4), 1171–1217. DOI:10.1093/rfs/hhi028.

[4] Hasbrouck, J., & Saar, G. (2013). Low-latency trading. *Journal of Financial Markets*, 16(4), 646–679. DOI:10.1016/j.finmar.2013.05.003.

[5] Menkveld, A. J. (2013). High-frequency trading and the new market makers. *Journal of Financial Markets*, 16(4), 712–740. DOI:10.1016/j.finmar.2013.06.006.

[6] Ho, T., & Stoll, H. R. (1981). Optimal dealer pricing under transactions and return uncertainty. *The Journal of Financial Economics*, 9(1), 47–73. DOI:10.1016/0304-405X(81)90020-9.

[7] Garman, M. B. (1976). Market microstructure. *Journal of Financial Economics*, 3(3), 257–275. DOI:10.1016/0304-405X(76)90004-1.

[8] Hollifield, B., Miller, R. A., Sandås, P., & Slive, J. (2006). Estimating the gains from trade in limit-order markets. *The Journal of Finance*, 61(6), 2753–2804. DOI:10.1111/j.1540-6261.2006.01007.x.

[9] Weber, P., & Rosenow, B. (2005). Order book approach to price impact. *Quantitative Finance*, 5(4), 357–364. DOI:10.1080/14697680500399258.

[10] Potters, M., & Bouchaud, J.-P. (2003). More statistical properties of order books and price impact. *Physica A: Statistical Mechanics and its Applications*, 324(1-2), 133–140. DOI:10.1016/S0378-4371(03)00036-7.

[11] Avellaneda, M., & Stoikov, S. (2008). High-frequency trading in a limit order book. *Quantitative Finance*, 8(3), 217–224. DOI:10.1080/14697680701381228.

[12] Gopikrishnan, P., Plerou, V., Gabaix, X., & Stanley, H. E. (2000). *Statistical properties of share volume traded in financial markets.* Physical Review E, 62(4), R4493–R4496. APS.

[13] Maslow, S., & Mills, T. C. (2001). *Investor behavior and limit order book dynamics: Empirical evidence from the NASDAQ.* Empirical Economics, 26(4), 857–888. Springer.

[14] Weber, P., & Rosenow, B. (2005). *Order book approach to price impact.* Quantitative Finance, 5(4), 357–364. Taylor & Francis.

[15] Potters, M., & Bouchaud, J.-P. (2003). *More statistical properties of order books and price impact.* Physica A: Statistical Mechanics and its Applications, 324(1-2), 133–140. Elsevier.

Appendix: Python Code

Here is the Python code used in this analysis:

Listing 1: High-frequency trading simulation code$_0$.01

```python
import math
import numpy as np
import matplotlib.pyplot as plt
import os
import random

#
cur_dir = os.path.dirname(os.path.realpath(__file__))
```

```
os.chdir(cur_dir)
plt.style.use('ggplot')
np.random.seed(123)

#
S0 = 100.0  #
T = 1.0  #
sigma = 2.0  #
dt = 0.005  #
M = int(T / dt)  #
Sim = 1000  #
gamma = 0.01  #
k = 1.5  #
A = 140  #
q0 = 0  #

#
inventory_s1 = [q0] * Sim
pnl_s1 = [0] * Sim
inventory_s2 = [q0] * Sim
pnl_s2 = [0] * Sim

#
price_a = [0] * (M + 1)
price_b = [0] * (M + 1)
midprice = [0] * (M + 1)
#
total_spread = 0  #
#
sym_spread = 0
for i in np.arange(0, T, dt):
    sym_spread += gamma * sigma**2 * (T - i) + (2 / gamma) * np.log(1 + (gamma / k
        ))
avg_sym_spread = sym_spread / M
prob = A * np.exp(-k * avg_sym_spread / 2) * dt

#
for i_sim in range(Sim):
    white_noise = sigma * math.sqrt(dt) * np.random.choice([0.82, -1], M)
    price_process = S0 + np.cumsum(white_noise)
    price_process = np.insert(price_process, 0, S0)  #

    for step, s in enumerate(price_process):
        #
        reservation_price = s - inventory_s1[i_sim] * gamma * sigma**2 * (T - step
             * dt)
        spread = gamma * sigma**2 * (T - step * dt) + (2 / gamma) * math.log(1 + (
            gamma / k))
        spread /= 2

        #
        total_spread += 2 * spread  #           bid_spread + ask_spread

        if reservation_price >= s:
            ask_spread = spread + (reservation_price - s)
            bid_spread = spread - (reservation_price - s)
        else:
            ask_spread = spread - (s - reservation_price)
            bid_spread = spread + (s - reservation_price)

        ask_prob = A * np.exp(-k * ask_spread) * dt
        bid_prob = A * np.exp(-k * bid_spread) * dt
        ask_prob = max(0, min(ask_prob, 1))
        bid_prob = max(0, min(bid_prob, 1))

        ask_action_s1 = np.random.choice([1, 0], p=[ask_prob, 1 - ask_prob])
        bid_action_s1 = np.random.choice([1, 0], p=[bid_prob, 1 - bid_prob])

        inventory_s1[i_sim] -= ask_action_s1
        pnl_s1[i_sim] += ask_action_s1 * (s + ask_spread)
        inventory_s1[i_sim] += bid_action_s1
        pnl_s1[i_sim] -= bid_action_s1 * (s - bid_spread)

        #
        if i_sim == 0:
            price_a[step] = s + ask_spread
            price_b[step] = s - bid_spread
            midprice[step] = s

        #
        ask_action_s2 = np.random.choice([1, 0], p=[prob, 1 - prob])
        bid_action_s2 = np.random.choice([1, 0], p=[prob, 1 - prob])
        inventory_s2[i_sim] -= ask_action_s2
        pnl_s2[i_sim] += ask_action_s2 * (s + avg_sym_spread / 2)
        inventory_s2[i_sim] += bid_action_s2
        pnl_s2[i_sim] -= bid_action_s2 * (s - avg_sym_spread / 2)

    pnl_s1[i_sim] += inventory_s1[i_sim] * s
    pnl_s2[i_sim] += inventory_s2[i_sim] * s

#
average_spread = total_spread / (Sim * M)

# PnL
x_range = [-50, 150]
y_range = [0, 180]
plt.figure(figsize=(16, 12), dpi=100)
bins = np.arange(-50, 150, 2)
plt.hist(pnl_s1, bins=bins, alpha=0.5, label="Inventory␣strategy", color="red", edgecolor="black")
plt.hist(pnl_s2, bins=bins, alpha=0.5, label="Symmetric␣strategy", color="black", edgecolor="white")
plt.ylabel('P&l')
plt.legend()
plt.axis(x_range + y_range)
plt.title("The␣P&L␣histogram␣of␣the␣two␣strategies")
plt.savefig('pnl.pdf', bbox_inches='tight', dpi=100, format='pdf')
plt.show()

#
x = np.linspace(0, T, M + 1)
plt.figure(figsize=(16, 12), dpi=100)
plt.plot(x, price_a, linewidth=1.0, linestyle="-", label="ASK")
plt.plot(x, price_b, linewidth=1.0, linestyle="-", label="BID")
plt.plot(x, midprice, linewidth=1.0, linestyle="-", label="MID-PRICE")
plt.legend()
plt.title("The␣mid-price␣and␣the␣optimal␣bid␣and␣ask␣quotes")
plt.savefig('prices.pdf', bbox_inches='tight', dpi=100, format='pdf')
plt.show()
#
print("P&L␣-␣Mean␣of␣the␣inventory␣strategy:␣{}".format(np.array(pnl_s1).mean()))
print("P&L␣-␣Mean␣of␣the␣symmetric␣strategy:␣{}".format(np.array(pnl_s2).mean()))
print("P&L␣-␣Standard␣deviation␣of␣the␣inventory␣strategy:␣{}".format(np.sqrt(np.
    array(pnl_s1).var())))
print("P&L␣-␣Standard␣deviation␣of␣the␣symmetric␣strategy:␣{}".format(np.sqrt(np.
    array(pnl_s2).var())))
print("INV␣-␣Mean␣of␣the␣inventory␣strategy:␣{}".format(np.array(inventory_s1).
    mean()))
print("INV␣-␣Mean␣of␣the␣symmetric␣strategy:␣{}".format(np.array(inventory_s2).
    mean()))
print("INV␣-␣Standard␣deviation␣of␣the␣inventory␣strategy:␣{}".format(np.sqrt(np.
    array(inventory_s1).var())))
print("INV␣-␣Standard␣deviation␣of␣the␣symmetric␣strategy:␣{}".format(np.sqrt(np.
    array(inventory_s2).var())))
print("Average␣spread:␣{}".format(average_spread))
```

