# Determine the robustness of the Almgren Chriss Optimal Execution algorithm with stochastic price impact parameters

Candidate Number: 1027764

April 8, 2019

## Contents

## Abstract

We propose a modified Almgren-Chriss model for optimal execution where the permanent and temporary price impact parameters are modelled as Cox-Ingersoll-Ross processes. We simulate the liquidation of a large block of shares using the proposed model and compare its performance to the original Almgren-Chriss strategy. We examine the effect of the urgency parameter, $\phi$, and the terminal liquidation penalty, $\alpha$, on the optimal trading strategy, inventory process and income generated in the liquidation in both models and suggest areas where the modified strategy might perform better. We also experiment with various combinations of parameters for the CIR process to optimise the performance of the proposed model.

## 1 Introduction

An order is classified as large when the number of shares to be traded significantly exceeds the volume of the best bid/ask offer in the limit order book [1]. Institutional investors often face the problem of executing large market orders in a relatively short period of time. In such cases, traders can adopt one of two methods: carry out the trade at the current known price but at a higher cost or execute the trade in smaller packets over a fixed horizon at a lower cost but greater uncertainty due to price risk [4]. Optimal trading strategies aim to find a trade-off between the price uncertainty around long trading periods and the price impact caused by trading too quickly. Trading large market orders quickly can cause the order to walk the book and obtain worse prices for execution. This is a temporary price impact as the order book is resilient and replenishes itself continuously [4]. An excess of sell orders puts a downward pressure on the price of the asset while the acquisition of a large block of shares pushes the asset price up. This is a permanent price impact on the dynamics of the mid-price of the asset. Price impact usually affects traders adversely as lower prices lead to lower revenues while higher prices increase costs [2].
The Almgren-Chriss trading strategy helps the agent to find an optimal trading speed which minimises execution costs. We define execution costs as direct ones like brokerage fees and commission and indirect ones such as price impact and the opportunity cost associated with long trading periods [3]. In the Almgren-Chriss model, the temporary and permanent price impact are linear in the speed of trading. In this paper, we examine the robustness of the Almgren-Chriss model when trading in a market where price impact is stochastic and compare its performance to the original model.
This paper is organised as follows. In Section 2, we define the key stochastic processes required to formulate a trading strategy and propose a model for the dynamics of the temporary and permanent price impact processes. In Section 3, we define the trader's value function and subsequently derive the Almgren-Chriss strategy, for both non-stochastic and stochastic price impact. In Section 4, we run numerical simulations to compare the performance of both trading strategies and examine the effect of different parameters on the trading speed, inventory and income generated. In Section 5, we outline any conclusions we reach from our research.

## 2 The Market Model

Consider an investor who wants to liquidate $N$ shares over a trading horizon $T > 0$ using only market orders. We use the market model outlined in [1] to define the following key stochastic processes over the filtered probability space $(\Omega, \mathcal{F}, \mathcal{F}_{0\leq t\leq T}, \mathcal{P})$:

- The speed at which the shares are being liquidated is $v = (v)_{0\leq t\leq T}$.
  This is the variable that can be controlled in the optimisation so it has to be chosen to maximise income earned from liquidation and minimise execution costs.

- The inventory process is $Q^v = Q^v_{0\leq t\leq T}$ depends on the trading speed and is given by:

$$dQ^v_t = -v_t dt, \quad Q^v_0 = N \tag{1}$$

- The mid-price process is $S^v = S^v_{0\leq t\leq T}$ where the mid-price is defined as the average of the best bid and the best ask.
- The execution price process is $E^v = E^v_{0\leq t\leq T}$ which describes the prices at which the trades are sold, either by walking the limit order book or by breaking the market order into smaller parts.
- The income generated from the liquidation of the shares is $X^v = X^v_{0\leq t\leq T}$. It follows the SDE:

$$dX^v_t = E^v_t v_t dt \tag{2}$$

**Permanent Price Impact**

The evolution of the mid-price is affected by the volatility and drift of the stock as well as the market impact due to the trading actions of the agent [4]. Trading has a direct effect on the equilibrium price of the asset, at least for the life of the liquidation process. Considering our current example, if the trader liquidates a large block of shares in smaller bits, the volume of sell orders will cause other market participants to adjust their bids and thus exert a downward pressure on the mid-price [4]. The mid-price evolves according to:

$$dS^v_t = -g(v_t)dt + \sigma dW_t, \quad S^v_0 = S_0 \tag{3}$$

where $W_t$ is a standard brownian motion driving the stock price, $\sigma$ is the volatility of the stock, $S_0$ is the initial mid-price of the stock and $g(v_t)$ describes the permanent price impact of the trading action on the mid-price process.
The function $g(v_t) : R \to R$ enters the equation with a negative sign and is always positive as the trades exert a permanent downward pressure on the mid-price [1]. If $g(v_t) < 0$, it implies that liquidation is pushing the mid-price upwards which is not likely to happen [2].
The Almgren-Chriss (AC) model assumes that the permanent price impact is linear in the speed of trading so that we have:

$$g(v_t) = bv_t \tag{4}$$

where $v_t > 0$ is the trading speed and $b$ is a positive constant. Modelling $(b_t)_{0\leq t\leq T}$ as a positive stochastic process with mean-reversion, we propose the dynamics of the Cox-Ingersoll-Ross (CIR) model where:

$$db_t = \eta_b(\mu_b - b_t)dt + \sigma_b\sqrt{b_t}dW^b_t \tag{5}$$

where $\eta_b$ is the rate of reversion to the mean, $\mu_b$ is the long-running mean of the process, $\sigma_b$ is the volatility of $b_t$ and $W^b_t$ is a standard brownian motion uncorrelated with $W_t$.

**Temporary Price Impact**

A large block of shares typically cannot be liquidated at the same price as there are a limited number of shares available at each price level in the limit order book. A large market order walks the book once the liquidity at the best bid/ask price is exhausted to use up the liquidity at each successive price level; this subsequently lowers the average execution price [4]. This impact is only temporary as the limit order book is continuously rebalanced so its liquidity is restored and it does not affect the mid-price of the asset [1]. The execution price for liquidation is given by:

$$E_t^v = S_t^v - (\frac{1}{2}\Delta + f(v_t))$$

where $S_t^v$ is the mid-price at time $t$, $\Delta \geq 0$ is the bid-ask spread and $f(v_t)$ is the temporary price impact function.
The function $f(v_t) : R \to R$ has a negative sign in the equation and is positive over the trading horizon to reflect that the liquidation lowers the average execution price. We can assume that $\Delta = 0$ as it does not affect the optimal trading strategy [1], giving us:

$$E_t^v = S_t^v - f(v_t) \tag{6}$$

In the AC model, the temporary price impact function is linear in the trading speed:

$$f(v_t) = kv_t \tag{7}$$

where $v_t > 0$ is the trading speed and $k$ is a positive constant. Similar to $b_t$, we propose a CIR model for $k_t$ as a positive stochastic process:

$$dk_t = \eta_k(\mu_k - k_t)dt + \sigma_k\sqrt{(}k_t)dW_t^k \tag{8}$$

where $\eta_k$ is the rate of reversion to the mean, $\mu_k$ is the long-running mean of the process, $\sigma_k$ is the volatility of $k_t$ and $W_t^k$ is a standard brownian motion uncorrelated with $W_t$.
According to [3], temporary and permanent price impact parameters are positively correlated and thus we assume $W_t^k$ and $W_t^b$ are correlated with $\rho$.

