Shrinking Horizon, Scenario-based Optimal Liquidation with Lower Partial Moments Criteria

by

Hassan Anis

A thesis submitted in conformity with the requirements for the degree of Master of Applied Science

Department of Mechanical & Industrial Engineering,
University of Toronto

© Copyright by Hassan Anis, 2018
Shrinking Horizon, Scenario-based
Optimal Liquidation with Lower Partial
Moments Criteria

Hassan Anis
Master of Applied Science

Department of Mechanical & Industrial Engineering
University of Toronto

Abstract

A quasi-multi-period model for optimal position liquidation in the presence of market impact is proposed. Two features distinguish the approach from alternatives. First, a shrinking horizon framework is implemented to update intraday parameters by incorporating new information while maintaining standard non-anticipativity constraints. The method is data-driven, numerically tractable, and reactive to the market. Second, lower partial moments, a downside risk measure, is used which captures traders’ increased risk aversion to losses better than symmetric risk measures. The performance of the proposed strategies is tested using historical, high-frequency New York Stock Exchange (NYSE) data. The proposed strategies outperform their benchmark on days with unfavorable market conditions, strongly supporting the use of lower partial moments as a risk measure. Additionally, results validate the use of a shrinking horizon framework as an adaptive, tractable alternative to dynamic programming for trading.
Acknowledgements

First and foremost, I would like to express my deepest gratitude towards my supervisor Prof. Roy Kwon. Firstly, for his support throughout my studies. He has been instrumental in giving me guidance whenever I was lost, while also giving me the freedom to pursue research directions that I found of interests. His feedback, advice, and encouragement have been invaluable. This thesis could not have come together without him. I look forward to many more years of fruitful collaboration with him.

Second, I would like to thank the committee members of my thesis defense, Prof. Dionne Aleman and Prof. Hani Naguib, for their precious time and helpful comments. I am truly appreciative.

I would also like to thank the members of the Financial Optimization and Risk Management Lab, Alexia Yeo-Giangaspero and Giorgio Costa, for the countless engaging discussions. Graduate school can be a lonely place; it was nice to know that I was not alone in this journey.

A massive thank you goes to my friends: Ali Elfeky, Andrew Effat, Nader Haddad, and Adham Rizk. Thank you for including me in your lives, and helping me get through the busiest and most demanding time of mine. Thank you for always being there. I could not have done this without you.

Finally, I would like to dedicate this thesis to my parents, Jailan Shindy and Tarek Anis. I will forever be in your debt for all the sacrifices that both of you made for me. This is as much your accomplishment as it is mine; we did it- together. Thank you for your endless support and love, for believing in me when I didn’t, for pushing me to be my best. I strive every day to have a fraction of your drive, character, integrity, and passion. I can never thank you enough for all that you’ve done. I love you.
“When my information changes, I alter my conclusions.
What do you do, Sir?”

- John Maynard Keynes
# Contents

## Acknowledgements

iii

## List of Figures

vii

## List of Tables

ix

## Abbreviations

x

1 Introduction 1

1.1 Background ................................................. 1
  1.1.1 Market Microstructure .............................. 2
  1.1.2 Optimal Execution ...................................... 2

1.2 Motivation ............................................... 4

1.3 Contributions ........................................... 8

1.4 Outline ...................................................... 8

2 Problem Formulation 10

3 General Framework 11

3.1 Risk Measure ............................................. 11

3.2 Stochastic Programming .................................. 13

3.3 Static Model ............................................. 18

3.4 Shrinking Horizon ....................................... 20

4 Case Study 22

4.1 Market Dynamics ......................................... 22

4.2 Market Impact Functions ................................. 23

5 Computational Experiments 25

5.1 Data ...................................................... 26

5.2 Model Parameter Estimation ................................ 26
  5.2.1 Volatility ............................................. 26
  5.2.2 Volume ................................................. 28

5.3 Simulation of Execution .................................. 32
  5.3.1 Experimental Setup .................................... 32
  5.3.2 Problem Parameter Selection ......................... 33
## List of Figures

1.1 Limit order book ......................................................... 3

2.1 Trading trajectory & schedule over time ............................... 10

3.1 Scenario tree vs. scenario paths for 4 time periods (Black dots represent
decisions) ................................................................. 17

3.2 Shrinking horizon .......................................................... 21

5.1 Estimated intraday volatility on for “KO” on 06-01-2017 ................. 29

5.2 Input for volume estimation ............................................... 29

5.3 Volume estimation error analysis ......................................... 30

5.4 Estimated intraday volume on for “KO” on 06-01-2017 ................. 32

5.5 Performance relative to benchmark vs. volatility The x-axis represents the
average of the benchmark capture subtracted from the strategies’ captures.
The y-axis is the standard deviation of the strategies’ captures compared
to the benchmark’s .......................................................... 37

5.6 Improvement over benchmark captures The upper and lower boundaries
of each box represent the 75th and 25th percentiles respectively, and the
red horizontal line inside the box depicts the median. The whiskers repre-
sent the minimum and maximum values within 1.5 times the interquartile
range, whereas plus signs represent outliers, i.e., any datum more than
1.5 times the interquartile range away from the boxes. .................. 38

