A Tale of Two Models: Implementing the Almgren-Chriss framework through nonlinear and dynamic programming

Sebastien David, Arthur Bagourd, Mounah Bizri

April 26, 2018

Abstract: In this study, done in the Almgren-Chriss framework, we focused on testing the impact of using different trading benchmarks (implementation shortfall, VWAP and TWAP) on the trading trajectory and the efficient frontier. We found out that the impact of risk aversion is the highest on the implementation shortfall benchmark and smaller on TWAP and VWAP. We also found out that using different volatility measures does not impact the trading trajectories. Finally, this study provides a foray into an alternative approach - dynamic programming and the Bellman principle - to solve the Almgren-Chriss optimal execution trade-off.

1 Introduction

What is the optimal pace to build or unwind a position? What is the optimal trade-off between a fast (but costly) execution and a slow execution (less costly but susceptible to timing risk)? Issues related to the optimal execution of trading positions have long been faced by traders, marker makers and brokers, but have long been relatively ignored by statisticians, economists and econophysicists.

In this regard, Bertsimas and Lo (1998) [1] and Almgren and Chriss (2001) [2] represent the first forays into research on optimal execution. More specifically, Almgren-Chriss take into account the risk that the execution price would move over the course of the liquidation. For example, an algorithm executing with high liquidation speed is more costly because of limited liquidity, but a slower execution speed exposes the trader to (unexpected) price fluctuations. Their main result is that under a set of assumptions on price trajectory and market impact, there exists an optimal trade-off between execution costs and timing risk.

Almgren and Chriss (2001) thus became the canonical model for optimal execution research, and have seen a number of extensions with more mathematically intensive implementations (such as but not limited to stochastic control, dynamic programming and reinforcement learning), more realistic model assumptions (dynamic parameters, market impact estimation) and multi-asset investment universes.

In this paper, we will focus on the Almgren-Chriss framework (in discrete-time) and volatility estimation, gravitating around the three following topics:

- Solving Almgren-Chriss framework with nonlinear programming (original method used by the authors)
- Solving Almgren-Chriss framework with dynamic programming
- Volatility estimation, microstructure biases and their integration into the Almgren-Chriss framework

2 The Almgren-Chriss framework: assumptions and solving approaches

We start by assuming that we are studying a simple trading strategy: a trader wants to sell/buy a quantity of X shares, i.e. the trader looks to unwind his positions over a time window $[0, T]$. This is modelled by a trading trajectory $x_0, ..., x_N$ where x_k is the number of its that we plan to hold at time t_k. Our initial holding is $x_0 = X$ and liquidation at T requires $x_N = 0$. We specifically specify a strategy by a trading list $n_1, ..., n_N$ where $n_k = x_{k-1} - x_k$ is the number of units that we will sell between times t_{k-1} and t_k:

$$x_k = X - \sum_{j=1}^{k} n_j$$

The trader is faced with execution costs. In this regard, the framework considers market impact to be the most important component of transaction costs. Yet they are difficult to measure, as it might depend on the trading volume and the chosen strategy. There are 2 types of market impact: temporary impact modeled by a function $v \to h(v)$ (i.e. disequilibrium between supply and demand when an order is executed) and permanent impact $v \to g(v)$ (i.e. the trader is leaking information to other traders):

$$g(v) = \gamma v^{\alpha}$$

$$h(v) = \varepsilon \operatorname{sign}(v) + \eta\Big(\frac{v}{\tau}\Big)^{\beta}$$

Since we cannot estimate both permanent and temporary impact functions, we need to make assumptions on the structure of both functions. Almgren and al. (2005) [3] through empirical evidence indicate that the temporary impact function should be concave, where $0 < \beta < 1$ (the authors choose $\beta = 0.5$). For permanent impact, they take $\alpha = 1$ in order to make the model free of arbitrage and have permanent impact independent of trading (cf. Gueant, 2016 [4]). For our part, we choose linear market impact functions: $\alpha = 1$ and $\beta = 1$.

For the stock price $(S_t)_{t>0}$, the authors choose arithmetic Brownian motion in order to ensure that the optimal strategy will be dynamically optimal. If we chose the geometric Brownian motion, the optimal strategy would have been path-dependent as it would be dependent on past historical price movements.

$$S_k = S_{k-1} + \sigma\sqrt{\tau}\zeta_k - \tau g(\frac{n_k}{\tau})$$

$$S_k = S_0 + \sigma \sum_{j=1}^{k} \sqrt{\tau}\zeta_j - \gamma(X - x_k)$$

$$\tilde{S}_k = S_{k-1} - h(\frac{n_k}{\tau})$$

With those assumptions, we can now model the goal of the trader. In the case of our paper, we will only focus on the following three algorithms:

- **Implementation shortfall**: it is a pure cost-driven algorithm that looks to minimize the difference between the average execution price that is actually achieved and a benchmark. The aim is to achieve an average execution price that minimizes the shortfall when compared with the decision price. The key is to strike the right balance between market impact and timing risk. Often, this means the algorithms tend to take only as long as it is necessary to prevent significant market impact.

- **Volume Weighted Average Price**: average price that takes into account the traded volume during a time period. It is calculated as the traded value divided by the trade volume. It is classified as an impact-driven algorithm (i.e. slicing larger orders into smaller child orders in order to lower market impact costs).
- **Time Weighted Average Price**: average price that takes into account the time evolution of prices during the day. It is a natural extension of order slicing strategies. It has a uniform time-based schedule. A TWAP trade buys or sells uniformly part of the volume in the first and second half of the day. It is also classified as an impact-driven algorithm.

For a given benchmark B we define the market impact as:

$$\text{Payoff} = \text{Benchmark Payoff} - \text{Realized Payoff}$$

$$X_B = XB - \sum_{k=1}^{n} n_k \tilde{S_k}$$

where X is the size of the inventory we want to trade. A rational trader would look to minimize the expected cost $\mathbb{E}(n)$ for a given level of timing risk $\mathbb{V}(n)$ and would rationally favor a strategy $n^* = (n_1^*, ..., n_N^*)$ with the lowest variance for a given level of expected transaction costs.

We are going to use the Constant Absolute Risk Aversion (CARA) utility function, with a γ risk aversion coefficient. While there is no *a priori* reason for choosing the CARA utility function, it is mathematically convenient as it is equivalent to a mean-variance framework:

$$\min_n = \mathbb{E}(e^{\gamma X_B}) = \mathbb{E}\Big[e^{\gamma(XB - \sum_{k=1}^{n} n_k \tilde{S_k})}\Big]$$

The advantage of using the expectation of an exponential is that every time we want to solve a mean-variance problem, we can transform the problem into minimizing an expectation. Thus, it allows us to directly apply a wide variety of approaches, such as nonlinear programming, stochastic control, dynamic programming and reinforcement learning. Thus:

$$\min_n \; e^{\gamma \mathbb{E}\left(XB - \sum_{k=1}^{n} n_k \tilde{S_k}\right) + \frac{\gamma}{2} \mathbb{V}\left(XB - \sum_{k=1}^{n} n_k \tilde{S_k}\right)}$$

and equivalently:

$$\min_n \; \mathbb{E}\big(XB - \sum_{k=1}^{n} n_k \tilde{S_k}\big) + \lambda \mathbb{V}\big(XB - \sum_{k=1}^{n} n_k \tilde{S_k}\big)$$

where λ is also used to model risk aversion (with $\lambda = \frac{\gamma}{2}$). We can thus create optimal strategies by solving a constrained optimization problem with a strictly convex objective function which will yield a unique solution.

In the context of our paper, we will only focus on the following 2 models:

- **Nonlinear programming**: an optimization method that looks to maximize (or minimize) nonlinear objective functions with equality and inequality constraints. This was the original method used by Almgren-Chriss (2001).
- **Dynamic programming**: an optimization method that breaks down a complex problem into smaller and simpler subproblems which are then put back together. This method is often used as a starting point for more advanced implementations using either stochastic control (ex. Hamilton-Jacobi-Bellman equation) or reinforcement learning (ex. Q-Learning).

3 Solving Almgren-Chriss through nonlinear programming

3.1 Nonlinear optimization

For determining the optimal liquidation we will use **sequential quadratic programming** (SLSQP), an iterative nonlinear optimization method available in the optimization libraries of the 'Scipy' Python package. This problem is modeled as followed:

$$\min_x \ f(x)$$

$$s.t. \quad g_i(x) < 0 \qquad i = 1, ..., n_{ineq}$$

$$s.t. \quad h_i(x) = 0 \qquad i = 1, ..., n_{eq}$$

$f(x)$ is the function to minimize, $g_i(x)$ represents the inequality constraints and $h_i(x)$ represents the equality constraints. By introducing a cost function (or Lagrangian), the nonlinear optimization will satisfy the Karush-Kuhn-Tucker (KKT) conditions thus guaranteeing that the solution of the QP will be optimal:

$$L(x, \lambda, \nu) = f(x) + \lambda^T g(x) + \nu^T h(x)$$

In regards to the specifics of SLSQP, it is similar to Newton's method in that it solves the objective function iteratively through a (quadratic) approximation of the Lagrangian:

$$\min_d \ \nabla f(x_k)^T d + \frac{1}{2} d^T H_L(x_k, \lambda_k, \nu_k) d$$

$$s.t. \quad g_i(x_k) + \nabla g_i(x_k)^T d < 0$$

$$s.t. \quad h_j(x_k) + \nabla h_j(x_k)^T d = 0$$

Convergence to the optimal solution is dependent on the Hessian of the Lagrangian $H_L(x_k, \lambda_k, \nu_k)$ and smoothness/convexity of the objective functions. Thus, we thus apply the SLSQP optimization algorithm on:

$$\min_n \mathbb{E}\big(XB - \sum_{k=1}^{n} n_k \tilde{S_k}\big) + \lambda \mathbb{V}\big(XB - \sum_{k=1}^{n} n_k \tilde{S_k}\big)$$

3.2 Different benchmarks

For a given benchmark B we define the market impact as: $XB - \sum_{k=1}^{n} n_k \tilde{S_k}$ where X is the size of the inventory we want to trade.