## 3 The Almgren-Chriss Optimal Execution strategy

**Derivation of the optimal speed**

The income earned from liquidating $N$ shares over the trading horizon is $X_T^v$. If any inventory is left over at time $T$, a market order can be executed to liquidate it at the current mid-price $S_T^v$. A terminal liquidation penalty is applied so that the income generated from the trade is $Q_T^v(S_T^v - \alpha Q_T^v)$ where $\alpha > 0$ is the penalty parameter. A running inventory penalty, $\phi \int_t^T (Q_u^v)^2 du$, is also applied where a high value of the urgency parameter $\phi > 0$ rewards strategies that liquidate

quickly and avoid the risk of price uncertainty that comes with trading over longer periods of time.
The optimal execution strategy aims to minimise the execution cost which is defined as the difference between the benchmark price, which is the mid-price of the asset, and the average price per share at which the shares are traded [1]. This is given by:

$$EC^v = NS_0 - \mathbf{E}[X_T^v + Q_T^v(S_T^v - \alpha Q_T^v) - \phi \int_t^T (Q_u^v)^2 du] \tag{9}$$

For optimal execution, the expected value of the income earned in (9) needs to be maximised. This gives rise to a value function:

$$H(t,x,S,q,k,b) = \sup_{v\in\mathcal{A}} H^v(t,x,S,q,k,b) = \sup_{v\in A} \mathbf{E}_{t,x,S,q}[X_T^v + Q_T^v(S_T^v - \alpha Q_T^v) - \phi \int_t^T (Q_u^v)^2 du] \tag{10}$$

where $X_t = x, S_t = S, Q_t = q, k_t = k, b_t = b$ and $\mathcal{A}$ is the set of admissible strategies.
We use equations (5) and (8) for the price impact functions. According to the dynamic programming principle, the value function satisfies the HJB equation:

$$\frac{\partial H}{\partial t} + \frac{1}{2}\sigma^2 \frac{\partial^2 H}{\partial S^2} - \phi q^2 + \mathcal{L}_H^{(k,b)} + \sup_v [v(S - kv)\frac{\partial H}{\partial x} - bv\frac{\partial H}{\partial S} - v\frac{\partial H}{\partial q}] = 0 \tag{11}$$

with $\mathcal{L}$ being the generator of the process $(k, b)$:

$$\mathcal{L}_H^{(k,b)} = \eta_k(\mu_k - k)\frac{\partial H}{\partial k} + \frac{1}{2}\sigma_k^2 k_t \frac{\partial^2 H}{\partial k^2} + \eta_b(\mu_b - b)\frac{\partial H}{\partial b} + \frac{1}{2}\sigma_b^2 b_t \frac{\partial^2 H}{\partial b^2} + \rho\sigma_k\sigma_b\sqrt{bk}\frac{\partial^2 H}{\partial b \partial k} \tag{12}$$

and the terminal condition $H(T,x,S,q,k,b) = x + Sq - \alpha q^2$.
This terminal condition points to the ansatz solution

$$H(t,x,S,q,k,b) = x + Sq + h(t,q,k,b), \quad h(T,q,k,b) = -\alpha q^2 \tag{13}$$

where $x$ is the cash earned from the strategy, $Sq$ is the book value of the remaining inventory at the mid-price and $h(t,q,k,b)$ is earned by optimally liquidating the rest of the shares and is independent of the mid-price.
By differentiating the supremum in (11) and substituting (13), we get an expression for the optimal speed:

$$v^* = -\frac{1}{2k}[\frac{\partial h}{\partial q} + bq] \tag{14}$$

where $h$ satisfies:

$$\frac{\partial h}{\partial t} + \mathcal{L}_h^{(k,b)} - \phi q^2 + \frac{1}{4k}[bq + \frac{\partial h}{\partial q}]^2 = 0 \tag{15}$$

Using the terminal condition for $h$ in (13), we could use separation of variables,

$$h(t,q,k,b) = q^2 h_2(t,k,b), \quad h_2(T,k,b) = -\alpha \tag{16}$$

Plugging (16) into (15) to get $h_2$,

$$q^2\frac{\partial h_2}{\partial t} + q^2\mathcal{L}_{h_2}^{(k,b)} - \phi q^2 + \frac{1}{4k}[bq + 2qh_2(t,k,b)]^2 = 0 \tag{17}$$

$$\frac{\partial h_2}{\partial t} + \mathcal{L}_{h_2}^{(k,b)} - \phi + \frac{1}{k}[\frac{1}{2}b + h_2(t,k,b)]^2 = 0 \tag{18}$$

We use Cartea and Jaimungal's [3] reasoning to propose an ansatz for $h_2$:

$$h_2(t,k,b) = h_a(t,k,b) + h_b(t,k,b) + h_c(t,k,b) \tag{19}$$

where:

$$\frac{\partial h_a}{\partial t} + \mathcal{L}_{h_a}^{(k,b)} + \frac{1}{4k}h_b^2 = 0 \tag{20}$$

$$\frac{\partial h_b}{\partial t} + \mathcal{L}_{h_b}^{(k,b)} + \frac{1}{2k}h_b(b + 2h_c) = 0 \tag{21}$$

$$\frac{\partial h_c}{\partial t} + \mathcal{L}_{h_c}^{(k,b)} - \phi + \frac{1}{4k}(b + 2h_c)^2 = 0 \tag{22}$$

with terminal conditions $h_a(t,k,b) = 0, \quad h_b(t,k,b) = 0, \quad h_c(t,k,b) = -\alpha$

Although the terminal conditions do not contain $b$ or $k$, equation (22) for $h_c$ contains a source term in $b$ which means we cannot definitively conclude that $h_c$ is a function of only time.

Another approach would be to follow the derivation of the optimal Almgren-Chriss speed used in [1], which assumes that $b$ and $k$ are constants, and then modify it to adjust to stochastic price impact. In this case, $(\eta_k, \eta_b, \sigma_k, \sigma_b) = (0,0,0,0)$ and thus $\mathcal{L}$ would be 0. Then we could redefine $h_2$ as a function of time:

$$\frac{\partial h_2}{\partial t} - \phi + \frac{1}{k}[\frac{1}{2}b + h_2(t)]^2 = 0 \tag{23}$$

This is a Riccati type ODE which reduces to:

$$h_2(t) = -\frac{1}{2}b + \sqrt{k\phi}\frac{1 + \zeta e^{2\gamma(T-t)}}{1 - \zeta e^{2\gamma(T-t)}} \tag{24}$$

where

$$\gamma = \sqrt{\frac{\phi}{k}} \tag{25}$$

$$\zeta = \frac{\alpha - \frac{1}{2}b + \sqrt{k\phi}}{\alpha - \frac{1}{2}b - \sqrt{k\phi}} \tag{26}$$

Using (14) to get the optimal speed,

$$v_t^* = \gamma\frac{\zeta e^{\gamma(T-t)} + e^{-\gamma(T-t)}}{\zeta e^{\gamma(T-t)} - e^{-\gamma(T-t)}}Q_t^{v^*} \tag{27}$$

Using (1) to get the speed in terms of the number of shares,

$$v_t^* = \gamma\frac{\zeta e^{\gamma(T-t)} + e^{-\gamma(T-t)}}{\zeta e^{\gamma T} - e^{-\gamma T}}N \tag{28}$$

Equation 28 is the speed in the Almgren-Chriss strategy for the optimal execution of the block of $N$ shares. We adapt this for stochastic processes $b_t$ and $k_t$,

$$v_t^* = \gamma_t \frac{\zeta_t e^{\gamma_t (T-t)} + e^{-\gamma_t (T-t)}}{\zeta_t e^{\gamma_t T} - e^{-\gamma_t T}} N \tag{29}$$

where

$$\gamma_t = \sqrt{\frac{\phi}{k_t}} \tag{30}$$

$$\zeta_t = \frac{\alpha - \frac{1}{2} b_t + \sqrt{k_t \phi}}{\alpha - \frac{1}{2} b_t - \sqrt{k_t \phi}} \tag{31}$$

## 4 Numerical Simulations

**Setting up the simulation**

In this section, we set up Monte Carlo simulations to compare the behaviour of the proposed model (referred to here as $B$) to the original Almgren-Chriss (referred to here as $A$) trading strategy when liquidating a large block of shares.