5.7 Improvement over benchmark captures by market conditions The upper
and lower boundaries of each box represent the 75th and 25th percentiles
respectively, and the red horizontal line inside the box depicts the median.
The whiskers represent the minimum and maximum values within 1.5
times the interquartile range, whereas plus signs represent outliers, i.e.,
any datum more than 1.5 times the interquartile range away from the boxes. 39

5.8 Performance relative to benchmark vs. volatility by market conditions
The x-axis represents the average of the benchmark capture subtracted
from the strategies’ captures. The y-axis is the standard deviation of the
strategies’ captures compared to the benchmark’s ......................... 40

5.9 Price & execution during 06-01-2017 Dotted lines represent periods of
increased trading .......................................................... 41

5.10 Price movement during first half hour of 06-01-2017 Dotted lines represent
periods of increased trading ................................................... 42

5.11 Price & execution during 06-07-2017 Dotted lines represent periods of
increased trading .......................................................... 43

5.12 Price & execution during positive market conditions Dotted lines represent
periods of increased trading ................................................... 44
5.13 Price & execution during negative market conditions Dotted lines represent periods of increased trading

5.14 Liquidation on 06-15-2017 from 12:00 to 13:40 Dotted lines represent periods of increased trading

5.15 Price & execution during extremely positive market conditions Dotted lines represent periods of increased trading
List of Tables

3.1 Information-decision dynamics ............................................. 15
5.1 Day of Week effect error analysis ......................................... 31
5.2 Improvements in volume prediction ....................................... 31
5.3 Improvement over benchmark aggregate results ..................... 36
5.4 Improvement over benchmark aggregate results by market condition (+):
  positive market conditions (−): negative market conditions .......... 39
## Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AIM</td>
<td>Agressive-in-the-money</td>
</tr>
<tr>
<td>ARMA</td>
<td>Autoregressive Moving Average</td>
</tr>
<tr>
<td>CVaR</td>
<td>Conditional Value at Risk</td>
</tr>
<tr>
<td>DoW</td>
<td>Day of Week</td>
</tr>
<tr>
<td>GARCH</td>
<td>Generalized Autoregressive Conditional Heteroskedasticity</td>
</tr>
<tr>
<td>IS</td>
<td>Implementation Shortfall</td>
</tr>
<tr>
<td>LPM</td>
<td>Lower Partial Moments</td>
</tr>
<tr>
<td>LOB</td>
<td>Limit Order Book</td>
</tr>
<tr>
<td>MAPE</td>
<td>Mean Absolute Percent Error</td>
</tr>
<tr>
<td>MGARCH</td>
<td>Multiplicative Generalized Autoregressive Conditional Heteroskedasticity</td>
</tr>
<tr>
<td>NYSE</td>
<td>New York Stock Exchange</td>
</tr>
<tr>
<td>PIM</td>
<td>Passive-in-the-money</td>
</tr>
<tr>
<td>SP</td>
<td>Stochastic Programming</td>
</tr>
<tr>
<td>TAQ</td>
<td>Trade and Quotes</td>
</tr>
<tr>
<td>TWAP</td>
<td>Time-Weighted Average Price</td>
</tr>
<tr>
<td>VWAP</td>
<td>Volume-Weighted Average Price</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

1.1 Background

An investment process typically involves three major phases: making an investment decision, implementing the associated trades, and finally evaluating its performance. For years, financial research, both academic and practical, has focused on the first phase. The first published works on systematic investing appeared in the aftermath of the Great Depression and lead to a shift in the field of investment away from speculation. Seminal works, such as Schabacker (1930)'s “Stock Market Theory and Practice” as well as Graham and Dodd (1934)'s “Security Analysis”, for example, introduced the public to technical and fundamental stock analysis, respectively. Harry Markowitz (1952)'s Modern Portfolio Theory was one of the pioneering works to adapt formal mathematical concepts to finance, demonstrating the inherent tension between risk and return, and leading to the birth of the field of quantitative finance. Further extensions to portfolio theory occurred in the 1960s in the form of the Nobel-winning development of the Capital Asset Pricing Model by Sharpe (1964), Lintner (1965), and Mossin (1966). These efforts were intended to further investors’ understanding of the market or improve their investment decision making.

Although option contracts have existed at least since the second half of the 16th century (Poitras, 2009), it was not until the establishment of the Chicago Board Options Exchange in the 1970s, and the standardization of said option contracts that interest in their use as well as academic research into their behaviour increased. The paradigm-shifting publication of the Black-Scholes formula by Black and Scholes (1973) “allowed [for] the explosion of the options market as we know it today” (Cesa, 2017). It is worth noting that the primary use of options contracts, and financial derivatives in general, is to manage risk and falls within the first stage of the investment process.
Whereas a lot of attention and research has gone towards the search of superior investment strategies and risk management techniques, i.e., the first phase of the investment process, a lot less interest has been shown towards the implementation/execution of the associated trades. However, starting in the 1980s, reports, such as the work of Treynor (1981), Loeb (1983), and Perold (1988) amongst others, started to record the amount of lost return during the implementation of an investment idea due to transaction costs, leading to the creation of the field of “efficient implementation— the process known as algorithmic trading” (Kissell, 2013).