Listing 2: High-frequency trading simulation code$_1$

```
import math
import numpy as np
import matplotlib.pyplot as plt
import os
import random

#
cur_dir = os.path.dirname(os.path.realpath(__file__))
os.chdir(cur_dir)
plt.style.use('ggplot')
np.random.seed(123)

#
S0 = 100.0  #
T = 1.0  #
sigma = 2.0  #
dt = 0.005  #
M = int(T / dt)  #
Sim = 1000  #
gamma = 1  #
k = 1.5  #
A = 140  #
q0 = 0  #

#
inventory_s1 = [q0] * Sim
pnl_s1 = [0] * Sim
inventory_s2 = [q0] * Sim
pnl_s2 = [0] * Sim

#
price_a = [0] * (M + 1)
price_b = [0] * (M + 1)
midprice = [0] * (M + 1)
#
total_spread = 0  #
#
sym_spread = 0
for i in np.arange(0, T, dt):
    sym_spread += gamma * sigma**2 * (T - i) + (2 / gamma) * np.log(1 + (gamma / k))
avg_sym_spread = sym_spread / M
prob = A * np.exp(-k * avg_sym_spread / 2) * dt

#
for i_sim in range(Sim):
    white_noise = sigma * math.sqrt(dt) * np.random.choice([0.82, -1], M)
    price_process = S0 + np.cumsum(white_noise)
    price_process = np.insert(price_process, 0, S0)  #

    for step, s in enumerate(price_process):
        #
        reservation_price = s - inventory_s1[i_sim] * gamma * sigma**2 * (T - step * dt)
        spread = gamma * sigma**2 * (T - step * dt) + (2 / gamma) * math.log(1 + (gamma / k))
        spread /= 2

        #
        total_spread += 2 * spread  #           bid_spread + ask_spread

        if reservation_price >= s:
            ask_spread = spread + (reservation_price - s)
            bid_spread = spread - (reservation_price - s)
        else:
            ask_spread = spread - (s - reservation_price)
            bid_spread = spread + (s - reservation_price)

        ask_prob = A * np.exp(-k * ask_spread) * dt
        bid_prob = A * np.exp(-k * bid_spread) * dt
        ask_prob = max(0, min(ask_prob, 1))
        bid_prob = max(0, min(bid_prob, 1))

        ask_action_s1 = np.random.choice([1, 0], p=[ask_prob, 1 - ask_prob])
        bid_action_s1 = np.random.choice([1, 0], p=[bid_prob, 1 - bid_prob])

        inventory_s1[i_sim] -= ask_action_s1
        pnl_s1[i_sim] += ask_action_s1 * (s + ask_spread)
        inventory_s1[i_sim] += bid_action_s1
        pnl_s1[i_sim] -= bid_action_s1 * (s - bid_spread)

        #
        if i_sim == 0:
            price_a[step] = s + ask_spread
            price_b[step] = s - bid_spread
            midprice[step] = s

        #
        ask_action_s2 = np.random.choice([1, 0], p=[prob, 1 - prob])
        bid_action_s2 = np.random.choice([1, 0], p=[prob, 1 - prob])
        inventory_s2[i_sim] -= ask_action_s2
        pnl_s2[i_sim] += ask_action_s2 * (s + avg_sym_spread / 2)
        inventory_s2[i_sim] += bid_action_s2
        pnl_s2[i_sim] -= bid_action_s2 * (s - avg_sym_spread / 2)

    pnl_s1[i_sim] += inventory_s1[i_sim] * s
    pnl_s2[i_sim] += inventory_s2[i_sim] * s