**Choosing parameters to simulate price impact** We assume that $b_t$ and $k_t$ are CIR processes with parameters $\eta, \mu, \sigma$ where $\mu$ is the long running mean, $\eta$ is the rate of mean-reversion and $\sigma$ is the volatility of the process and that they and initially start at the long-run mean value $\mu$. They satisfy the SDEs (5) and (8) defined in Section 2. In the strategy $A$, where $b$ and $k$ are constants, this translates to $(b, k) = (\mu_b, \mu_k)$. For the price impact processes to remain constant, their parameters must satisfy the Feller condition:

$$2\eta\mu > \sigma^2$$

Figure 1 shows the effect of $\mu_k$ and $\mu_b$ on the income and inventory process in strategy $A$. We can see that as $k_t$ increases, the number of shares sold and subsequently the income earned decreases. Equation (6) suggests that a high temporary price impact would lower the execution price sizeably causing a loss of income and vice versa. From (30), we see that $k_t$ is inversely proportional to $\gamma_t$ so as $k_t \to 0, \gamma_t \to \infty$ leading to the rapid liquidation of inventory caused by high trading speeds. Higher values of $b_t$ generate lower income but do not affect the inventory process nearly as much as $k_t$. A high permanent price impact would drive down the mid-price of the asset significantly which explains the drop in income as $b_t \to \infty$. We choose $(\mu_k, \mu_b) = (0.001, 0.001)$ as these values are also used in simulations in [1].

Figure 2 shows us that for low values of $\sigma$, the CIR process barely deviates from $\mu$. We choose relatively high values for $\sigma$ so the stochastic processes do not stay constant as this will not differentiate strategy $B$ from $A$. Once $\sigma$ and $\mu$ are chosen, we use the Feller condition and get $\eta > \frac{0.01^2}{2 \times 0.001} = 0.05$.

Table 1 lists all the parameters used to simulate $k_t$ and $b_t$.

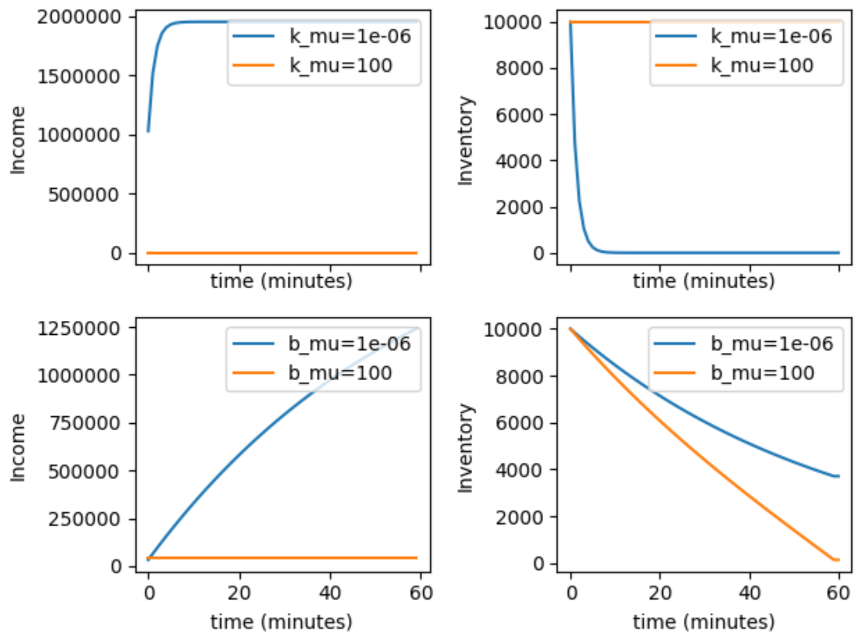

Figure 1: Effect of extreme values of $\mu_k$ and $\mu_b$ on the income and inventory paths in strategy $A$

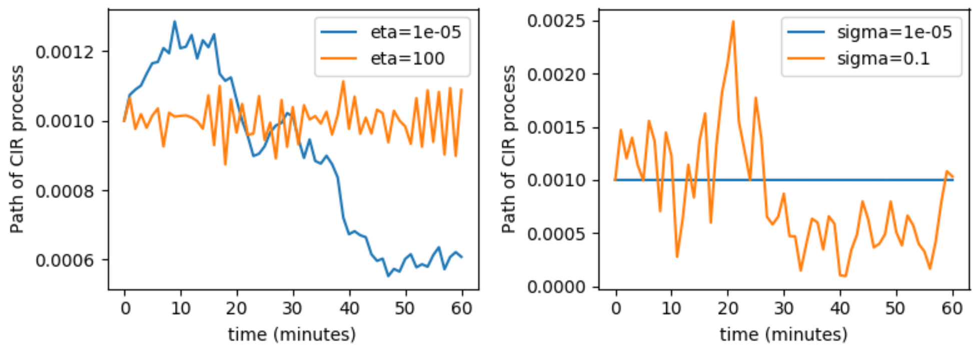

Figure 2: Path of a CIR process with ($\mu = 0.001, \sigma = 0.01$) and ($\mu = 0.001, \eta = 1$) respectively

| Parameter | Value |
|---|---|
| $k_0$ or $\mu_k$ | 0.001 |
| $b_0$ or $\mu_b$ | 0.001 |
| $\eta_k$ | 1 |
| $\eta_b$ | 1 |
| $\sigma_k$ | 0.01 |
| $\sigma_b$ | 0.01 |

Table 1: Parameters used to simulate the stochastic processes $k_t$ and $b_t$

**Choosing parameters to simulate liquidation** For the simulation, we assume that the investor has to liquidate 10000 shares of Apple with current market price $S_0 = 195.35$ and a daily volatility $\sigma = 0.015$. To calculate $\sigma$, we use AAPL stock price data from Nasdaq over 90 days and the formula:

$$\sigma = \sqrt{\frac{\sum_{i=0}^{90} r_i - \overline{r}}{90}} \tag{32}$$

$$r_i = \frac{p_i}{p_{i-1}} - 1 \tag{33}$$

where $p_i$ is the price of the stock on the $i^{th}$ day and $\overline{r}$ is the mean return.
We now need to choose the values of the penalty parameters, $\alpha$ and $\phi$. From figure 3, we see that the optimal trading speed increases as $\alpha$ increases leading to faster liquidation of assets while as $\alpha \to 0$, the trading speed drops significantly causing very little inventory to be sold. A high value of $\alpha$ corresponds to a severe terminal liquidation penalty which increases the incentive for the optimal strategy to trade faster and decrease the terminal inventory while a low $\alpha$ has a lower penalty and results in slower strategies.