1.1.1 Market Microstructure

The location in the market where buy and sell orders meet, and the forces of supply and demand set an asset’s price is the limit order book (LOB) (See Figure 1.1). It records all interested buyers or sellers of a financial instrument. The orders on a LOB list the price at which investors are willing to sell (or buy) a share, as well as the number of shares that are available (or sought) at that price. These passive orders are called limit orders. They are listed first by price priority, then, for orders at the same price level, by time priority. The lowest (highest) price that investors are willing to sell (buy) a certain asset for is called the best-ask (best-bid) price. The difference between the best-ask and the best-bid is called the bid-ask spread. It is always positive since the best-ask is always higher than the best-bid; otherwise there would be a pair of buyers and sellers that is in agreement and a transaction/trade would occur immediately.

Whereas a limit order specifies the exact price at which a trade is to be executed, a market order is an instruction to execute a trade of a given quantity at the best price possible. Thus, a buy (sell) market order would execute at the best-ask (best-bid) price. If a market order has a demand higher than the number of units available at the best price level, then the order will move onto the next price level in order to fulfill the remaining shares on the order. Thus, whereas market orders are guaranteed to be executed, the effective price at which the total trade occurs is unknown. Limit orders, conversely, have a specified price yet no guarantees that they will indeed execute. Due to how the two types of orders interact with one another, limit orders can be thought of as liquidity providers and market orders as liquidity takers.

1.1.2 Optimal Execution

Once the investment decision is made, the associated trades have to be carried out. This thesis is not concerned with how the investor arrived at the investment decision. Rather, the role of a buy-side broker/trader who is simply given an asset’s name and an amount
The reason behind the investor’s order submission is not explored. It could be to rebalance a portfolio or a stand alone investment decision. The broker accepts the parent order and is expected to perform all necessary trades in the market during a timeframe and based on an execution benchmark that are agreed upon with the investor.

One of the most important problems faced by stock traders is how to execute large orders of security shares. Unlike for small trades, the significant impact that these orders have on the market must be taken into consideration. There are two basic strategies that demonstrate what is commonly known as the “trader’s dilemma”. First, a trader can execute the whole order instantaneously. This strategy will have a high market impact and incur a high trading cost. Due to the order’s large magnitude, it will keep executing according to the prices and quantities of the pre-existing limit orders and eat away at the LOB. Second, the trader can divide the parent order into smaller, equal-sized orders and execute each suborder consecutively. Although this strategy leads to smaller market impact, it allows for possible adverse price movements to lead to losses since each small market order executes at the current best price. In essence, traders liquidating too fast incur high execution costs, while liquidating too slowly exposes them to possible adverse price fluctuations, which may effectively lead to liquidation at lower-than-expected prices.

The problem of optimal execution is concerned with optimally splitting the executions into smaller suborders to be traded in such a way to balance the tradeoff.

\footnote{Note that if the interest instead was in buying, only the appropriate signs would need to change.}
between execution costs, market impact and price risk. This set of smaller suborders is known as the trading schedule. The study of optimal execution started with Bertsimas and Lo (1998), who focused solely on minimizing the expected execution costs. The seminal work of Almgren and Chriss (2001) extends Bertsimas and Lo (1998)’s framework by adding a market impact term. Similar to the objective of the mean-variance portfolio optimization presented by Markowitz (1952), their framework balances the market impact cost, which leads toward slow trading so as to reduce the expected value of execution cost, against market volatility, which attempts to trade more rapidly in order to reduce the execution cost’s variance. The two strategies used in Section 1.1.2 to illustrate the “trader’s dilemma” represent both extremes involved in the tradeoff at the heart of Almgren and Chriss (2001)’s execution framework. It is easy to see the strategy that only minimizes the market impact is the aforementioned constant trading strategy. It uses the slowest possible trading rate that still guarantees the liquidation of all units in the parent order. On the other hand, the strategy that solely minimizes the execution cost’s variance is the other strategy which executes the large parent order all at once. It has the smallest variance with regard to the execution cost since it only submits one order. The main drawback of Bertsimas and Lo (1998) and Almgren and Chriss (2001) is the inability of their proposed trading strategies to respond dynamically to realized or changed market conditions. Both propose static strategies that are only computed once at the outset of trading. Yet, executing a prescribed sequence of trades based only on the information provided prior is inherently suboptimal since it ignores the new information coming into the market as the schedule progresses.

Extensions of Almgren and Chriss (2001) and more recent works have attempted to remedy this flaw, usually by way of dynamic programming (see Almgren and Lorenz (2006), Forsyth (2011), Almgren (2012), Guéant et al. (2012), Kato (2014), Frei and Westray (2015), Guéant (2015), Schöneborn (2016) amongst others), which tend to impose either a set of assumptions on the problem in order to make it tractable (Frei and Westray (2015), for example, assume the intraday relative volume curve can be modelled by a gamma bridge) or stylize the problem such that it becomes unrealistic (Almgren and Lorenz (2006), for example, assume that “current [market] conditions will last until the end of trading”).