```

```
#
average_spread = total_spread / (Sim * M)

# PnL
x_range = [-50, 150]
y_range = [0, 180]
plt.figure(figsize=(16, 12), dpi=100)
bins = np.arange(-50, 150, 2)
plt.hist(pnl_s1, bins=bins, alpha=0.5, label="Inventory␣strategy", color="red", edgecolor="black")
plt.hist(pnl_s2, bins=bins, alpha=0.5, label="Symmetric␣strategy", color="black", edgecolor="white")
plt.ylabel('P&l')
plt.legend()
plt.axis(x_range + y_range)
plt.title("The␣P&L␣histogram␣of␣the␣two␣strategies")
plt.savefig('pnl.pdf', bbox_inches='tight', dpi=100, format='pdf')
plt.show()

#
x = np.linspace(0, T, M + 1)
plt.figure(figsize=(16, 12), dpi=100)
plt.plot(x, price_a, linewidth=1.0, linestyle="-", label="ASK")
plt.plot(x, price_b, linewidth=1.0, linestyle="-", label="BID")
plt.plot(x, midprice, linewidth=1.0, linestyle="-", label="MID-PRICE")
plt.legend()
plt.title("The␣mid-price␣and␣the␣optimal␣bid␣and␣ask␣quotes")
plt.savefig('prices.pdf', bbox_inches='tight', dpi=100, format='pdf')
plt.show()
#
print("P&L␣-␣Mean␣of␣the␣inventory␣strategy:␣{}".format(np.array(pnl_s1).mean()))
print("P&L␣-␣Mean␣of␣the␣symmetric␣strategy:␣{}".format(np.array(pnl_s2).mean()))
print("P&L␣-␣Standard␣deviation␣of␣the␣inventory␣strategy:␣{}".format(np.sqrt(np.array(pnl_s1).var())))
print("P&L␣-␣Standard␣deviation␣of␣the␣symmetric␣strategy:␣{}".format(np.sqrt(np.array(pnl_s2).var())))
print("INV␣-␣Mean␣of␣the␣inventory␣strategy:␣{}".format(np.array(inventory_s1).mean()))
print("INV␣-␣Mean␣of␣the␣symmetric␣strategy:␣{}".format(np.array(inventory_s2).mean()))
print("INV␣-␣Standard␣deviation␣of␣the␣inventory␣strategy:␣{}".format(np.sqrt(np.array(inventory_s1).var())))
print("INV␣-␣Standard␣deviation␣of␣the␣symmetric␣strategy:␣{}".format(np.sqrt(np.array(inventory_s2).var())))
print("Average␣spread:␣{}".format(average_spread))
```