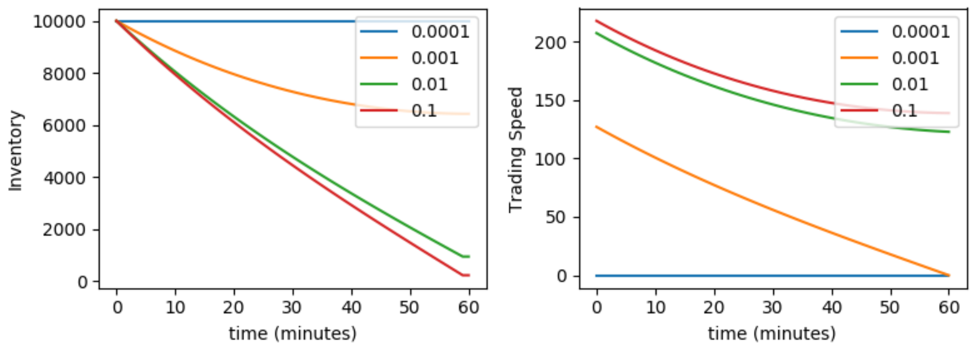

Figure 3: Effect of $\alpha$ on the inventory levels and trading speed in model $A$. Remaining model parameters are in Table 2.

From figure 4 we see that as $\phi$ increases, the inventory and speed processes become more convex as shares are sold faster while as $\phi \to 0$, the trading speed is almost constant and fewer shares are liquidated over $T$. A high $\phi$ rewards risk-averse strategies that liquidate sooner rather than risking the price uncertainty of longer horizons as it represents the trader's urgency. We choose $(\alpha, \phi) = (0.001, 0.001)$ in the Monte Carlo simulations.

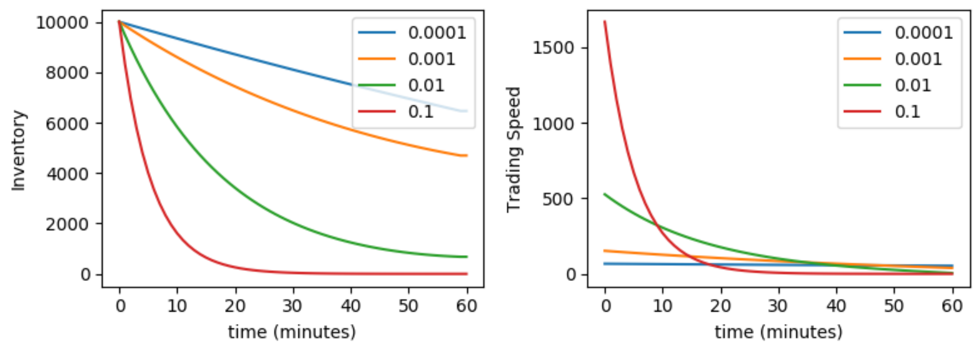

Figure 4: Effect of $\phi$ on the inventory levels and trading speed in model $A$. Remaining model parameters are in Table 2.

All the SDEs are discretised using the Euler-Maruyama method and the code to run both strategies is given in Appendix 1. The parameters used to simulate both strategies are given in Table 2.

| Parameter | Value | Notes |
|---|---|---|
| $Paths$ | 10000 | Number of Monte-Carlo paths |
| $N$ | 10000 | Number of shares to liquidate |
| $T$ | 1 hour | Trading horizon |
| $S_0$ | 195.35 | Price of the AAPL stock on 04/04/2019 |
| $\sigma$ | 0.015 | Daily volatility of the stock |
| $\alpha$ | 0.001 | Terminal liquidation penalty |
| $\phi$ | 0.001 | Running Inventory penalty |

Table 2: Parameters used to simulate liquidation of 10000 shares of the AAPL stock

### Simulation Performance

To measure the relative performance of the stochastic strategy relative to the Almgren-Chriss strategy, we use [3]:

$$\frac{X_T^{v^*} - X_T^{AC}}{X_T^{AC}} \times 10^4 \tag{34}$$

where $X_T^{v^*}$ is the cash generated from strategy $B$ and $X_T^{AC}$ is the income earned from strategy $A$, all other parameters remaining the same. In our Monte Carlo simulations, we obtain the mean relative performance over 10000 simulations to be 3.54 bps. Figure 5 is a histogram of the relative financial performance of the two strategies against each other.

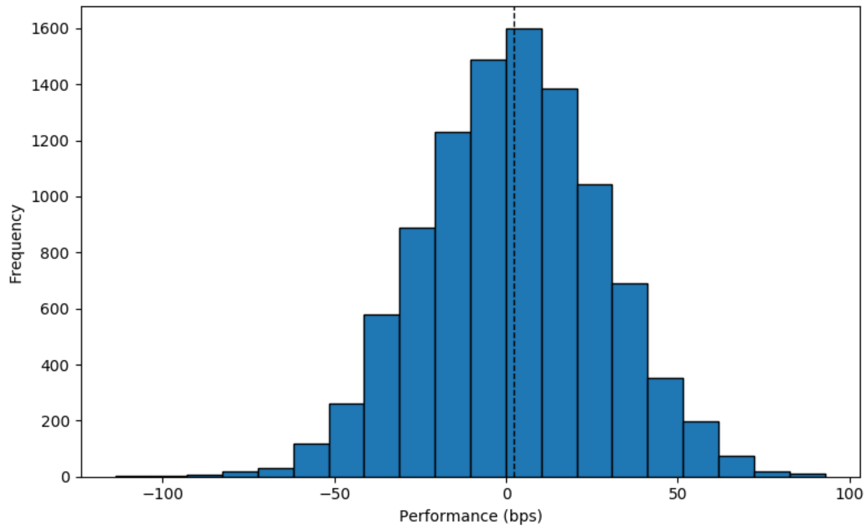

Figure 5: Relative performance as given in (34) of stochastic strategy to the Almgren-Chriss model. The vertical dashed line stands for the performance mean.

In the following plots, when we tweak the price impact process parameters, we always ensure the Feller condition is upheld to keep the processes positive. Figure 6 shows us a comparison of the paths of the optimal trading strategies $v_t^*$ and the income generated in each model. The trading speed is much more volatile in the stochastic model while it seems to decrease linearly in the original model. It is always positive as it is truncated to $max(v_t^*, 0)$ to prevent any assets being repurchased during the liquidation process [3].