1.2 Motivation

In this thesis a more data-driven approach is proposed. Data-driven approaches, in contrast to classical/stylized models, such as dynamic programming, do not necessarily call for the intermediate step of estimating a statistical model of or imposing assumptions
on the shape of the problem’s parameters’ behaviour\textsuperscript{2}. As a result, such approaches can have a better out-of-sample performance compared to the classical/stylized models, which are very sensitive to model estimation errors, and may fail to replicate their in-sample performance or maintain their desired characteristics. A classic example in the field of finance is the improvement of the classic mean-variance portfolio optimization through Michaud Resampling, which manages to reduce the model’s out of sample estimation error (see Michaud and Michaud (2007)).

A common method used to deal with the inherent uncertainty in financial problems, especially ones that are multiperiod in nature, is stochastic programming (SP). It has been used in a variety of applications: from the classical asset-liability management problem (Consigli and Dempster (1998), Zenios and Ziemba (2007)) and large enterprise-wide risk management problems (Zenios and Ziemba (2007)), to multi-stage portfolio management problems (Dantzig and Infanger (1993), Calafiore (2015), Köksalan and Şakar (2016)). Butenko et al. (2005) and Krokhmal and Uryasev (2007) seem to be the only two works to have used SP within the context of the optimal execution problem. Butenko et al. (2005) create a multinomial scenario-tree to represent possible future returns. They attempt to maximize to total return first with no constraint on risk and then with the trading strategy subject to a Conditional Value at Risk (CVaR) constraint. They solve the problem first using Linear Programming methods and then using dynamic programming methods. The proposed framework provides path-dependent strategies that trade some fraction of the total security amount depending upon where the security’s price is at any given moment. The main drawback with the work is the amount of nodes required in the multinomial trees in order to create an accurate representation of possible future scenarios.

Similarly, Krokhmal and Uryasev (2007) develop a dynamic optimal transaction implementation strategy based on a collection of sample paths representing possible future realizations of state variable processes. Additionally, they apply a heuristic sample-path grouping method to direct their future decisions. Each group contains sample paths with similar numerical attributes at each time step. Thus, whenever the actual price path realization falls within a group at each time step, a different trade is executed. The aforementioned heuristic helps mitigate the exponential growth of the action space that occurs in multi-stage decision problems in order to model their approximately conditional nature while keeping the solution truly multi-period (similar to the approach of Butenko et al. (2005)). In this thesis, a \textit{truly multi-period} solution is defined as one that is generated in the first time period and is not modified in future time steps. Such a solution will account for different outcomes in the future realizations and self-adjust

\textsuperscript{2}Note, while the data can be (and usually is) produced using a simple parametric statistical model or have assumptions imposed on it, this model-based approximation is not necessary.
accordingly, i.e., the problem is not resolved as future realizations occur. Thus, according to the aforementioned definition, Butenko et al. (2005) is also truly multi-period as it is only calculated at the outset of trading. Similar to Butenko et al. (2005), Krokhmal and Uryasev (2007) also add a CVaR constraint on the problem to hedge against extreme tail risk.

While the SP framework in general is truly multi-period in nature, it is static in terms of the information set used in its computations. The results of any proposed SP is a set of action-state pairs that, once the optimization is executed at the beginning of a problem’s time horizon, remains unchanged. The future decisions are then mapped onto the set, and depending on where the future realization lands within the generated state space its corresponding action is taken. In SP, once scenarios are projected, the paths are kept static and are not amenable to updates. Although the method’s rational is that scenarios help induce robustness, the model itself is static. The new information that is observed, or enters the market throughout the liquidation period in the problem, is not taken into consideration beyond the stochastic process’ position within the action space. In the optimal liquidation problem, a regular SP would only take the asset’s current price into consideration when making a decision. If any changes in the underlying market microstructure occur, they are not taken into consideration if they were not anticipated in the initial scenario generation.

Additionally, while a regular SP with scenario trees is tractable and appropriate when working on a daily timescale, similar to the one used in both Butenko et al. (2005) and Krokhmal and Uryasev (2007)’s computational experiments, such an approach would not be as suitable for problems with shorter timescales. For example, for a problem with minutely decision periods to be able to capture all possible realizations of the underlying price process, including any regime changes in the asset price’s volatility or volume, the number of generated scenarios has to be extremely large. Combining the large number of periods with the large state-action pairs at each time period, the resulting problem would be rendered computationally intractable due to the explosion in the problem’s dimensionality. The solution is to either reduce the number of groups, leading to generic, suboptimal results, or to use a small number of decision periods, which would not be desirable in the current high-frequency market.

In contrast, a novel quasi-multi-period model for optimal position liquidation in the presence of market impact is proposed. Instead of the common stylized approach of modelling the problem as a dynamic program with static trading rates, the problem is framed as a SP, in a similar vein to Krokhmal and Uryasev (2007) which uses a sample paths to represent possible future realizations of state variables, and is solved using sample average optimization as proposed by Kleywegt et al. (2002). In order to
combat the explosion in dimensionality typical of SP problems and take new incoming information into account, a Shrinking Horizon framework, similar to the one used by Calafiore (2015) in a portfolio optimization setting alongside SP, and by Busseti and Boyd (2015), who use it in tandem with dynamic programming to tackle a variation of the optimal execution problem, is implemented\(^3\). The Shrinking Horizon framework allows the problem to take advantage of new incoming information while maintaining standard non-anticipativity constraints. It also significantly reduces the problem’s action space rendering it more tractable and easily solvable. Thus, the trader is able to dynamically update their trading decisions based on the most current market conditions with ease and incorporate new information in their decision making process.