Recall that the price trajectory is given by

$$S_k = S_{k-1} + \sigma_k \sqrt{\tau} \zeta_k - \tau g(\frac{n_k}{\tau})$$

$$g(v) = \gamma v$$

$$S_k = S_0 + \sum_{j=1}^{k} \sigma_k \sqrt{\tau} \zeta_j - \gamma(X - x_k)$$

$$h(v) = \varepsilon sign(n_i) + \frac{\eta}{\tau} n_i$$

$$\tilde{S}_k = S_{k-1} - h(\frac{n_k}{\tau})$$

NB: The volatility σ can be defined as constant or dynamic.

3.2.1 Implementation Shortfall

The implementation Shortfall (IS) is defined as the difference between the decision price and the final execution price: $IS = XS_0$.
The payoff is given by: $P = XS_0 - \sum_{k=1}^{n} n_k \tilde{S}_k$.

We have:

$$\mathbb{E}[P] = \sum_{k=1}^{n} \tau x_k g(\frac{n_k}{\tau}) + \sum_{k=1}^{n} n_k h(\frac{n_k}{\tau}) \tag{1}$$

and

$$\mathbb{V}[P] = \sum_{k=1}^{n} \sigma_k^2 \tau x_k \tag{2}$$

3.2.2 TWAP

TWAP is defined as the time weighted average price : $TWAP = \frac{\sum_{k=1}^{n} \tau S_k}{\sum_{k=1}^{n} \tau}$

The payoff is given by $P = X \frac{\sum_{k=1}^{n} \tau S_k}{\sum_{k=1}^{n} \tau} - \sum_{k=1}^{n} n_k \tilde{S}_k$.
Let us notice that $\sum_{k=1}^{n} \tau = T$ and that $\sum_{k=1}^{n} n_k = X$.

We have:

$$\begin{aligned}
\mathbb{E}[P] &= \mathbb{E}\Big[X \frac{\sum_{k=1}^{n} \tau S_k}{\sum_{k=1}^{n} \tau} - \sum_{k=1}^{n} n_k \tilde{S}_k\Big] \\
&= \mathbb{E}\Big[\frac{X\tau}{T} \sum_{k=1}^{n} S_k - \sum_{k=1}^{n} n_k \tilde{S}_k\Big] \\
&= \mathbb{E}\Big[\sum_{k=1}^{n} \frac{X\tau}{T}(S_0 + \sigma_k \sqrt{\tau} \sum_{j=1}^{k} \zeta_j - \sum_{j=1}^{k} \tau g(\frac{n_j}{\tau})) - n_k(S_0 + \sigma_k \sqrt{\tau} \sum_{j=1}^{k-1} \zeta_j - \sum_{j=1}^{k-1} \tau g(\frac{n_j}{\tau}) - h(\frac{n_k}{\tau}))\Big] \\
&= \sum_{k=1}^{n} \Big(\frac{X\tau}{T} - n_k\Big)\Big(S_0 - \tau \sum_{j=1}^{k-1} g(\frac{n_j}{\tau})\Big) - \frac{X}{T}\tau^2 g(\frac{n_k}{\tau}) + n_k h(\frac{n_k}{\tau})
\end{aligned} \tag{3}$$

$$\begin{aligned}\mathbb{V}[P] &= \mathbb{V}\Big[\sum_{k=1}^{n}\frac{X\tau}{T}(S_0+\sigma\sqrt{\tau}\sum_{j=1}^{k}\zeta_j-\sum_{j=1}^{k}\tau g(\frac{n_j}{\tau}))-n_k(S_0+\sigma\sqrt{\tau}\sum_{j=1}^{k-1}\zeta_j-\sum_{j=1}^{k-1}\tau g(\frac{n_j}{\tau})-h(\frac{n_k}{\tau}))\Big]\\ &= \sum_{k=}^{n}(\frac{X\tau}{T}\sigma^2\tau^2)\end{aligned} \tag{4}$$

To compute the Variance of the market impact we need the covariance terms so we can use the following simplification.

$$\begin{aligned}TWAP &= \frac{\sum_{k=1}^{n}\tau S_k}{\sum_{k=1}^{n}\tau}\\ &= \frac{1}{T}\sum_{k=1}^{n}\tau\big(S_0+\sigma_k\sqrt{\tau}\sum_{j=1}^{k}\zeta_j-\sum_{j=1}^{k}\tau g(\frac{n_j}{\tau})\big)\\ &= \frac{1}{T}\sum_{k=1}^{n}(\sigma_k\sqrt{\tau}\zeta_k\sum_{j=1}^{k}\tau)+a(n_k)\end{aligned} \tag{5}$$

And from [REF] we have:

$$\begin{aligned}\sum_{k=1}^{n}n_k\tilde{S_k} &= \sum_{k=1}^{n}n_k(S_0+\sigma_k\sqrt{\tau}\sum_{j=1}^{k-1}\zeta_j-\sum_{j=1}^{k-1}\tau g(\frac{n_j}{\tau})-h(\frac{n_k}{\tau}))\\ &= \sum_{k=1}^{n}\sigma_k\sqrt{\tau}\zeta_k x_k+b(n_k)\end{aligned} \tag{6}$$

Using these two decompositions we get the simplified variance:

$$\begin{aligned}\mathbb{V}\Big[P\Big] &= \mathbb{V}\Big[\frac{X}{T}\sum_{k=1}^{n}(\sigma_k\sqrt{\tau}\zeta_k\sum_{j=1}^{k}\tau)+a(n_k)-\sum_{k=1}^{n}\sigma_k\sqrt{\tau}\zeta_k x_k+b(n_k)\Big]\\ &= \mathbb{V}\Big[\frac{X}{T}\sum_{k=1}^{n}\sigma_k\sqrt{\tau}\zeta_k\sum_{j=1}^{k}\tau-\sum_{k=1}^{n}\sigma_k\sqrt{\tau}\zeta_k x_k\Big]\\ &= \mathbb{V}\Big[\sum_{k=1}^{n}\sigma_k\sqrt{\tau}\zeta_k\big(\frac{X}{T}\sum_{j=1}^{k}\tau-x_k\big)\Big]\\ &= \sum_{k=1}^{n}\sigma_k^2\tau\big(\frac{X}{T}\sum_{j=1}^{k}\tau-x_k\big)^2\end{aligned} \tag{7}$$

3.2.3 VWAP

VWAP is defined as the volume weighted average price : $VWAP=\frac{\sum_{k=1}^{n}v_kS_k}{\sum_{k=1}^{n}v_k}$

The payoff is given by $P=X\frac{\sum_{k=1}^{n}v_kS_k}{\sum_{k=1}^{n}v_k}-\sum_{k=1}^{n}n_k\tilde{S_k}$.
Let us notice that $\sum_{k=1}^{n}v_k=V$ and that $\sum_{k=1}^{n}n_k=X$.

We have:

$$\begin{aligned}
\mathbb{E}[P] &= \mathbb{E}\Big[X\frac{\sum_{k=1}^{n} v_k S_k}{\sum_{k=1}^{n} v_k} - \sum_{k=1}^{n} n_k \tilde{S_k}\Big] \\
&= \mathbb{E}\Big[\frac{X}{V}\sum_{k=1}^{n} v_k S_k - n_k \tilde{S_k}\Big] \\
&= \mathbb{E}\Big[\frac{X}{V}\sum_{k=1}^{n} V_k\big(S_0 + \sigma_k\sqrt{\tau}\sum_{j=1}^{k}\zeta_j - \sum_{j=1}^{k}\tau g(\frac{n_j}{\tau})\big) - n_k\big(S_0 + \sigma_k\sqrt{\tau}\sum_{j=1}^{k-1}\zeta_j - \sum_{j=1}^{k-1}\tau g(\frac{n_j}{\tau}) - h(\frac{n_k}{\tau})\big)\Big] \\
&= \sum_{k=1}^{n}\big(\frac{XV_i}{V} - n_k\big)\big(S_0 - \sum_{j=1}^{k-1}\tau g(\frac{n_j}{\tau})\big) - \frac{XV_k}{V}\tau g(\frac{n_k}{\tau}) + n_k h(\frac{n_k}{\tau})
\end{aligned} \tag{8}$$

Similarly as with the TWAP, we need the covariance terms to compute the Variance of the market impact so we can use the following simplification.

$$\begin{aligned}
VWAP &= \frac{\sum_{k=1}^{n} v_k S_k}{V} \\
&= \frac{1}{V}\sum_{k=1}^{n} v_k\big(S_0 + \sigma_k\sqrt{\tau}\sum_{j=1}^{k}\zeta_j - \sum_{j=1}^{k}\tau g(\frac{n_j}{\tau})\big) \\
&= \frac{1}{V}\sum_{k=1}^{n}(\sigma_k\sqrt{\tau}\zeta_k\sum_{j=1}^{k} v_j) + a(n_k)
\end{aligned} \tag{9}$$

And from [REF] we have:

$$\begin{aligned}
\sum_{k=1}^{n} n_k\tilde{S_k} &= \sum_{k=1}^{n} n_k(S_0 + \sigma_k\sqrt{\tau}\sum_{j=1}^{k-1}\zeta_j - \sum_{j=1}^{k-1}\tau g(\frac{n_j}{\tau}) - h(\frac{n_k}{\tau})) \\
&= \sum_{k=1}^{n}\sigma_k\sqrt{\tau}\zeta_k x_k + b(n_k)
\end{aligned} \tag{10}$$

Using these two decompositions we get the simplified variance:

$$\begin{aligned}
\mathbb{V}\Big[P\Big] &= \mathbb{V}\Big[\frac{X}{V}\sum_{k=1}^{n}(\sigma_k\sqrt{\tau}\zeta_k\sum_{j=1}^{k} v_j) + a(n_k) - \sum_{k=1}^{n}\sigma_k\sqrt{\tau}\zeta_k x_k + b(n_k)\Big] \\
&= \mathbb{V}\Big[\frac{X}{V}\sum_{k=1}^{n}\sigma_k\sqrt{\tau}\zeta_k\sum_{j=1}^{k} v_j - \sum_{k=1}^{n}\sigma_k\sqrt{\tau}\zeta_k x_k\Big] \\
&= \mathbb{V}\Big[\sum_{k=1}^{n}\sigma_k\sqrt{\tau}\zeta_k\big(\frac{X}{V}\sum_{j=1}^{k} v_j - x_k\big)\Big] \\
&= \sum_{k=1}^{n}\sigma_k^2\tau\big(\frac{X}{V}\sum_{j=1}^{k} v_j - x_k\big)^2
\end{aligned} \tag{11}$$

We get the last line by noticing that ζ_k are Gaussians and therefore have a variance of 1.