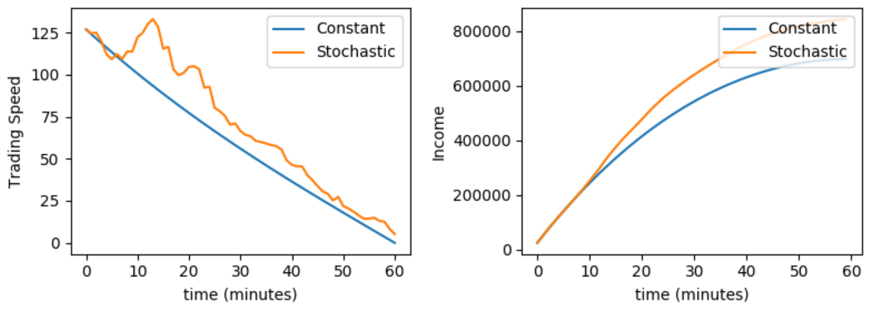

Figure 6: Optimal trading speed and income paths in the constant and stochastic models

Figure 7 shows the effect of the temporary price impact process on the optimal strategy $v_t^*$. As $\mu_k \to 0, \sigma_k \to 0$, $v_t^*$ follows an exponential decay curve with a rapid liquidation of all the inventory. Since temporary impact models stochastic liquidity, a low value could imply that more shares can be liquidated at the best execution price in the limit order book and thus a large order does not have to walk the book as much and the downward pressure on the average execution price is less. The left plot is similar to figure 6, just with lower speeds. Similarly, Figure 8 shows

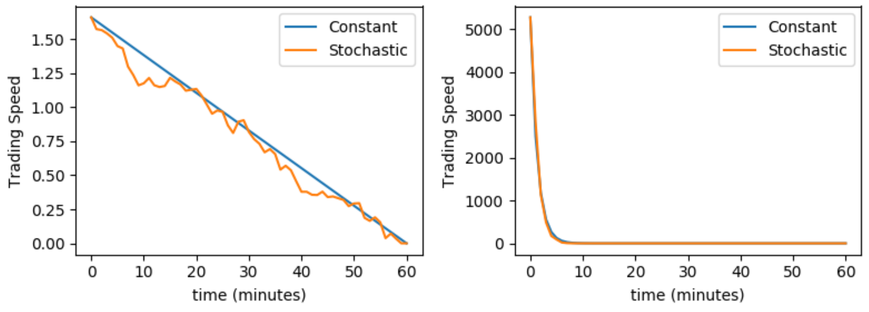

Figure 7: Trading speed with ($\mu_k = 0.1, \eta_k = 1, \sigma_k = 0.01$) (left) and ($\mu_k = 0.000001, \eta_k = 1, \sigma_k = 0.001$) (right)

the evolution of $v_t^*$ for different permanent impact processes. Unlike $k_t$, a smoother $b_t$ with $\mu_b \to 0, \sigma_b \to 0$ does not significantly change the shape of the trading speed curve. A greater permanent price impact leads to an erratic speed curve (left) where the truncation of negative speeds is evident. This could be because the term $\alpha - \frac{1}{2}b_t$ becomes negative for $b_t > \alpha$ which, over time, leads to $v_t^* < 0$.

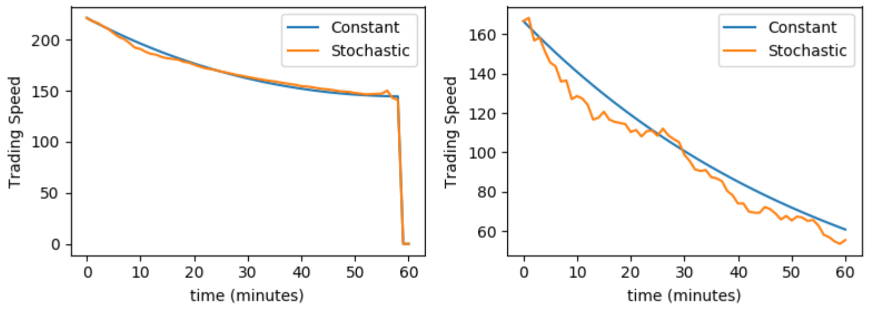

Figure 8: Trading speed with ($\mu_b = 0.1, \eta_b = 1, \sigma_b = 0.01$) (left) and ($\mu_b = 0.000001, \eta_b = 1, \sigma_b = 0.001$) (right)

Figure 9 shows the evolution of the mid-price of the stock $S_t$ and the execution price $E_t$ over the liquidation process. In both models, we see that the permanent price impact of the liquidation puts a downward pressure on $S_t$ and that $E_t$ is always less than $S_t$ due to the temporary price impact.

We now investigate how different parameters affect the relative performance of the modified

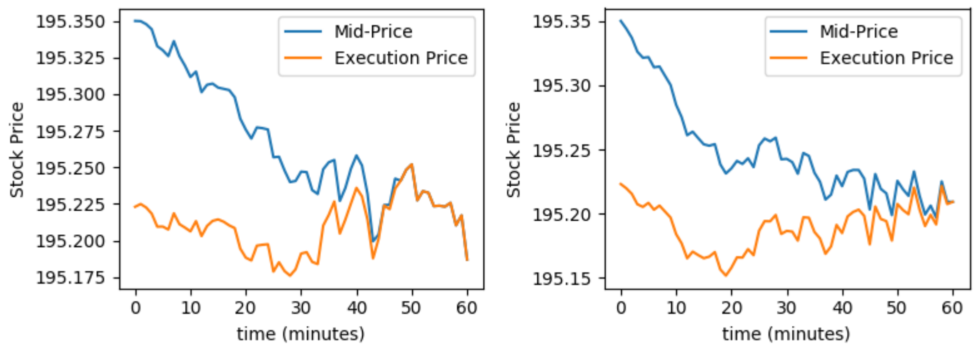

Figure 9: Mid-price and Execution price of the stock in the stochastic model (left) and constant model (right)

strategy. Since the values of these parameters are small, we use a log scale for the plots. From Figure 10 we see that as $\phi$ increases, the relative performance between the two models decreases. This could be because higher values of $\phi$ reward more risk-averse strategies and as the agent's urgency to liquidate increases, both strategies start to mirror each other thus reducing the performance difference. The relationship between $\alpha$ and the relative performance is less clear and seems to be at a maximum around $\alpha = 0.001$ which is the value we used for the simulation.

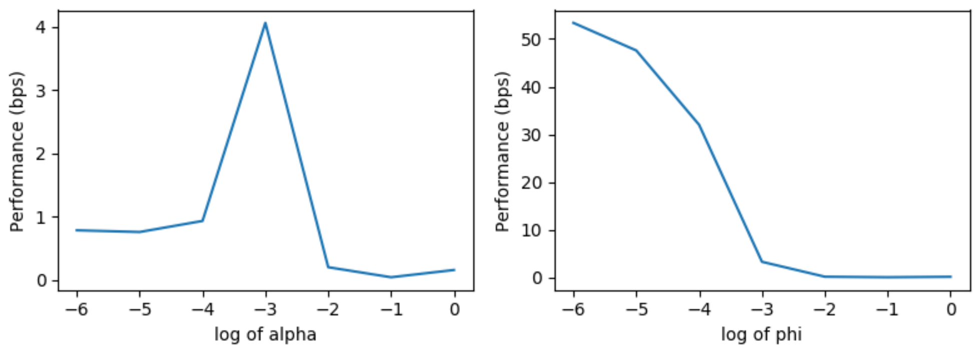

Figure 10: Effect of $\alpha$ (left) and $\phi$ (right) on the relative performance

Figure 11 supports our theory about $\phi$ in figure 10. As $\phi \to 0$, the paths of the trading strategies overlap causing a lower difference in performance. Similarly, for a relatively high $\phi$ (right), strategy $B$ is much more volatile than $A$ which explains the high performance difference between the two.