Additionally, although Butenko et al. (2005) and Krokhmal and Uryasev (2007)’s use of a downside risk measure in the form of CVaR in the optimal liquidation problem is more appropriate than a symmetric risk measure (such as variance used in Almgren and Chriss (2001)’s seminal work) as it better encapsulates traders’ risk aversion according to prospect theory (Kahneman and Tversky (1979)), a more general and modular downside risk measure, lower partial moments (LPM), is used. The reason is threefold. First, whereas CVaR hedges against extreme tail risk, the first LPM aims to minimize the losses against any benchmark set by the user instead of losses at a specific percentile. Note, if the benchmark is set equal to a tail percentile, then the first LPM can be shown to be equivalent to CVaR. Second, since the proposed risk measure can tackle other aspects of the objective function such as its semi-variance, or the skewness and kurtosis of the function’s outcomes below a benchmark, it can be considered more modular and offers investors more control. Thus, using LPM, one can minimize the amount of losses expressed as their central tendency below the benchmark, the dispersion of said losses below the benchmark, as well as control the shape of their distribution. Third, the CVaR constraint implemented by Butenko et al. (2005) and Krokhmal and Uryasev (2007) might not be suitable in the current problem and in fact leads to inconsistencies within their problem formulation. Specifically, the CVaR constraint is in direct disagreement with their formulation of the price dynamics as following an Arithmetic Brownian Motion, a common modelling assumption of intra-day price dynamics. The formulation is only allowed if it assumed that an asset’s price cannot have extremely large negative movements within a single time horizon; otherwise, the price dynamic would permit negative asset prices. As a tail risk measure, it is precisely these large downward deviations that CVaR is supposed to hedge the strategy against.

\(^3\)It is stressed that whereas Busseti and Boyd (2015) apply the Shrinking Horizon framework in an attempt to track a Volume-Weighted Average Price (VWAP) strategy, the investor in this thesis is not tied to any specific strategy. Here, no predefined strategy is being tracked but rather the liquidation trajectory that the model recommends is executed.
Chapter 1

1.3 Contributions

The work presented in this thesis is an alternative to Butenko et al. (2005) and Krokhmal and Uryasev (2007) that addresses their works’ aforementioned drawbacks. Its main contributions to the existing literature are twofold. First, is the development of a quasi-multi-period model for optimal position liquidation that uses SP to deal with the problem’s inherent uncertainty, and a Shrinking Horizon framework to avoid the dimensionality curse and incorporate incoming market information. The main model is data-driven, modular, reactive, and does not assume any structure on its parameters. Second, this work is the first to implement the LPM risk measure while tackling the problem of optimal execution.

The following contributions have been made to the literature:

Publications


Presentations


1.4 Outline

The thesis is structured as follows: Chapter 2 formally defines the problem of optimal execution. In Chapter 3, the general model is introduced as well as the Shrinking Horizon framework. Chapter 4 presents the price dynamics and all relevant variables for
the sample paths used in the numerical experiments in Chapter 5 which are based on real NYSE market data. Chapter 6 concludes the thesis with a discussion of the results.
Chapter 2

Problem Formulation

Suppose an investor holds a block of \( X \in \mathbb{R}_+ \) units of a security to be liquidated completely before a time \( T \). Note, in reality the holdings \( x_t \) and trades \( u_t \) are integer, not real numbers. This approximation is acceptable since the typical number of shares held is much greater than 1 making the rounding error negligible. Again, the reason behind the decision to sell is not examined; the only concern is that a large block of assets is to be sold. The time horizon \( T \), during which the whole position is to be liquidated, is divided into equidistant intervals corresponding to the trading times \( t = 0, ..., T \). Define a trading trajectory \( \{x_0, x_1, ..., x_T\} \) to be a list where \( x_t \) is the number of units of the security to be held at time \( t \). Thus, the investor’s initial holding is \( x_0 = X \) and the complete liquidation constraint translates to the terminal condition that \( x_T = 0 \).

Further, let the trading schedule \( \{u_0, u_1, ..., u_{T-1}\} \) be a list where \( u_t \) is the number of units of the security to be sold in the period between times \( t \) and \( t + 1 \) (Figure 2.1). It is clear that \( u_t = x_t - x_{t+1} \). Thus, \( x_t \) and \( u_t \) are related as follows:

\[
x_t = X - \sum_{j=0}^{t-1} u_j = \sum_{j=t}^{T-1} u_j \quad \text{for} \quad t = 0, ..., T \tag{2.1}
\]

![Figure 2.1: Trading trajectory & schedule over time](image)

\(^1\)Note that this definition of trading schedule and its relation with the trading trajectory differs slightly from the one found in Almgren and Chriss (2001). Here, the trades are defined as occurring between times \( t \) and \( t + 1 \) as opposed to between times \( t - 1 \) and \( t \). This change is meant to reflect the nature of when the trade decision is taken and the effects of said change is reflected in Equation (2.1).
Chapter 3

General Framework

3.1 Risk Measure

The first trading decision an investor has to make is selecting an implementation benchmark. Based on this macro-level trading specification, the investor will choose an execution strategy which will eventually be analyzed against said benchmark once trading has concluded. Define a capture \( C(x, u) \in \mathbb{R}^+ \) as the full trading revenue upon completion of all trades \( u_t \) and let \( \Psi \) denote a given benchmark against which the capture \( C \) is evaluated. Note that \( \Psi \) is not restricted to a specific benchmark. The investor is free to choose any. Some common choices are the arrival price similar to Perold (1988)’s Implementation Shortfall (IS), the Time-Weighted Average Price (TWAP) which is the arithmetic average of a day’s prices, or the VWAP which is an average price in proportion to the intraday volume curve.