3.3 Results

We implemented in Python the nonlinear optimization problem we outlined in the previous sections. For our choice of parameters, we follow the original implementation by Almgren-Chriss (2001) by selecting the following values:

- Initial holdings: $X = 40000$
- Liquidation periods: $N = 50$
- Magnitude of permanent costs: $\gamma = 2.5 \times 10^{-7}$
- Magnitude of temporary costs: $\eta = 2.5 \times 10^{-6}$
- Fixed costs: $\varepsilon = 0.0625$
- Volatility: $\sigma = 0.3$

And additional parameters for the VWAP:

- Total volume of the set of trading periods: $V = 4000000$
- Volume traded per trading period i: $V_i = 80000, \;\; i = 1, ..., N$
- Initial stock price: $S_0 = 100$

3.3.1 Optimal liquidation trajectories

For each benchmark we look at the impact of the risk aversion coefficient λ on trading trajectories.

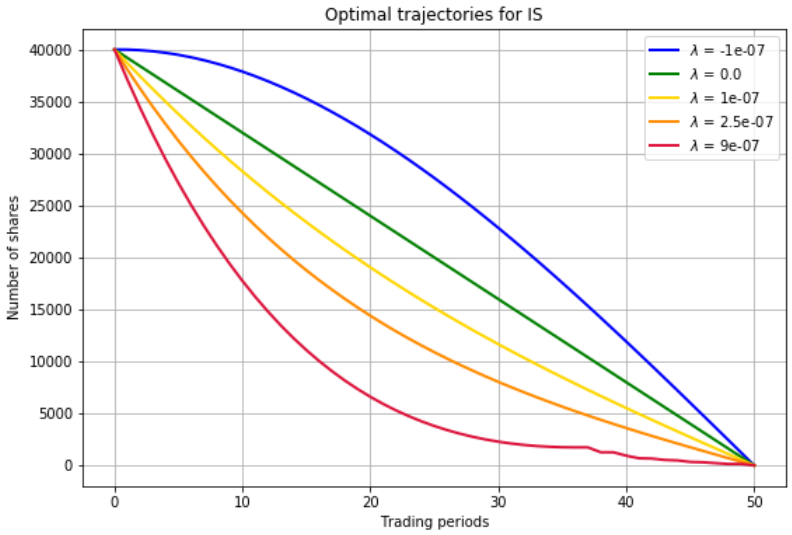


Figure 1: Optimal liquidation for implementation shortfall

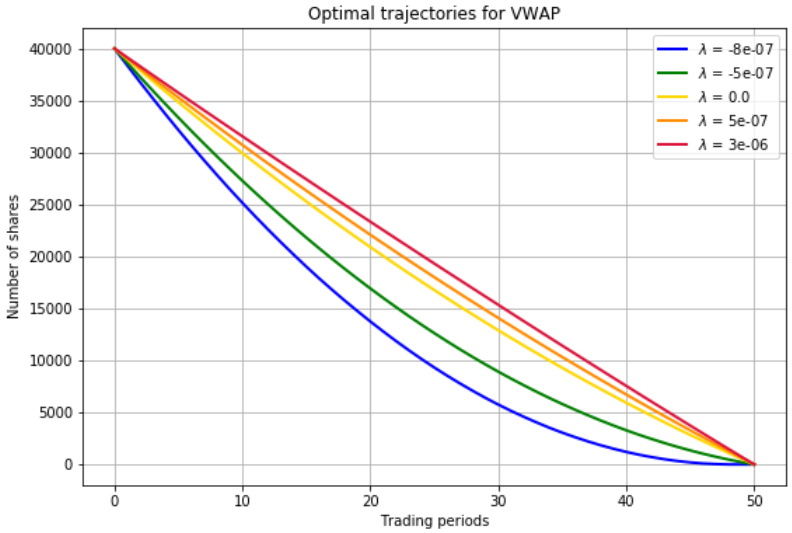


Figure 2: Optimal liquidation for VWAP

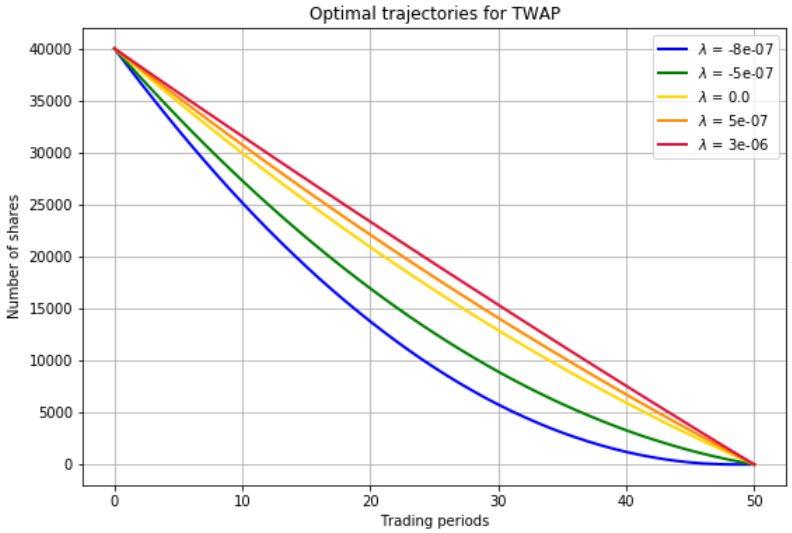


Figure 3: Optimal liquidation for TWAP

We see that for the TWAP and VWAP, the trading trajectories don't seem to be as sensitive as Implementation Shortfall to the choice of risk aversion coefficient.

But we do notice that the relationship between risk aversion and the liquidation trajectory invert when going from the VWAP/TWAP to Implementation Shortfall. In the case of the latter, a high risk aversion leads to a very fast liquidation of the inventory while with a small one we wait longer before liquidating the inventory, which is in line with what one can expect. But for the former, we get the opposite scenario: the risk-averse agent will adopt a wait-and-see approach while the risk-taking agent will liquidate at a much faster pace. One possible explanation is that the risk averse agent is waiting for the stock price to converge towards the benchmark VWAP/TWAP over the course of the liquidation

process.

One thing to notice is that the trajectories for VWAP and TWAP are identical. This is confirmed by O. Gueant (2016) [4] who states that both VWAP and TWAP coincide with each other in the case of a flat market volume, which is the case with our implementation.

Compared to the existing literature, the trajectories seem to be in line with the results from Almgren-Chriss (2001) [2] in the case of implementation shortfall. The VWAP/TWAP trajectories seem to be in line with the trading curve found in Gueant (2016) [4].

3.3.2 Efficient frontiers

Now let us compare the efficient frontiers for each benchmark:

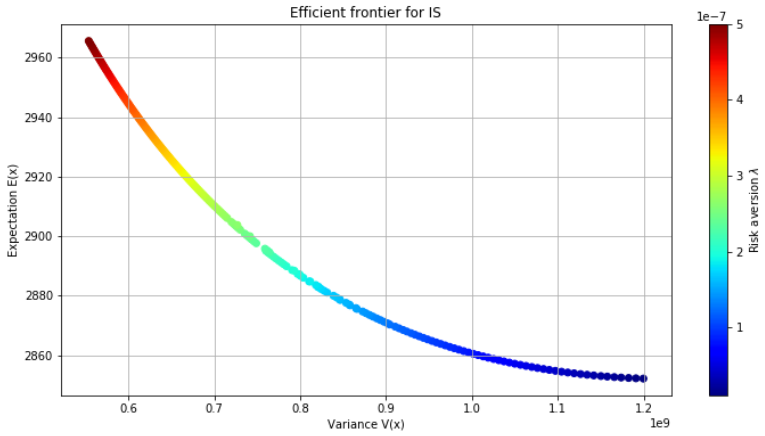


Figure 4: Efficient frontier for implementation shortfall

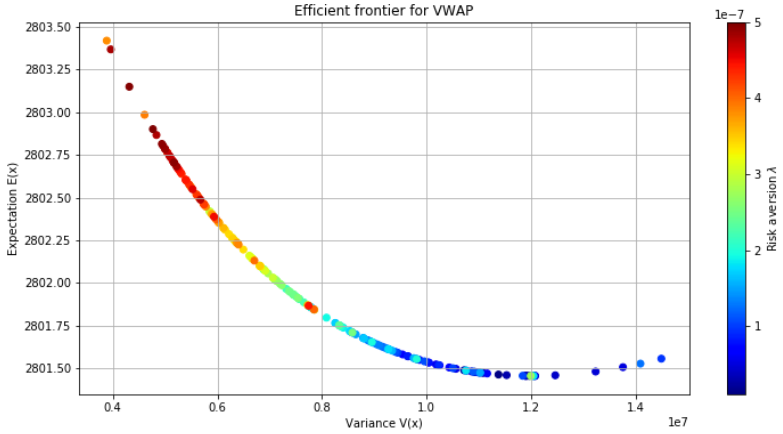


Figure 5: Efficient frontier for VWAP

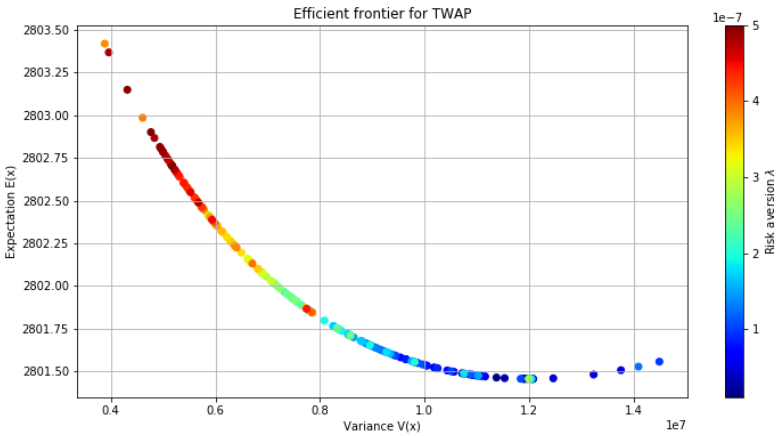


Figure 6: Efficient frontier for TWAP

These results seem to confirm that our utility functions and optimization procedures are capable of yielding a (convex) efficient frontier of optimal liquidation strategies that provide a level of timing risk $\mathbb{V}(x)$ for a given level of expected transaction cost $\mathbb{E}(x)$.

Also, depending on the trader's risk aversion it might be better to use either VWAP (on the left part of the graph as it has the steepest frontier for the left part) either IS (on the right part of the graph).

We do observe a strange feature on these curves: on the right, the frontier is up for the VWAP and the TWAP benchmarks, thus yielding to non optimal choices. For the IS benchmark, we do not observe this.

3.3.3 Choice of market impact parameters

We only focus on the choice of degree for the power functions used in the market impact functions. Recall that we use the following market impact functions:

$$g(v) = \gamma v^{\alpha}$$

$$h(v) = \varepsilon \operatorname{sign}(v) + \eta\Big(\frac{v}{\tau}\Big)^{\beta}$$

We relaunch our efficient frontier algorithms on the VWAP for the following values:

- $\alpha = 1, \beta = 1$
- $\alpha = 1, \beta = 0.5$
- $\alpha = 0.75, \beta = 1$
- $\alpha = 0.5, \beta = 1$
- $\alpha = 0.75, \beta = 0.5$

This yields the following efficient frontiers:

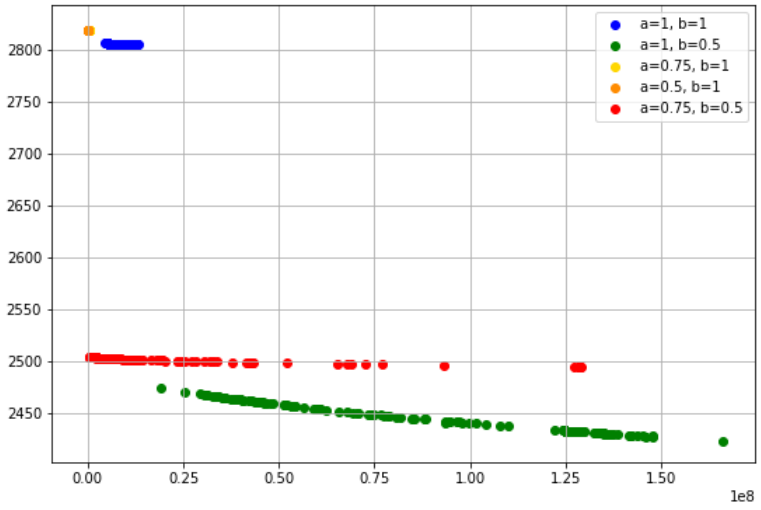


Figure 7: VWAP efficient frontiers and market impact power coefficients α, β

The efficient frontiers using a low β, ie a low temporary market impact (in green and red) are much more spread out than the one using a high β.

4 Solving Almgren-Chriss through dynamic programming

4.1 Dynamic programming and the Bellman equation

Note: we will limit the Almgren-Chriss implementation here to implementation shortfall, since we were unable to formulate Bellman equations for VWAP and TWAP.

What are the justifications for using dynamic programming to solve the Almgren-Chriss framework? According to D. Silver [5], dynamic programming can be used in problems that satisfy the following 2 properties:

- **Optimal substructure**: the optimal solution can be broken down into subproblems.
- **Overlapping subproblems**: subproblems recur multiple times and the solutions can be stored and reused.

One field dynamic programming can be used for is the solving of Markov decision processes (MDP), which is for a given set of states, a Markov chain with rewards (gain, loss) for each potential state and action (compared to the simple Markov chain, the agent in this setting has more control over his decision making). More importantly, the MDP has a state-value function and an action-value function which both respectively tell how is it to be in a particular state and to take a particular action (this state value function is known as the Bellman equation). Dynamic programming will look to find best policy (i.e. a set of actions to take) among the set of all possible policies, the one that will maximize/minimize the value function.

This effectively applies for the Almgren-Chriss optimal execution framework, which can be modeled as a Markov decision process: both optimal substructure (i.e. the objective of optimal liquidation at the lowest cost can be broken down into multiple trading periods) and overlapping sub-problems are applicable here (since the same issues of portfolio liquidation recur for each trading period and a value function can evaluate how well the liquidation is going). We also know that the Almgren-Chriss by definition terminates (no issue of infinite MDPs) and the final rewards (i.e.

the cost of liquidation) are known at the end. This makes dynamic programming a suitable technique for solving Almgren-Chriss.

Note: compared to reinforcement learning algorithms (ex. temporal difference and Monte Carlo methods) which are model-free, dynamic programming assumes that the agent has full knowledge of the MDP and its internal transition probabilities between different states, which can then be used for finding the best policy (control problem).

Now that we have justified the use of dynamic programming with the Almgren-Chriss framework, how do we translate this into a model that can be implemented? The core principle of this approach is first to transition from the mean-variance problem (used in nonlinear programming) into the Bellman equation (used in dynamic programming).

Consider that the market impact benchmark $X_B = XB - \sum_{k=1}^{n} n_k \tilde{S_k}$ is a Gaussian random variable. If we still use the CARA utility function, we have:

$$\min_n \; e^{\gamma \mathbb{E}\left(XB - \sum_{k=1}^{n} n_k \tilde{S_k}\right) + \frac{\gamma}{2} \mathbb{V}\left(XB - \sum_{k=1}^{n} n_k \tilde{S_k}\right)}$$

which as we previously showed can be transformed into

$$\min_n \; \mathbb{E}\Big(XB - \sum_{k=1}^{n} n_k \tilde{S_k}\Big) + \lambda \mathbb{V}\Big(XB - \sum_{k=1}^{n} n_k \tilde{S_k}\Big)$$

Since the minimization of the CARA utility function is equivalent to the minimization of an expectation, we can consider that the Almgren-Chriss framework can be also be reformulated as the minimization of an expectation:

$$\min_n \mathbb{E}(e^{-\gamma X_B}) = \min_n \mathbb{E}(e^{-\gamma X S_0 - \gamma \sum_{i=1}^{N} n_i \tilde{S}_i}) = \min_n \mathbb{E}(e^{-\gamma \sum_{i=1}^{N} n_i \tilde{S}_i})$$

Let's consider s as the stock price, x as the inventory, r the revenue of the liquidation and t the time step, the Almgren-Chriss problem thus leads to the following Bellman equation:

- Terminal condition:

$$v(r, x, s, T-1) = e^{-\gamma \left[r + x\left(s - g\left(\frac{x}{\tau}\right)\right)\right]}$$

- Backwards equation:

$$v(r, x, s, t) = \min_n \; \mathbb{E}(v^*, x^*, s^*, t+1)$$

$$r^* = r + x\Big[s - g\Big(\frac{n}{\tau}\Big)\Big]$$

$$s^* = s + \sigma\zeta\sqrt{\tau} - \tau h\Big(\frac{n}{\tau}\Big)$$

$$x^* = x - n$$

Implementing this Bellman equation however is fraught with difficulties. Since stock prices s can take many different values, so thus the revenue r and inventory x. Any combination (x, t) (which can be modelled as a 2×2 matrix) can take any value for the stock price s. Similarly the tensor (x, t, s) can take any possible value for the revenue r, thus creating a tensor (x, t, s, r). Therefore, this implementation suffers from the curse of dimensionality, and solving

Almgren-Chriss with these sets of equations will become a computational nightmare (especially since our updates are synchronous and do full sweeps). An illustration of the curse of dimensionality is provided in the following figure:

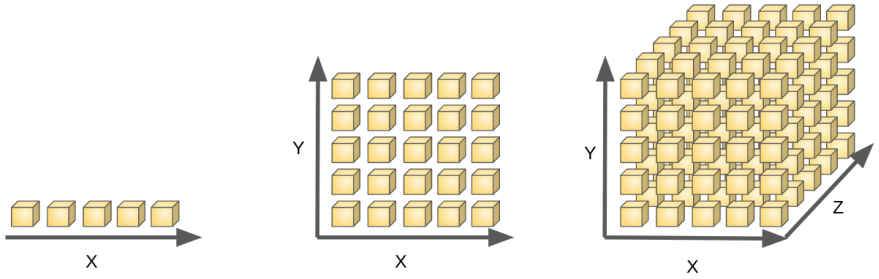


Figure 8: An illustration of the curse of dimensionality

The first solution to the curse of dimensionality is to approximate the value function $v(.)$ with methods such as reinforcement learning (ex. Q-Learning). Another approach (specific to implementation shortfall) is to simplify directly $v(.)$:

$$v(r, x, s, t) = u(x, t)e^{-\gamma(r+xs)}$$

with $u(x, t)$ being a simplified value function that only depends on x and t. We thus eliminated the variables s and r from the Bellman equation and reduced our value function matrix from a 4-dimensional tensor to a 2-dimensional matrix. We can know implement our dynamic programming principle on $u(t, x)$, instead of $v(r, x, s, t)$. Thus:

- Terminal condition:

$$u(x, T-1) = e^{\gamma x g\left(\frac{x}{\tau}\right)}$$

- Backwards equation:

$$u(x,t) = \min_{0<n<x} u(x-n, t+1)\, e^{\gamma n g\left(\frac{n}{\tau}\right) + \gamma(x-n)\tau h\left(\frac{n}{\tau}\right) + \frac{1}{2}\gamma^2(x-n)^2\sigma^2\tau}$$

Note: this change of variable relies on the assumption that $r + xs$ is the mark-to-market valuation of our portfolio at any time t during the entire liquidation process.

4.2 Results

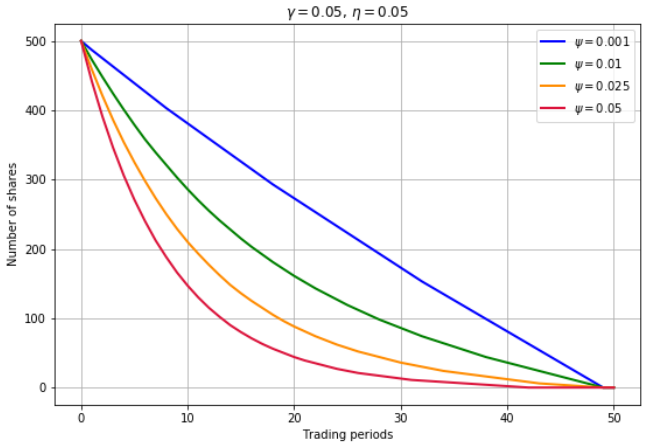
We run an implementation in Python of the Bellman equation $u(x, t)$.

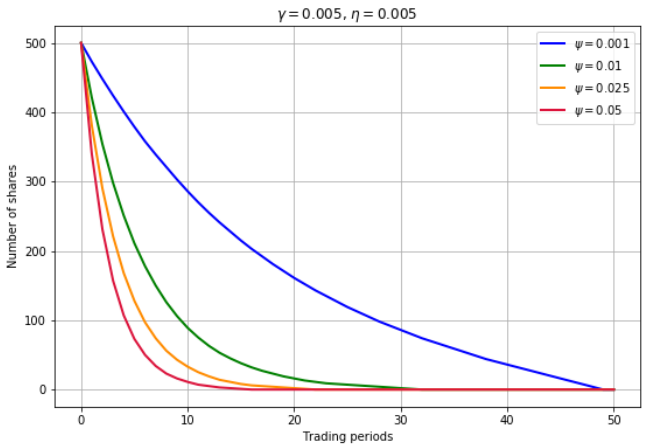
4.2.1 Simulations

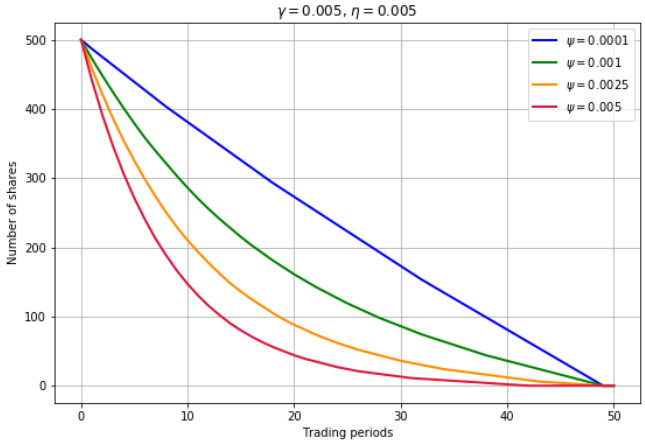
Since there are no papers that implement this particular approach to solving Almgren-Chriss, we ran a battery of simulations in order to find the best parameters that will yield realistic (curved) liquidation curves. We decided to play with the three following parameters:

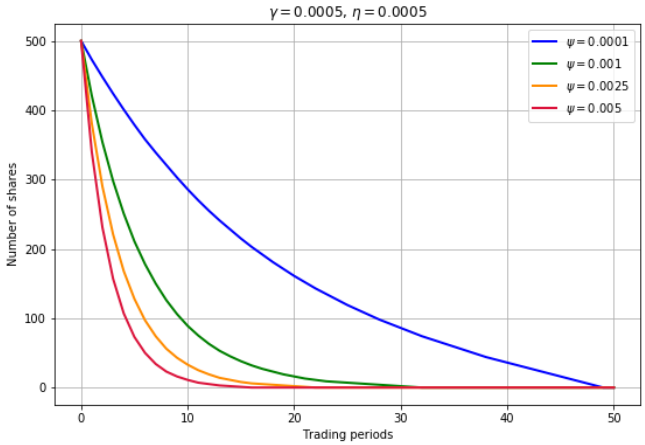
- Risk aversion coefficient ψ
- Magnitude of permanent impact γ
- Magnitude of temporary impact η

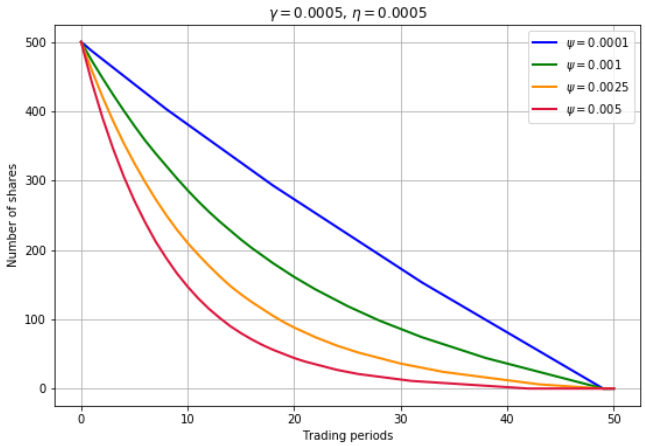
Assuming that $X = 500$, $\alpha = 1$, $\beta = 1$, $\tau = 0.5$ and $N = 50$, we ran 32 dynamic programming simulations with varying values for ψ, γ and η:






$\gamma = 0.005$, $\eta = 0.005$
$\psi = 0.0001$
$\psi = 0.001$
$\psi = 0.0025$
$\psi = 0.005$
Number of shares
Trading periods


$\gamma = 0.0005$, $\eta = 0.0005$
$\psi = 0.0001$
$\psi = 0.001$
$\psi = 0.0025$
$\psi = 0.005$
Number of shares
Trading periods


$\gamma = 0.0005$, $\eta = 0.0005$
$\psi = 0.0001$
$\psi = 0.001$
$\psi = 0.0025$
$\psi = 0.005$
Number of shares
Trading periods

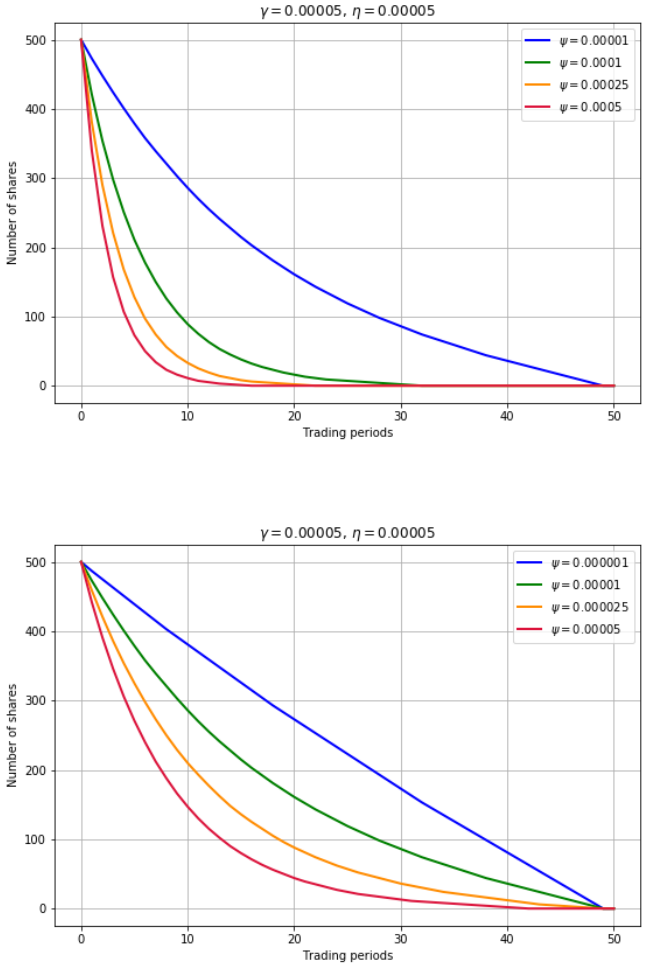


Figure 9: Hyperparameter testing of ψ, γ and η on the Bellman equation

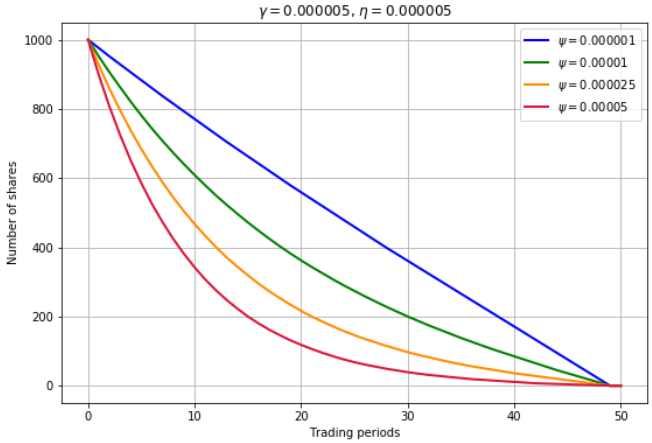
These tests clearly show that it is possible to match the liquidation curves obtained through nonlinear programming, as our set of parameters closely converges to the inputs we used in the computing of the efficient frontiers (recall that we chose $\gamma = 2.5 \times 10^{-7}$ and $\eta = 2.5 \times 10^{-6}$).