Figure 12 shows us that the relative performance increases as $\sigma$ increases. From figure 2, we saw that the for a small $\sigma$, the CIR process does not deviate much from the mean. Thus, for smaller $\sigma$ values, the paths of the two strategies largely overlap which decreases the performance

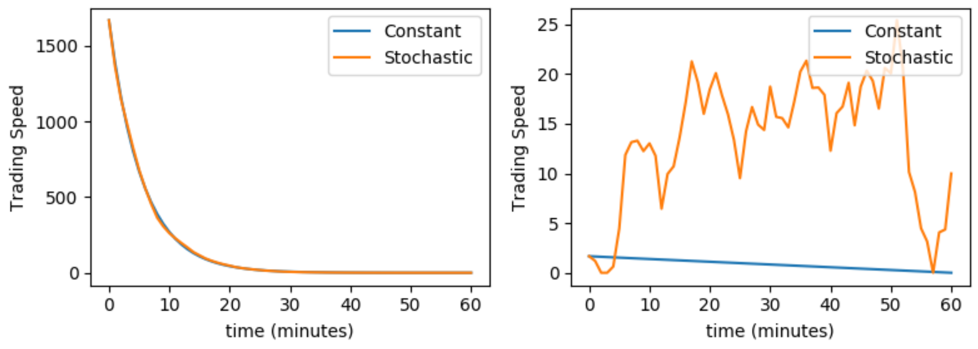

Figure 11: Effect of $\phi = 0.1$ (left) and $\phi = 0.00001$ (right) on the trading speed

difference between them. As $\sigma$ increases and the price impact processes become more volatile, strategy B tends to perform better. Similarly the relative performance generally decreases as $\eta$ increases as a high rate of mean-reversion causes the process to oscillate around the mean while a low $\eta$ allows the CIR process to deviate more increasing the relative performance of $B$

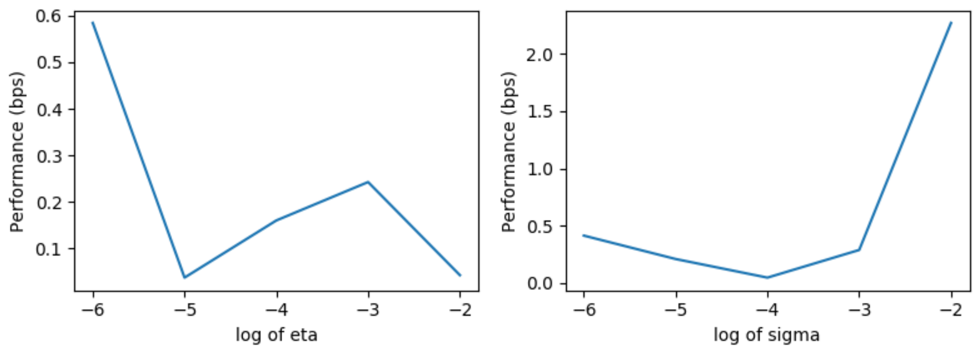

Figure 12: Effect of $\eta$ (left) and $\sigma$ (right) on the relative performance

## 5 Conclusion

In this paper, we examine the robustness of the Almgren-Chriss optimal execution model with stochastic price impact parameters which follow the dynamics of a Cox-Ingersoll-Ross process. We run 10000 Monte Carlo simulations to compare the performance of the proposed strategy to the original model and the modified model exceeds the performance of the original by 3.54 bps. On average, the strategy performed better when the price impact processes were more volatile and had a lower rate of mean reversion. The optimal value for the terminal liquidation penalty $\alpha$ was approximately 0.001 in the stochastic model. For very low and high values of $\alpha$, the relative performance diminished and the strategies mirrored each other. This could be because when the agent is either very slow or very fast to liquidate, the two strategies end up overlapping. Similarly, the proposed model performed better for lower values of $\phi$, the urgency parameter. This is because a less risk-averse agent wouldn't liquidate rapidly to avoid the price uncertainty of longer horizons, as is the case with larger $\phi$ values. However, very low values of $\phi$ are less realistic as agents are more likely to be risk-averse in real-life markets and tend to protect themselves against price risk. On the other hand, the $\alpha$ value depends on the agent's preference and can be altered accordingly.

## References

[1] Cartea, Á., S. Jaimungal, and J. Penalva (2015). Algorithmic and high-frequency trading. Cambridge University Press.

[2] Barger, W. and Lorig, M. (2018). Optimal liquidation under stochastic price impact. arXiv preprint arXiv:1804.04170.

[3] Cartea, Á., and S. Jaimungal. 2016b. Incorporating Order-Flow into Optimal Execution. Mathematics and Financial Economics 10 (3): 339–364.

[4] Almgren, R. and N. Chriss (2001). Optimal execution of portfolio transactions. Journal of Risk, 5–40.