Listing 3: High-frequency trading simulation code$_0$.1

```
import math
import numpy as np
import matplotlib.pyplot as plt
import os
import random

#
cur_dir = os.path.dirname(os.path.realpath(__file__))
os.chdir(cur_dir)
plt.style.use('ggplot')
np.random.seed(123)
```

```

#
S0 = 100.0  #
T = 1.0  #
sigma = 2.0  #
dt = 0.005  #
M = int(T / dt)  #
Sim = 1000  #
gamma = 0.1  #
k = 1.5  #
A = 140  #
q0 = 0  #

#
inventory_s1 = [q0] * Sim
pnl_s1 = [0] * Sim
inventory_s2 = [q0] * Sim
pnl_s2 = [0] * Sim
#                      PnL


#
price_a = [0] * (M + 1)
price_b = [0] * (M + 1)
midprice = [0] * (M + 1)

#
total_spread = 0  #

#
sym_spread = 0
for i in np.arange(0, T, dt):
    sym_spread += gamma * sigma**2 * (T - i) + (2 / gamma) * np.log(1 + (gamma / k))
avg_sym_spread = sym_spread / M
prob = A * np.exp(-k * avg_sym_spread / 2) * dt
#                                    Poisson

#
for i_sim in range(Sim):
    white_noise = sigma * math.sqrt(dt) * np.random.choice([0.82, -1], M)
    price_process = S0 + np.cumsum(white_noise)
    price_process = np.insert(price_process, 0, S0)  #

    for step, s in enumerate(price_process):
        #
        reservation_price = s - inventory_s1[i_sim] * gamma * sigma**2 * (T - step * dt)
        spread = gamma * sigma**2 * (T - step * dt) + (2 / gamma) * math.log(1 + (gamma / k))
        spread /= 2

        #
        total_spread += 2 * spread  #           bid_spread + ask_spread

        if reservation_price >= s:
            ask_spread = spread + (reservation_price - s)
            bid_spread = spread - (reservation_price - s)
        else:
            ask_spread = spread - (s - reservation_price)
            bid_spread = spread + (s - reservation_price)

        ask_prob = A * np.exp(-k * ask_spread) * dt
        bid_prob = A * np.exp(-k * bid_spread) * dt
        ask_prob = max(0, min(ask_prob, 1))
        bid_prob = max(0, min(bid_prob, 1))
        #

        ask_action_s1 = np.random.choice([1, 0], p=[ask_prob, 1 - ask_prob]) #

        bid_action_s1 = np.random.choice([1, 0], p=[bid_prob, 1 - bid_prob]) #


        #
        inventory_s1[i_sim] -= ask_action_s1
        pnl_s1[i_sim] += ask_action_s1 * (s + ask_spread)
        inventory_s1[i_sim] += bid_action_s1
        pnl_s1[i_sim] -= bid_action_s1 * (s - bid_spread)

        #
        if i_sim == 0:
            price_a[step] = s + ask_spread
            price_b[step] = s - bid_spread
            midprice[step] = s

        #
        #


        #


        ask_action_s2 = np.random.choice([1, 0], p=[prob, 1 - prob])
        bid_action_s2 = np.random.choice([1, 0], p=[prob, 1 - prob])
        inventory_s2[i_sim] -= ask_action_s2 #
        pnl_s2[i_sim] += ask_action_s2 * (s + avg_sym_spread / 2) #

        inventory_s2[i_sim] += bid_action_s2 #
        pnl_s2[i_sim] -= bid_action_s2 * (s - avg_sym_spread / 2) #
        #
    pnl_s1[i_sim] += inventory_s1[i_sim] * s
    pnl_s2[i_sim] += inventory_s2[i_sim] * s

#
average_spread = total_spread / (Sim * M)

# PnL
x_range = [-50, 150]
y_range = [0, 180]
plt.figure(figsize=(16, 12), dpi=100)
bins = np.arange(-50, 150, 2)
plt.hist(pnl_s1, bins=bins, alpha=0.5, label="Inventory␣strategy", color="red",
    edgecolor="black")
plt.hist(pnl_s2, bins=bins, alpha=0.5, label="Symmetric␣strategy", color="black",
    edgecolor="white")
plt.ylabel('P&l')
plt.legend()
plt.axis(x_range + y_range)
plt.title("The␣P&L␣histogram␣of␣the␣two␣strategies")
plt.savefig('pnl.pdf', bbox_inches='tight', dpi=100, format='pdf')
plt.show()

#
x = np.linspace(0, T, M + 1)
plt.figure(figsize=(16, 12), dpi=100)
plt.plot(x, price_a, linewidth=1.0, linestyle="-", label="ASK")
plt.plot(x, price_b, linewidth=1.0, linestyle="-", label="BID")
plt.plot(x, midprice, linewidth=1.0, linestyle="-", label="MID-PRICE")
plt.legend()
plt.title("The␣mid-price␣and␣the␣optimal␣bid␣and␣ask␣quotes")
plt.savefig('prices.pdf', bbox_inches='tight', dpi=100, format='pdf')
plt.show()
#
print("P&L␣-␣Mean␣of␣the␣inventory␣strategy:␣{}".format(np.array(pnl_s1).mean()))
print("P&L␣-␣Mean␣of␣the␣symmetric␣strategy:␣{}".format(np.array(pnl_s2).mean()))
print("P&L␣-␣Standard␣deviation␣of␣the␣inventory␣strategy:␣{}".format(np.sqrt(np.
    array(pnl_s1).var())))
print("P&L␣-␣Standard␣deviation␣of␣the␣symmetric␣strategy:␣{}".format(np.sqrt(np.
    array(pnl_s2).var())))
print("INV␣-␣Mean␣of␣the␣inventory␣strategy:␣{}".format(np.array(inventory_s1).
    mean()))
print("INV␣-␣Mean␣of␣the␣symmetric␣strategy:␣{}".format(np.array(inventory_s2).
    mean()))
print("INV␣-␣Standard␣deviation␣of␣the␣inventory␣strategy:␣{}".format(np.sqrt(np.
    array(inventory_s1).var())))
print("INV␣-␣Standard␣deviation␣of␣the␣symmetric␣strategy:␣{}".format(np.sqrt(np.
    array(inventory_s2).var())))
print("Average␣spread:␣{}".format(average_spread))
```