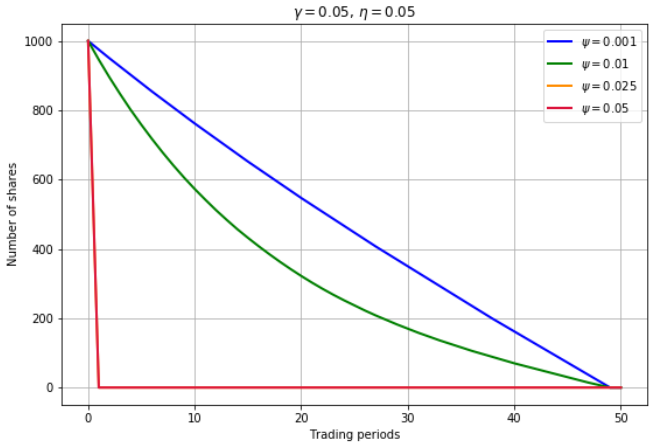
We do need to ensure however that our liquidation curves are robust for different levels of initial inventory X.

4.2.2 The issue of the inventory size X

While the change of variable has made the implementation of the Bellman principle more manageable, it is still computationally expensive. As an illustration, we attempted to run our Python implementation of Bellman's equation on an initial inventory of $X = 40000$ shares since we used the same amount for our nonlinear programming implementations. The implementation lasted six hours. This is the reason why we decided to only implement the code on inventories of $X = 500$ shares.

But does the size of the initial inventory X play a role in the liquidation? We decide to run the same simulations again, this time with double the initial inventory amount ($X = 1000$ shares):





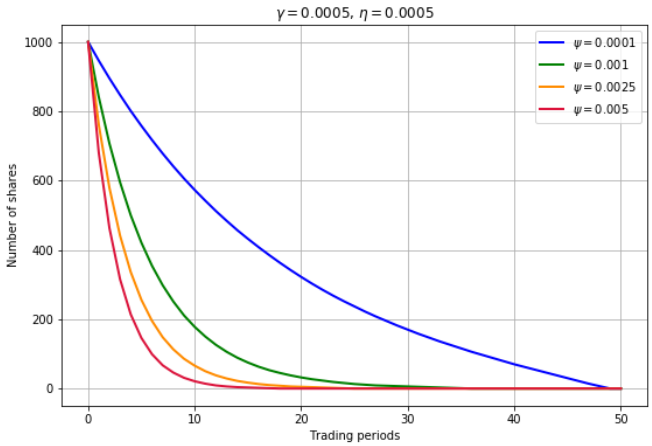


Figure 10: Hyperparameter testing of inventory X

These simulations seem to show that for smaller values of γ and η, the liquidation curves are more robust to changes in X. They won't give us a definite answer over the set of parameters γ,η or ϕ we are supposed to use should we decide to run the Bellman equation with $X = 40000$. But at the very least they do give us a hint of how the liquidation curves behave when X is increased.

4.2.3 Comparison between dynamic and nonlinear programming

We implemented both approaches with a very similar set of parameters:

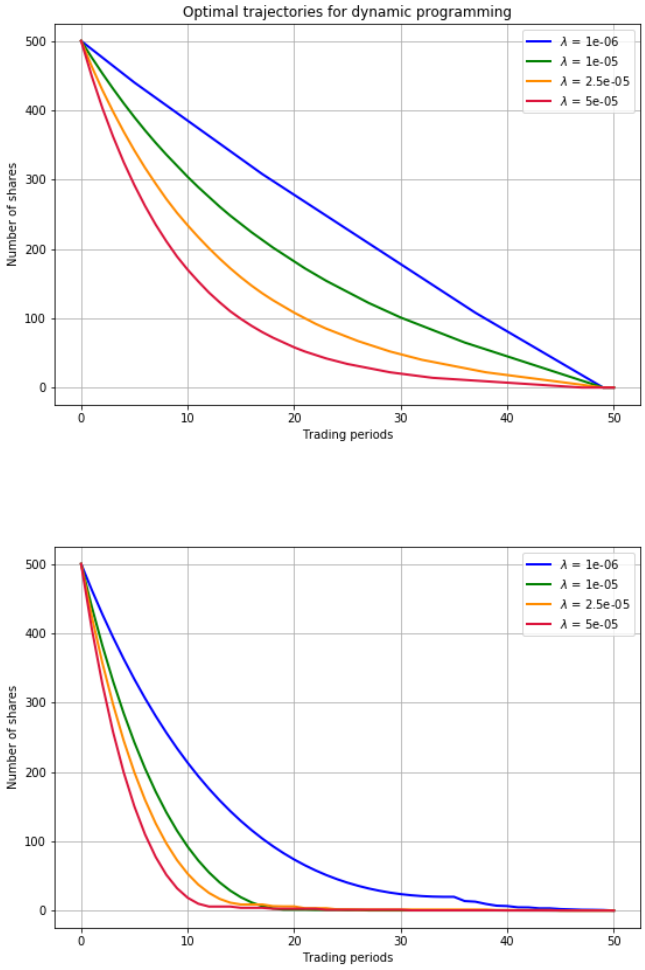


Figure 11: Optimal liquidation curves for dynamic (1st) and quadratic programming (2nd)

The comparison between both approaches is difficult due to the fact that we are not dealing with the same initial inventory X (500 for dynamic against 40000 for nonlinear) and we had to rely on brute force to find suitable parameters that yielded realistic liquidation curves (in the case of dynamic programming).

It is clear that the choice between nonlinear optimization and dynamic programming shouldn't be based on the idea that both approaches are mutually exclusive. The motivations and purposes for both methods differ:

- Dynamic programming: it allows us to model more realistically how portfolio liquidation works, integrate

dynamic parameters (volatility) more seamlessly and has many promising research venues (especially with the rising popularity of reinforcement learning). Dynamic programming however is very limited by the curse of dimensionality and we encountered computational difficulties.

- Nonlinear programming: it is much faster than the dynamic programming approach and allows for *ceteris paribus*-type comparative static analysis through the shifting parameters (all things considered equal) in order to assess their impact. It is however limited in scope.

5 Volatility estimation, microstructure biases and Almgren-Chriss

5.1 Volatility estimation and microstructure noise

We have seen how to solve the Almgren-Chriss framework either through nonlinear optimization or dynamic programming. But we did make simplistic assumptions on the real-life market parameters. We can now augment Almgren-Chriss by integrating microstructure biases (bid-ask spread and tick size).

The unexpected nature of events and signals in the real-life environment of optimal liquidation forces us to rely on alternative metrics for estimating key parameters, such as volatility (which can be seen as how likely a stock price will go up or down). Let's take for example the naive volatility estimator, i.e. sum of sampled squared returns, given by

$$[X, X]_T = \sum_{t_i} (X_{t_{i+1}} - X_{t_i})^2$$

5.1.1 Model-free approach

According to Zhang, Mykland and Ait-Sahalia (2005) [6], using this estimator to compute volatility on intra-day data forces you into the following conundrum: if you sample the price very closely (low sampling time interval), you will have few data-points and lots of statistical errors (high variance). Yet if your sampling time interval is very high (high-frequency), then you will take into account lots of noise.

Reducing sampling frequency is not sufficient to correct the volatility estimator for microstructure biases, it merely increases its impact on volatility estimation. One common solution used by researchers (also done here by Zhang, Mykland and Ait-Sahalia) is to model stock prices as:

$$Y_t = X_t + \varepsilon_t$$

where X_t is a latent variable for the unobserved (but "real" stock prices), Y_t our best estimation of the latent returns and ε_t some microstructure noise such that they are not by nature embedded in the dynamics of the stock price.

In their paper, Zhang, Mykland and Ait-Sahalia offer 5 volatility estimators to implement: the naive estimator, the naive estimator sampled to lower frequencies, an estimator with its optimal frequency estimated through least squared regression, an estimator that takes the average of subsampled estimators and the same previous estimator minus a correction (here the naive estimator).