# 6 Appendix

## 6.1 Monte Carlo Simulations

```
import numpy as np
from matplotlib import pyplot as plt
import pandas as pd

# define all parameters
T = 1 # expiry (hours)
N = 1e4 # number of shares
s_sigma = 0.015 # vol of the stock
s0 = 195.35 # initial stock price
phi = 1e-3 # running inventory penalty
alpha = 1e-3 # terminal liquidation penalty

k_eta = 1 # mean reversion rate of k
k_mu = 1e-3 # long running mean of k
k_sigma = 0.01 # vol of k

b_eta = 1 # mean reversion rate of b
b_mu = 1e-3 # long running mean of b
b_sigma = 0.01 # vol of b


def run_almgren_chriss_with_constant_pi(alpha=alpha, phi=phi, T=T, k_mu=k_mu,
    b_mu=b_mu, s_sigma=s_sigma):
    # time in minutes
    t = T * 60
    dt = 1 / 60.

    # stochastic processes
    w = np.random.randn(t + 1) # brownian motion for stock price, normal random
        variables
    s = np.zeros(t + 1) # mid-price process
    e = np.zeros(t + 1) # execution price process
    v = np.zeros(t + 1) # trading speed process
    x = np.zeros(t + 1) # income process
    q = np.zeros(t + 1) # inventory process

    gamma = np.sqrt(phi / k_mu)
    zeta = (alpha - 0.5 * b_mu + np.sqrt(phi * k_mu)) / (alpha - 0.5 * b_mu -
        np.sqrt(phi * k_mu))
    for i in range(t + 1):
        q[i] = N if i == 0 else q[i - 1] - min(v[i - 1], q[i - 1])
        s[i] = s0 if i == 0 else max(s[i - 1] + (- b_mu * v[i - 1] * dt + s_sigma * w[i]
            * np.sqrt(i * dt)), 0)

        tT = (t - i) / float(t)
        v_num = zeta * np.exp(gamma * tT) + np.exp(-gamma * tT)
        v_denom = zeta * np.exp(gamma * tT) - np.exp(-gamma * tT)
        v[i] = gamma * (v_num / v_denom) * q[i]
        v[i] = max(v[i], 0) * dt
        e[i] = max(s[i] - k_mu * v[i], 0)
        if i == t:
            x[i] = 0
            q[i] = q[i - 1]
        else:
            x[i] = e[i] * min(v[i], q[i])

    x[t] = q[t] * (s[t] - alpha * q[t])
    return x.cumsum(), q, v


def calculate_stochastic_path(mu, eta, sigma):
    t = T * 60
    dt = 1/60.
    w = np.random.randn(t + 1) # brownian motion for the price impact param
    pi_path = np.zeros(t + 1) # path for the price impact param
    pi_path[0] = mu

    for i in range(1, t + 1):
        pi_path[i] = pi_path[i - 1] + (eta * mu - eta * pi_path[i - 1]) * dt + sigma *
            w[i] * np.sqrt(pi_path[i-1] * dt)
    return pi_path


def run_almgren_chriss_with_stochastic_pi(alpha=alpha, phi=phi, T=T, k_mu=k_mu,
    k_eta=k_eta, k_sigma=k_sigma, b_mu=b_mu, b_eta=b_eta, b_sigma=b_sigma,
                                  s_sigma=s_sigma):
    # time in minutes
    t = T * 60
    dt = 1 / 60.

    # stochastic processes
    w = np.random.randn(t + 1) # brownian motion for stock price, normal random
        variables
    s = np.zeros(t + 1) # mid-price process
    e = np.zeros(t + 1) # execution price process
    v = np.zeros(t + 1) # trading speed process
    x = np.zeros(t + 1) # income process
    q = np.zeros(t + 1) # inventory process

    # coefficients for optimal speed
    gamma = np.zeros(t + 1)
    zeta = np.zeros(t + 1)

    k = calculate_stochastic_path(k_mu, k_eta, k_sigma)
    b = calculate_stochastic_path(b_mu, b_eta, b_sigma)

    for i in range(t + 1):
        gamma[i] = np.sqrt(phi / k[i])
        zeta[i] = (alpha - 0.5 * b[i] + np.sqrt(phi * k[i])) / (alpha - 0.5 * b[i] -
            np.sqrt(phi * k[i]))

        q[i] = N if i == 0 else q[i - 1] - min(v[i - 1], q[i - 1])
        s[i] = s0 if i == 0 else max(s[i - 1] - b[i] * v[i - 1] * dt + s_sigma * w[i] *
            np.sqrt(i * dt), 0)

        tT = (t - i) / float(t)
        v_num = zeta[i] * np.exp(gamma[i] * tT) + np.exp(-gamma[i] * tT)
        v_denom = zeta[i] * np.exp(gamma[i] * tT) - np.exp(-gamma[i] * tT)
        v[i] = gamma[i] * (v_num / v_denom) * q[i]
        v[i] = max(v[i], 0) * dt
        e[i] = max(s[i] - k[i] * v[i], 0)
        if i == t:
            x[i] = 0
            q[i] = q[i - 1]
        else:
            x[i] = e[i] * min(v[i], q[i])

    x[t] = q[t] * (s[t] - alpha * q[t])
    return x.cumsum(), q, v

def calculate_performance(mc_paths, alpha=alpha, phi=phi, T=T, k_mu=k_mu, k_eta=k_eta,
    k_sigma=k_sigma, b_mu=b_mu, b_eta=b_eta, b_sigma=b_sigma,
                        s_sigma=s_sigma):
    cash_from_constant_strategy = np.zeros(mc_paths)
    cash_from_stochastic_strategy = np.zeros(mc_paths)
    performance = np.zeros(mc_paths)

    for x in range(mc_paths):
        cash_from_constant_strategy[x] =
            run_almgren_chriss_with_constant_pi(alpha=alpha, phi=phi, T=T, k_mu=k_mu,
            b_mu=b_mu, s_sigma=s_sigma)[0][-1]
        cash_from_stochastic_strategy[x] =
            run_almgren_chriss_with_stochastic_pi(alpha=alpha, phi=phi, T=T, k_mu=k_mu,
            k_eta=k_eta, k_sigma=k_sigma, b_mu=b_mu, b_eta=b_eta, b_sigma=b_sigma,
            s_sigma=s_sigma)[0][-1]
        performance[x] = (cash_from_stochastic_strategy[x] -
            cash_from_constant_strategy[x]) / cash_from_constant_strategy[x]
        performance[x] *= 10000

    return performance
```