Examining the capture function’s two possible outcomes, over-performance \( (C(x, u) \geq \Psi) \) and underperformance \( (C(x, u) < \Psi) \), it becomes apparent that if a symmetric risk measure is being minimized then both outcomes would be penalized equally. Variance, for example, is a symmetric measure of (squared) distance of a random variable from its expected value. Thus, were one to minimize the variance of an assets return, a random variable itself, one would be minimizing positive deviations from the mean, i.e., gains, as well as negative deviations from the mean, i.e., losses, equally.

The most cited paper ever to appear in Econometrica, the most prestigious academic journal in the field of economics, is titled “Prospect Theory: An Analysis of Decision under Risk” and was authored by the two psychologists Kahneman and Tversky (1979). In it, the two describe the decision making process under risk. Some of the work’s most important contributions are:
1. Value is assigned to gains and losses rather than to final assets.

2. The value function used in decision making is defined on deviations from a reference point.

3. The value function is concave for gains, convex for losses, and is generally steeper for losses than for gains implying loss aversion.

It is clear that whereas variance does satisfy the second finding, where the reference point is the expected value of the random variable, it does not satisfy the third finding as it treats both losses and gains equally.

In order to accommodate the fact that investors are not averse to deviations from the expected target if these deviations exceed it, according to Prospect Theory, a downside risk measure is implemented. This is one of the main contributions of this framework, which again differs in this point from most literature regarding optimal liquidation (see Almgren and Chriss (2001), Lehalle (2009), Labadie and Lehalle (2010), Kissell (2013)'s $I^*$ Model, and Busseti and Boyd (2015)), which uses variance. The use of LPM is proposed, which, similar to Calafiore (2015), is defined as

$$\text{LPM}_\nu(\Psi, C) = \max (0, \Psi - C)$$

where the “moments” in the measure’s name refer to the central moments of the function. The superscript $\nu$ signifies the moment (where $\nu = 1$ is the mean, $\nu = 2$ is the variance, $\nu = 3$ is the skewness, etc.) Thus, the measure is evaluating the lower moment of the capture $C(x, u)$ compared to the benchmark, $\Psi$.

Analyzing Equation (3.1) in terms of the two aforementioned outcomes, the max function only considers a capture $C$ if it underperforms the benchmark $\Psi$ when the inner expression, $\Psi - C(x, u)$, is positive. Thus, the first lower partial moment, $\text{LPM}_1(\Psi, C(x, u))$, is a cost function measuring the empirical average of the capture of the trading trajectory $C(u, x)$ falling below the benchmark $\Psi$, or the average losses. Similarly, the second lower partial moment, $\text{LPM}_2(\Psi, C(x, u))$, measures the average of the squares of the same deviations, or the semi-variance of the losses compared to the benchmark. Also note, that by comparing the capture to a benchmark, the problem is now framed in terms of gains and losses instead of the final total trading revenue, thus conforming with the first finding of Prospect Theory stated above.

It is noteworthy that, although the function described in Equation (3.1) conforms with the first two listed findings of Kahneman and Tversky (1979) fully, it does not satisfy the third finding completely. As mentioned, Equation (3.1) only evaluates the capture $C(x, u)$ if it underperforms the benchmark $\Psi$. Thus, the function completely
ignores gains. With regards to the third finding, Equation (3.1) is steeper for losses than for gains by assigning positive values to losses and a zero to gains. Yet, since the evaluation of the gains results in a line at zero, the value function of the LPM risk measure is, ipso facto, convex, which is in conflict with investors’ risk aversion as the risk measure is not concave for gains as required.

There are two justifications for using LPM despite its violation of the concavity requirement for gains. First, the function still partially observes investors’ risk aversion by focusing on the losses more than the gains. Indeed, by assigning a value of zero to gains, it focuses solely on losses, which is similar to most other financial downside risk measures. Second, LPM is more modular than other common financial downside risk measures. As mentioned earlier, setting the benchmark $\Psi$ of a first LPM equal to a tail percentile (of the losses), then both the resulting first LPM and the CVaR risk measures can be shown to be equivalent. But, the LPM function can also evaluate the capture’s semi-variance below the benchmark as well as its skewness or kurtosis. Thus, it can be used in a similar fashion to other risk measures but also affords the user more options and allows for more control over the capture’s performance compared to the benchmark. The use of the LPM risk measure represents a clear improvement over symmetric risk measures such as variance since it satisfies the findings of Kahneman and Tversky (1979) with regards to investors’ decision making more aptly, and offers users more control, through its additional modularity, than other downside risk measures used in the field such as CVaR.