We will focus on their "first-best approach", the average of subsampled estimators corrected by the naive estimator. We start by computing the naive estimator:

$$[Y, Y]_T^{(all)} = \sum_{i=1}^{N} (Y_{t_{i+1}} - Y_{t_i})^2$$

with N the number of observation of our data sample over all period T. Since $[Y,Y]_T^{(all)}$ is prone to market microstructure biases but at the same time it is hard to find the optimal sampling frequency for the volatility estimator. One solution proposed by Zhang, Mykland and Ait-Sahalia is to average estimators $[Y,Y]_T^{(k)}$ across K grids of average size $\bar{n}$, given by $[Y,Y]_T^{(avg)}$:

$$[Y,Y]_T^{(avg)} = \frac{1}{K}\sum_{k=1}^{K}[Y,Y]_T^{(k)}$$

The "first-best approach" combines both the naive estimator $[Y,Y]_T^{(all)}$ and the average of subsampled estimators $[Y,Y]_T^{(avg)}$, and gives an estimation of the latent process X_t:

$$[X,\hat{X}]_T = [Y,Y]_T^{(avg)} - \frac{\bar{n}}{n}[Y,Y]_T^{(all)}$$

where n is the number of observations, $\bar{n}$ is the subsample size of a grid (which we get through $\bar{n} = \frac{n}{K}$) and $K = cn^{2/3}$ is the number of grids with $c = \left(\frac{T}{12(E\varepsilon^2)^2}\int_0^T \sigma_t^4 dt\right)^{-1/3}$. Finally, we compute the adjusted estimator:

$$[X,\hat{X}]_T^{(adj)} = \left(1 - \frac{\bar{n}}{n}\right)^{-1}[X,\hat{X}]_T$$

5.1.2 Alternative approaches

Instead of using the model-free estimator from Zhang, Mykland and Ait-Sahalia, we could use instead the following estimators, which can be found in Bennett & Gil (2012) [7]. The goal of all of these models detailed in [7] is to solve the shortcomings of the naive approach in order to have a volatility estimation that reflects more closely the reality of financial market microstructure. Those issues are :

- Intra-Trading period Information
- Non-Zero Drift
- Jumps

a. Parkinson Estimator: the issue with the usual way to calculate volatility is that we only take the Close-to-close price. Thus the estimator does not really capture the information of the price moves during the day. To correct this drawback, Parkinson used the High(H) and Low(L) prices for each period of computation. It allows to better capture the moves during the period of computation.

$$\sigma_{Park} = \sqrt{\frac{F}{N}}\sqrt{\frac{1}{4\ln(2)}\sum_{i=1}^{N}\left[\ln(\frac{h_i}{l_i})\right]^2}$$

b. Garman-Klass estimator: there are different methods of calculating volatility using some or all of the open (O), high (H), low (L) and close (C). The Garman-Klass estimator is an extension of Parkinson which includes opening and closing prices. The limit of the Parkinson estimator is that even if it takes into account the variation within a trading period, it does not take into account the Open and the Close whereas there are the moment where the trading volumes are the most intense, thus they are non-negligible (if opening prices are not available the close from the previous day can be used instead).

$$\sigma_{GK} = \sqrt{\frac{F}{N}}\sqrt{\sum_{i=1}^{N}\Big[\frac{1}{2}[\ln\Big(\frac{h_i}{l_i}\Big)\Big]^2 - (2\ln(2)-1)\Big[\ln(\frac{c_i}{o_i})^2\Big]}$$

c. Rogers-Satchell Estimator: all the previous models tried to provide a consistent solution to estimate the volatility. However, they all assumed a zero-drift on average. They assumed there was no trend for the price path. Usually, for option pricing, we use a Geometric Brownian motion with a drift to model the stock price path. Consequently, it is a flaw that has to be fixed. Rogers Satchell tried to solve this issue by using the product of the log fraction of both high and low over the open and the close. Then, they take the difference of both. However, it underestimates volatility as it does not take jumps into account.

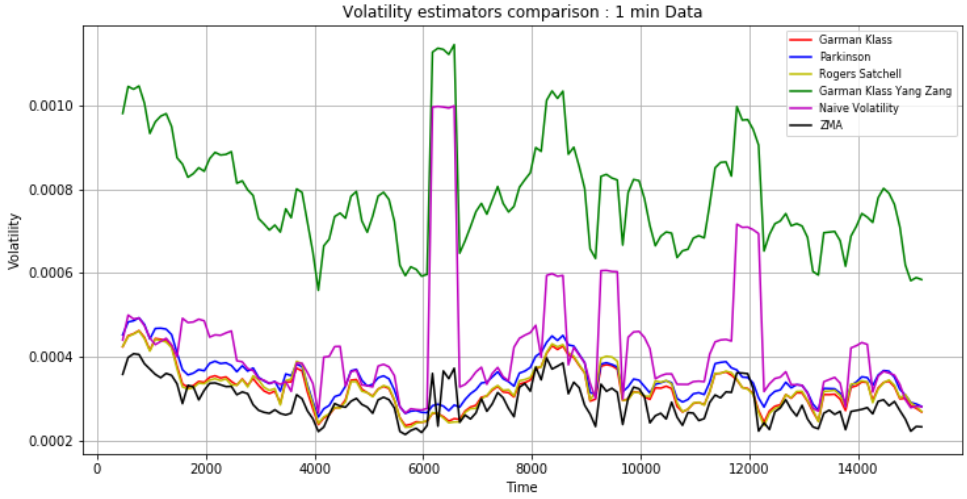
$$\sigma_{RS} = \sqrt{\frac{F}{N}}\sqrt{\sum_{i=1}^{N}\Big[ln\Big(\frac{h_i}{c_i}\Big)\ln\Big(\frac{h_i}{o_i}\Big) + \ln\Big(\frac{l_i}{c_i}\Big)ln\Big(\frac{l_i}{o_i}\Big)\Big]}$$

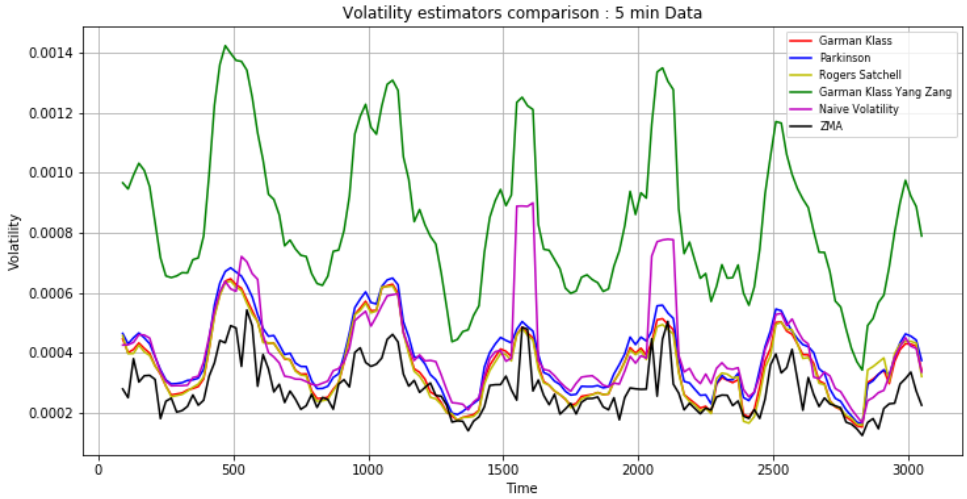
d. Garman-Klass Estimator with Yang-Zang Extension: one of the main drawbacks of the estimators previously introduced are that they do not take into account jumps in prices. This involves a downward bias, because volatility is underestimated. Yang Zang tried to provide an extension to the Garman-Klass estimator so that jumps are handled. The most common jumps come from the change between closing and opening. To take it into account, they simply add another commponent to the Garman-Klass model which is the squared log fraction between the closing of the previous day and the opening today. This will result in an estimation which is always greater than the Garman-Klass one unless closing and opening are equals through the whole trading period.

$$\sigma_{GKYZ} = \sqrt{\frac{F}{N}}\sqrt{\sum_{i=1}^{N}\Big(\Big[\ln\Big(\frac{o_i}{c_{i-1}}\Big)\Big]^2 + \frac{1}{2}\Big[\ln\Big(\frac{h_i}{l_i}\Big)\Big]^2 - (2\ln(2)-1)\Big[\ln\Big(\frac{c_i}{o_i}\Big)\Big]^2\Big)}$$

5.2 Implementation on market data

We implement our volatility estimators on intraday market data (sampling frequencies are 1 minute, 5 minutes and 10 minutes) of Total SA extracted on Bloomberg, from March 1st 2018 to April 13th 2018.


Volatility estimators comparison : 1 min Data
Garman Klass
Parkinson
Rogers Satchell
Garman Klass Yang Zang
Naive Volatility
ZMA
Volatility
0.0010
0.0008
0.0006
0.0004
0.0002
0
2000
4000
6000
8000
10000
12000
14000
Time


Volatility estimators comparison : 5 min Data
Garman Klass
Parkinson
Rogers Satchell
Garman Klass Yang Zang
Naive Volatility
ZMA
Volatility
0.0014
0.0012
0.0010
0.0008
0.0006
0.0004
0.0002
0
500
1000
1500
2000
2500
3000
Time

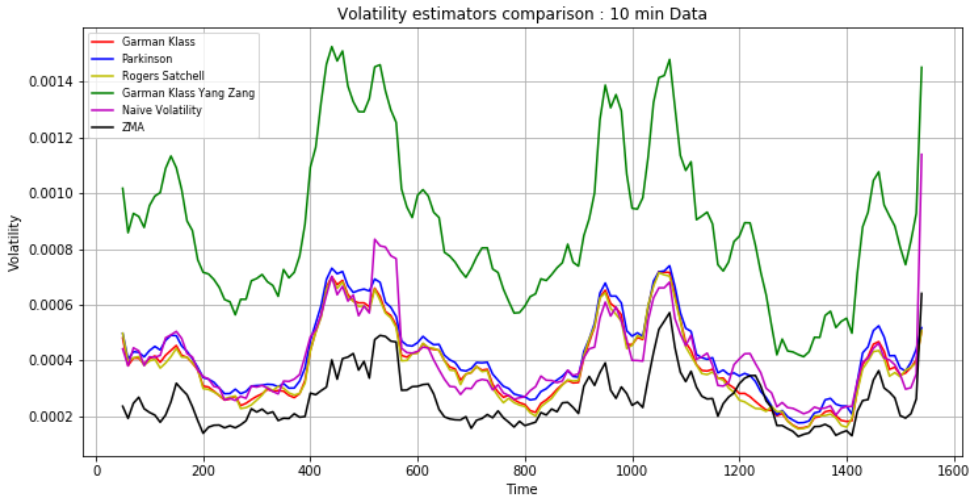


Figure 12: Volatility estimators on intraday market data from Total SA

We start by looking at 1 minute intraday data. From there, we can categorize the estimators into 3 categories:

- **Highest estimation of volatility**: Garman-Klass-Yang-Zang. This is due to the added term that involves a clear upward bias.
- **Overreaction to jumps**: Naive volatility. The naive volatility approach overreacts to the jumps which creates an unsmoothness in the estimator
- **Stable/Downward biased estimators**: Parkinson, Rogers-Satchell, Garman-Klass and ZMA got quite similar values even if ZMA is always on the lower bound and Parkinson on the upper bound.