## 6.2 Plotting

```
from optimal_execution import *

def plot_performance_histogram():
    f, ax = plt.subplots(figsize=(8, 5))
    ax.set(ylabel='Frequency', xlabel='Performance (bps)')
    a = calculate_performance(10000)
    plt.hist(a, density=False, bins=20, edgecolor='k')
    plt.axvline(a.mean(), color='k', linestyle='dashed', linewidth=1)
    f.tight_layout()
    plt.show()


def plot_midprice():
    f, ax = plt.subplots(figsize=(5,3))
    ax.set(ylabel='Stock Price', xlabel='time (minutes)')
    plt.plot(ec, label='Constant')
    plt.plot(es, label='Stochastic')
    plt.legend()
    plt.tight_layout()
    plt.show()


def plot_midprice_vs_execution():
    f, (ax1, ax2) = plt.subplots(1, 2, sharex='col', figsize=(8,3))
    ax1.set(ylabel='Stock Price', xlabel='time (minutes)')
    ax1.plot(ss, label='Mid-Price')
    ax1.plot(es, label='Execution Price')
    ax1.legend()

    ax2.set(ylabel='Stock Price', xlabel='time (minutes)')
    ax2.plot(sc, label='Mid-Price')
    ax2.plot(ec, label='Execution Price')
    ax2.legend()

    plt.tight_layout()
    plt.show()


def plot_x_and_q_vs_alpha():
    a0001, b0001, c0001= run_almgren_chriss_with_constant_pi(alpha=0.0001)
    a001, b001, c001 = run_almgren_chriss_with_constant_pi(alpha=0.001)
    a01, b01, c01 = run_almgren_chriss_with_constant_pi(alpha=0.01)
    a1, b1, c1 = run_almgren_chriss_with_constant_pi(alpha=0.1)

    f, (ax1,ax3) = plt.subplots(1, 2, sharex='col', figsize=(8,3))
    ax1.set(ylabel='Inventory', xlabel='time (minutes)')
    ax1.plot(b0001, label='0.0001')
    ax1.plot(b001, label='0.001')
    ax1.plot(b01, label='0.01')
    ax1.plot(b1, label='0.1')
    ax1.legend(loc="upper right")

    ax3.set(ylabel='Trading Speed', xlabel='time (minutes)')
    ax3.plot(c0001, label='0.0001')
    ax3.plot(c001, label='0.001')
    ax3.plot(c01, label='0.01')
    ax3.plot(c1, label='0.1')
    ax3.legend(loc="upper right")
    plt.tight_layout()
    plt.show()


def plot_x_and_q_vs_phi():
    a0001, b0001, c0001 = run_almgren_chriss_with_constant_pi(phi=0.0001)
    a001, b001, c001 = run_almgren_chriss_with_constant_pi(phi=0.001)
    a01, b01, c01 = run_almgren_chriss_with_constant_pi(phi=0.01)
    a1, b1, c1 = run_almgren_chriss_with_constant_pi(phi=0.1)

    f, (ax1, ax3) = plt.subplots(1, 2, sharex='col', figsize=(8,3))
    ax1.set(ylabel='Inventory', xlabel='time (minutes)')
    ax1.plot(b0001, label='0.0001')
    ax1.plot(b001, label='0.001')
    ax1.plot(b01, label='0.01')
    ax1.plot(b1, label='0.1')
    ax1.legend(loc="upper right")

    ax3.set(ylabel='Trading Speed', xlabel='time (minutes)')
    ax3.plot(c0001, label='0.0001')
    ax3.plot(c001, label='0.001')
    ax3.plot(c01, label='0.01')
    ax3.plot(c1, label='0.1')
    ax3.legend(loc="upper right")
    plt.tight_layout()


def plot_CIR_path():
    low_eta = 0.00001
    high_eta = 100
    etahigh = calculate_stochastic_path(k_mu, high_eta, k_sigma)
    etalow = calculate_stochastic_path(k_mu, low_eta, k_sigma)

    low_vol = 0.00001
    high_vol = 0.1
    volhigh = calculate_stochastic_path(k_mu, 1, high_vol)
    vollow = calculate_stochastic_path(k_mu, 1, low_vol)

    f, (ax2, ax3) = plt.subplots(1, 2, sharex='col', figsize=(8,3))
    ax2.set(ylabel='Path of CIR process', xlabel='time (minutes)')
    ax2.plot(etalow, label='eta={}'.format(low_eta))
    ax2.plot(etahigh, label='eta={}'.format(high_eta))
    ax2.legend(loc="upper right")

    ax3.set(ylabel='Path of CIR process', xlabel='time (minutes)')
    ax3.plot(vollow, label='sigma={}'.format(low_vol))
    ax3.plot(volhigh, label='sigma={}'.format(high_vol))
    ax3.legend(loc="upper right")
    plt.tight_layout()
    plt.show()
```

```
def plot_x_and_q_in_both_models():
    cx, cq, cv = run_almgren_chriss_with_constant_pi()
    sx, sq, sv = run_almgren_chriss_with_stochastic_pi()
    f, (ax2,ax3) = plt.subplots(1,2, sharex='col', figsize=(8, 3))
    ax2.set(ylabel='Trading Speed', xlabel='time (minutes)')
    ax2.plot(cv, label='Constant')
    ax2.plot(sv, label='Stochastic')
    ax2.legend(loc="upper right")

    ax3.set(ylabel='Income', xlabel='time (minutes)')
    ax3.plot(cx[:-1], label='Constant')
    ax3.plot(sx[:-1], label='Stochastic')
    ax3.legend(loc="upper right")
    plt.tight_layout()
    plt.show()


def trading_speeds_against_variable():
    cx, cq, cv = run_almgren_chriss_with_constant_pi(phi=1e-1)
    cx1, cq1, cv1 = run_almgren_chriss_with_constant_pi(phi=1e-5)
    sx, sq, sv = run_almgren_chriss_with_stochastic_pi(phi=1e-1)
    sx1, sq1, sv1 = run_almgren_chriss_with_stochastic_pi(phi=1e-5)
    f, (ax2,ax3) = plt.subplots(1,2, sharex='col', figsize=(8, 3))
    ax2.set(ylabel='Trading Speed', xlabel='time (minutes)')
    ax2.plot(cv, label='Constant')
    ax2.plot(sv, label='Stochastic')
    ax2.legend(loc="upper right")

    ax3.set(ylabel='Trading Speed', xlabel='time (minutes)')
    ax3.plot(cv1, label='Constant')
    ax3.plot(sv1, label='Stochastic')
    ax3.legend(loc="upper right")
    plt.tight_layout()
    plt.show()


def plot_performance_vs_alpha_phi():
    powers = [10**x for x in range(-6,1)]
    alpha_values = [calculate_performance(100, alpha=x).mean() for x in powers]
    phi_values = [calculate_performance(100, phi=x).mean() for x in powers]

    f, (ax2, ax3) = plt.subplots(1, 2, sharex='col', figsize=(8,3))
    ax2.set(ylabel='Performance (bps)', xlabel='log of alpha')
    ax2.plot(np.log10(powers), np.abs(alpha_values))
    ax3.set(ylabel='Performance (bps)', xlabel='log of phi')
    ax3.plot(np.log10(powers), np.abs(phi_values))
    plt.tight_layout()
    plt.show()


def plot_performance_vs_cir_params():
    # sigma generated with keta as 1
    powers = [10**x for x in range(-6,-1)]
    eta_vals = [calculate_performance(100, b_mu=x, b_sigma=0.0001).mean() for x in
        powers]
    # bmu_vals = [calculate_performance(1000, b_sigma=x, b_eta=1).mean() for x in
        powers]
    sigma_vals = [calculate_performance(100, k_mu=x, k_sigma=0.0001).mean() for x in
        powers]
    f, (ax2, ax3) = plt.subplots(1, 2, sharex='col', figsize=(8,3))
    ax2.set(ylabel='Performance (bps)', xlabel='log of b_mu')
    ax2.plot(np.log10(powers), eta_vals)
    ax3.set(ylabel='Performance (bps)', xlabel='log of k_mu')
    ax3.plot(np.log10(powers), sigma_vals)
    plt.tight_layout()
    plt.show()


def plot_income_against_CIR():
    low_kmu = 0.000001
    kmu_med=0.001
    high_kmu = 100
    kmu_low = run_almgren_chriss_with_constant_pi(k_mu=low_kmu)
    kmu = run_almgren_chriss_with_constant_pi(k_mu=kmu_med)
    kmu_high = run_almgren_chriss_with_constant_pi(k_mu=high_kmu)

    low_bmu = 0.000001
    bmu_med=0.001
    high_bmu = 100
    bmu_low = run_almgren_chriss_with_constant_pi(b_mu=low_bmu)
    bmu = run_almgren_chriss_with_constant_pi(b_mu=bmu_med)
    bmu_high = run_almgren_chriss_with_constant_pi(b_mu=high_bmu)
    plot_income_and_inventory_against_cir_params(low_bmu, low_kmu, kmu_med, high_bmu,
        high_kmu, bmu_med, kmu_low, kmu_high, kmu, bmu_low, bmu_high, bmu, 'k_mu',
        'b_mu')
```

```
def plot_income_and_inventory_against_cir_params(low_b, low_k, k_med, high_b, high_k,
    b_med, k_low, k_high, kmu, b_low, b_high, bmu, param_k, param_b):
    f, ((ax1, ax2), (ax3, ax4)) = plt.subplots(2, 2, sharex='col')
    ax1.set(ylabel='Income', xlabel='time (minutes)')
    ax1.plot(k_low[0][:-1], label='{}={}'.format(param_k, low_k))
    # ax1.plot(kmu[0][:-1], label='{}={}'.format(param_k, k_med))
    ax1.plot(k_high[0][:-1], label='{}={}'.format(param_k, high_k))
    ax1.legend(loc="upper right")

    ax2.set(ylabel='Inventory', xlabel='time (minutes)')
    ax2.plot(k_low[1], label='{}={}'.format(param_k, low_k))
    # ax2.plot(kmu[1], label='{}={}'.format(param_k, k_med))
    ax2.plot(k_high[1], label='{}={}'.format(param_k, high_k))
    ax2.legend(loc="upper right")

    ax3.set(ylabel='Income', xlabel='time (minutes)')
    ax3.plot(b_low[0][:-1], label='{}={}'.format(param_b, low_b))
    # ax3.plot(bmu[0][:-1], label='{}={}'.format(param_b, b_med))
    ax3.plot(b_high[0][:-1], label='{}={}'.format(param_b, high_b))
    ax3.legend(loc="upper right")

    ax4.set(ylabel='Inventory', xlabel='time (minutes)')
    ax4.plot(b_low[1], label='{}={}'.format(param_b, low_b))
    # ax4.plot(bmu[1], label='{}={}'.format(param_b, b_med))
    ax4.plot(b_high[1], label='{}={}'.format(param_b, high_b))
    ax4.legend(loc="upper right")
    plt.tight_layout()
    plt.show()
```