Finally, it is important to note that the function described in Equation (3.1) is convex regardless of the value that the superscript $\nu$ takes on. Thus, placing Equation (3.1) into an optimization problem’s objective function and combining it with similarly convex constraints (preferably either linear or quadratic constraints) would result in convex mathematical programs which are efficiently solved using off-the-shelf solvers.

### 3.2 Stochastic Programming

*Stochastic programming* is a method for modeling optimization problems that involve uncertain parameters. Whereas the parameters of deterministic optimization problems are known with certainty, real world problems almost always include parameters which are unknown when a decision is being made. In the current problem, for example, the parameters governing a financial instrument’s price process such as its volatility, drift, or the participating volume are unknown. As a result, an investor at 10:00am does not know with certainty what the price of an asset will be at 10:30am. Yet, these uncertain parameters can be assumed to lie in some given set of possible values, and one might seek
to find a solution that is feasible for these parameter possibilities, and that optimizes a given objective function.

As mentioned earlier, SP has long been used to model multi-period problems. Specifically, the technique has been used to solve sequential decision-making problems in the presence of uncertainty. Given an objective function \( f(x, \zeta) \), where \( \zeta \) is a random vector with a finite number of possible realizations representing the uncertainty in the system, one would seek to find the decisions, \( x = [x_0, x_1, ..., x_T] \), over the problem’s horizon \( T \) that would optimize said function. A *scenario* is thus defined as the an outcome of the random process \( \zeta = [\zeta_1, \zeta_2, ..., \zeta_T] \) where each \( \zeta_t \) is a random variable itself. It is assumed that although the values of \( \zeta \) are unknown, its probability distribution is known or can be estimated. Another implicit assumption is that although the function \( f \) cannot be computed exactly, it can be reasonably estimated.

SP attempts to optimize the problem in expectation. The reasoning is that if the underlying stochastic process repeats itself enough times, then, by the Law of Large Numbers, for a given solution \( x \), the average of the objective function value over the problem’s horizon \( T \) will converge to its true expectation \( \mathbb{E}[f(x, \zeta)] \). Indeed, in that case, \( x \) will be optimal on average.

The standard approach to model SP problems is by using *scenario trees* which discretize the random process \( \zeta \). Starting with an initial/root node, representing the first decision stage and the initial absence of observations, \( k \) possible realizations of \( \zeta_1 \) are generated. The root node is connected to each of the children nodes associated with the \( k \) possible realizations. Next, conditional on every possible realization of \( \zeta_1 \), for each child node at the second decision stage, several (usually the same number \( k \)) realizations of \( \zeta_2 \) are generated, representing possible outcomes of the second decision stage. Then, each node of stage 2 is connected to children nodes associated with the third decision stage, one for each possible outcome of \( \zeta_2 \). This process is repeated until the terminal stage is reached. The realizations associated with the nodes on unique paths from the root node to each terminal node define the scenarios representing possible outcomes of the underlying random process. Probabilities are assigned to the arcs/branches connecting the nodes which represent the conditional probability of moving from one (parent) node to the next (child) node. Assuming that for each \( \zeta_t \), for \( t = 1, ..., T \), \( k \) possible realizations are generated, then the scenario tree would have a total of \( N = k_1 \times k_2 \times ... \times k_T = k^T \) scenarios. Thus, the probability of scenario \( i \), for \( i = 1, ..., N \), of occurring is

\[
\mathbb{P}(\zeta^{(i)}) = \mathbb{P}(\zeta^{(i)}_1) \times \mathbb{P}(\zeta^{(i)}_2 | \zeta^{(i)}_1) \times ... \times \mathbb{P}(\zeta^{(i)}_T | \zeta^{(i)}_1, ..., \zeta^{(i)}_{T-1})
\]

(3.2)
Another consideration that has to be made when using SP is how decisions should be taken and when the realizations of the random processes are revealed. A decision \( x_t \) will depend on the decision immediately preceding it, \( x_{t-1} \). For example, if a trader at time \( t \) has 1000 shares left to liquidate and sells 100 shares in the next decision period, then the decision made \( t + 1 \), irrespective of what the decision itself is, should have the investor’s initial position for the period at 900 shares. Additionally, a decision \( x_t \) can only take the realizations up to \( \zeta_{t-1} \) into consideration when determining what decision to take. For example, an investor at time \( t \) only knows how the asset’s price has evolved up to exactly that moment. Any decision made at the time can only take the realizations of \( \zeta_t \) to \( \zeta_{t-1} \) into consideration. The investor cannot make a decision at time \( t \) based on the actual realizations of any of \( \zeta_t \) to \( \zeta_T \) because that information is yet to be revealed. This condition is called the non-anticipativity constraint since it states that decisions should not depend on anticipated outcomes. Please refer to Table 3.1 for a representation of the dynamics of decisions with respect to information.