We have two extreme cases:

- First, there is the Garman-Klass-Yang-Zang estimator. It seems to overestimate volatility. However, the strength of this model is that it takes into account jumps in a smooth way which allows to have less variance in volatility estimation compared to the naive estimator. Nonetheless, we may question the use of the jump parameter during all the trading period as it triggers a clear upward bias.
- For the stable estimators, we observe few differences among the estimates. It is because there are not, or at least much less, sensitive to jumps. This lack of sensitivity seems to be one of the root cause for this underestimation. Then, it is really interesting to notice that Garman-Klass and Rogers-Satchell yield almost the same values during the whole period. This is certainly due to the fact that they both use differences of product of log fractions. As for ZMA, it seems to be always on the lower bound as it basically smooth the Naive Volatility, then subtracts the value of the Naive Volatility from the smoothed estimation. We can notice that for small values of K, it almost matches the other stable estimators. Parkinson is always on the upper bound because it only takes the log fraction of high over low, hence some precious information is missing.

If we switch to 5 minute intraday day, the results confirms our analysis on the 1 minute data. As the data is smoother, some facts are more obvious:

- It is clear that Garman-Klass-Yang-Zang is over sensitive to jumps which creates a very important variance of the estimator. The gap between Garman-Klass-Yang-Zang seems to almost double in periods of jumps.
- ZMA is still on the lower bound of the downward biased estimators. We can notice clearly that is not sensitive to jumps.

- Parkinson, Garman-Klass and Rogers-Satchell estimators seem to have even closer results, which means that with smoothed data, the impact of the adjustment made are minimal.
- Naive volatility is less smoother and matches Parkinson, Garman-Klass and Rogers-Satchell in periods with no jumps.

Finally, when we switch to 10 minute intraday data, our previous analysis is further confirmed: first, naive volatility matches Parkinson, Garman-Klass and Rogers-Satchell which are really close to each other. ZMA is still on the lower bound but seems to be more sensitive to jumps than with higher frequency data. Finally, this graph does not add new insight on the Garman-Klass-Yang-Zang estimator

5.3 Integrating dynamic volatility into Almgren-Chriss

5.3.1 Nonlinear programming

We wanted to stress-test the AC algorithm when it faces different volatilities. Using the volatility estimators defined previously we tested the quadratic optimization with these volatilities for different benchmarks (TWAP, VWAP and IS) and different frequencies (1 min., 5 min., 10 min.). Our intuition was that the volatility should impact the trading trajectory.

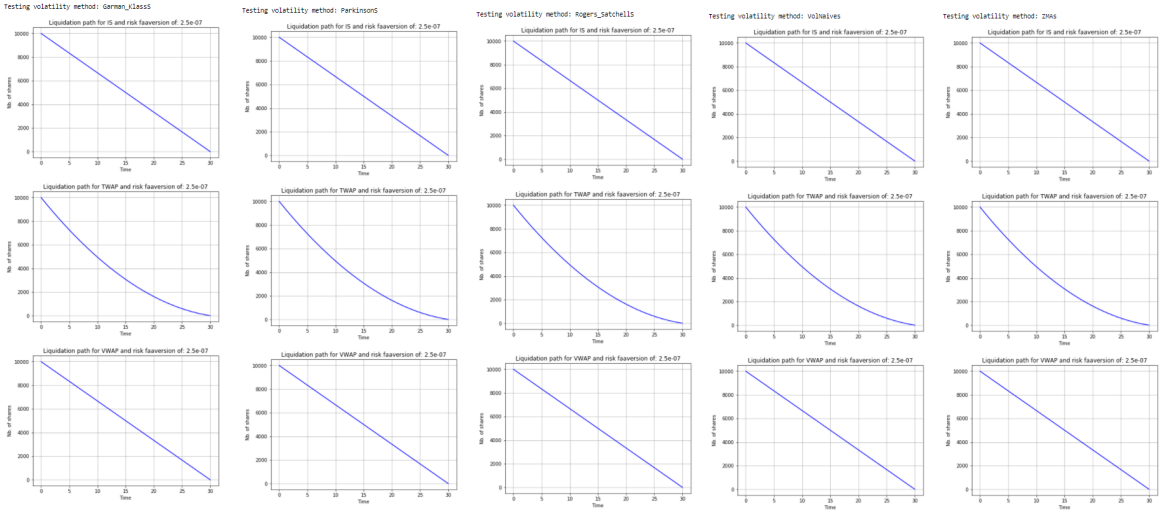


Figure 13: Dynamic volatility and nonlinear optimization

As you can see on the graph above, for a frequency of 1 min., the use of a different volatility estimator does not seem to change anything. It remains true for other frequencies. For market practitioners this means that they can use a naive volatility and do not need to take care of this. This might be less true in a continuous time framework with bid, asks and market depth.

5.3.2 Dynamic programming

Finally, we run the same volatility estimators on the dynamic programming implementation of Almgren-Chriss. For this set of rolling window volatility estimators applied on Total SA, we found that it had little impact on the liquidation. This is most likely because our volatility parameter in the Bellman equation only seems to respond to large changes in

σ. Thus the benefits of including dynamic volatility in the liquidation algorithm clearly dependents on the choice of parameters for ψ, γ, η and X which adds even further complexity to the implementation.

6 Conclusion

The Almgren-Chriss model has become the canonical reference for optimal execution research. This is mainly due to the following reasons:

- Allows one to graphically represent the trade-off between slow execution with low transaction costs but high timing risk, versus a faster execution with lower exposure to future price fluctuations but higher transaction costs;
- Ease to choose one's own preferred approach to model market impact costs: while we used linear market impact functions, it is possible to use more complex market impact functions that take into market phenomenology. Their integration into Almgren-Chriss is seamless;
- Avoids the temptation of modeling the market as a physical system fully enclosed in its own data (which is commonly referred to as *physics envy*);
- Designed from a bottom-top approach: the Almgren-Chriss model was not built in the same way as traditional models in mathematical economics or finance. The framework is at first a simplistic modeling of the execution process, but its flexibility allows for numerous enhancements;
- Provides a multitude of modeling approaches to compute the slow or fast execution trade-off: e.g. nonlinear optimization, dynamic programming, stochastic control, reinforcement learning, etc.

Through our paper, we have done the following:

- Computed the efficient frontiers and optimal liquidation curves (with nonlinear programming) for the implementation shortfall, VWAP and TWAP algorithms. Our results were consistent with the original paper by Almgren-Chriss and research literature;
- Implemented a simplified version of the Bellman equation for solving the Almgren-Chriss model;
- Implemented numerous volatility estimators that take into account microstructure biases and integrated them into both the nonlinear and dynamic programming approaches.

Our main results are the following:

- The IS benchmark is much more sensitive to the risk aversion factor than VWAP and TWAP benchmarks;
- Using different volatility measures does not impact the trading trajectories of the three benchmarks.

Possible future avenues of research:

- Implement stochastic control methods (ex. Hamilton-Jacobi-Bellman equation and nonlinear PDEs) for solving the Almgren-Chriss framework;
- Implement reinforcement learning algorithms such as Q-Learning. This will necessitate market data from order books (ex. Robinson-Monroe algorithm);
- Trading algorithms for limit orders instead of market orders: implementing variants of dynamic programming which include the simplified Bellman equation applied to limit orders or the Avellaneda-Stoikov market making model.

References

[1] B. Bertsimas, A. Lo. Optimal control of execution costs. *Journal of Financial Markets*, 1(1):1-50, 1998.

[2] R. Almgren, N. Chriss. Optimal execution of portfolio transactions. *Journal of Risk*, 3:5-40, 2001.

[3] R. Almgren, C. Thum, E. Hauptmann, H. Li. Direct estimation of equity market impact. *Risk*, 18(7):58?62, 2005.

[4] O. Gueant, *The Financial Mathematics of Market Liquidity: From Optimal Execution to Market Making*, CRC Press, Taylor & Francis Group LLC, 2016

[5] D. Silver: Reinforcement Learning, UCL
`http://www0.cs.ucl.ac.uk/staff/d.silver/web/Teaching.html`

[6] L. Zhang, Per A. Mykland, Y. Ait-Sahalia. Determining Integrated Volatility With Noisy High-Frequency Data. *Journal of the American Statistical Association*, Vol. 100, No. 472, 2005

[7] C. Bennett, M. A. Gil. Measuring historical volatility: Close-to-Close, Exponentially Weighted, Parkinson, Garman-Klass, Rogers-Satchell and Yang-Zhang Volatility. Santander Global Banking & Markets, Equity Derivatives, February 3, 2012.