Thus, given a feasible set \( \mathcal{X}_t(\zeta^{(i)}) \), for decision period \( t \) in scenario \( i = 1, ..., N \), a SP problem would, for each possible outcome \( \zeta^{(i)} = [\zeta_1^{(i)}, ..., \zeta_T^{(i)}] \) of \( \zeta \) which occurs according to the probability in Equation (3.2), have associated decision variables \( x^{(i)} = [x_0^{(i)}, ..., x_{T-1}^{(i)}] \), representing a decision for each possible realization of \( \zeta \). The mathematical formulation with a total of \( N \) scenarios is:

\[
\min_{x^{(i)} \forall i} \mathbb{E} [f(x, \zeta)] \approx \sum_{i=1}^{N} \mathbb{P}(\zeta^{(i)}) \times f(x^{(i)}, \zeta^{(i)})
\]

s.t. \( x_{t-1}^{(i)} \in \mathcal{X}_t(\zeta^{(i)}) \quad \forall t = 1, ..., T, \forall i \)

\( x_1^{(i)} = x_1^{(j)} \quad \forall i, j \)

\( x_{t-1}^{(i)} = x_{t-1}^{(j)} \quad \forall t = 1, ..., T, \forall i, j \) when \( \zeta_t^{(i)} = \zeta_t^{(j)} \)

The last two constraints are added to enforce the aforementioned non-anticipativity dynamics.

---

1 Actual realizations of \( \zeta \) are identified with a ‘*’ in their superscript.
While SP provides a conceptually sound framework for truly multi-period decision problems within an analytical setting, from a computational standpoint it often becomes impervious to exact and efficient numerical solution (Kleywegt et al., 2002). The key difficulty in the traditional stochastic programming formulations stems from the sheer number of time-state pairs as well as the added non-anticipativity constraints which render the problem computationally intractable. To model the conditional nature of the problem in some tractable way, the decision space is discretized and represented using scenario trees and lattices. These trees may grow exponentially in order for the discretization to be accurate and representative which may lead to considerable computational difficulties. This is known as the “curse of dimensionality”.

If branching is kept low, the resulting discretized decision space will be a poor approximation of the uncertain future. Thus, the trade-off is always between a small number of branches per node in a tree, which cannot guaranteed to be a reliable representation of reality, or an increasing of the branches, which results in exponential growth of the overall number of scenarios in the tree. Additionally, as mentioned earlier, in problems with a high degree of granularity, such as optimal liquidation in high frequency markets, a scenario tree would become too unwieldy. A multi-stage tree depicting a minute-by-minute decision for the duration of a trading day (390 periods) would be computationally intractable. If one were to assume that a financial instrument can only go up or down at each decision period, then the resulting minutely binomial tree would have $2^k - 1 = 2^{390} - 1 \approx 2.5 \times 10^{117}$ decision variables. A common assumption is that asset returns are normally distributed which means that the previous binary branching, which is a very inaccurate representation of asset movements, is still intractable in an minutely decision-making setting. Note that an increasing of the branches by $b \geq 1$ branches would result in $\gg 2.5 \times 10^{117}$ decision variables.

A number of methods have been proposed to alleviate the “curse of dimensionality” that occurs when modelling SPs using scenario trees falling within three general areas of research:

- **Decomposition Algorithms**: These algorithms try to exploit the unique structure of a problem in order to reduce computational times. The vast majority of the work concerning the decomposition of multistage stochastic programs (MPS) deals with linear MPS, using variations of the classic L-shaped or Benders decomposition to solve them (Please refer to Birge and Louveaux (2011) [Chapter 6.1] for a survey of the example solution methods). Very few works deal with nonlinear MPS with Louveaux (1980) being the first to tackle quadratic MPS, Birge and Rosa (1996) applying variations of Benders decomposition to solve problems with
nonlinear constraints, and, more recently, Berkelaar et al. (2005) proposing a new decomposition algorithm for convex MPS based on interior point methods.

- **Scenario Generation**: These methods attempt to generate small scenario trees that nevertheless accurately represent the system’s underlying stochastic processes. This is usually achieved by minimizing a probabilistic distance between the generated tree and the underlying stochastic processes, similar to the work proposed by Høyland and Wallace (2001), and Pflug (2001). The probabilistic distance is minimized by solving non-linear optimization problems.

- **Scenario Reduction**: These techniques try to reduce the number of scenarios in a given scenario tree while retaining some of its essential features. In general, the method tries to find the subset of the scenarios with a given probability measure that is closest to the original scenario tree’s same probability measure. These scenario reduction techniques were first proposed by Dupačová et al. (2003), and developed further by Heitsch and Römisch (2003).

The problem with the aforementioned remedies to the scenario trees’ “curse of dimensionality” is that they would only provide marginal improvement in computational tractability for a granular MPS. The non-linear programs resulting from the scenario generation methods become extremely difficult to solve for problems with plenty of time periods (Latorre et al. (2007)), whereas the scenario reduction methods would not be able to reduce the number of scenarios enough to make it tractable while preserving the problem’s representativeness.

Another common alternative is to model the uncertain future as a collection of sample paths instead of a scenario tree. Each independent path would represent a possible future outcome. This approach overcomes the “curse of dimensionality” as it
increases linearly along with the increase of the number of nodes and the number of time steps, in contrast to scenario trees which grow exponentially in both cases. Please refer to Figure 3.1 which depicts the difference between a (binomial) scenario tree (Figure 3.1(a)) and a collection of ($N = 4$) sample paths (Figure 3.1(b)). The method known as sample average approximation (SAA) of the problem is used where it is assumed that the same decision is taken in all future realizations and optimize over all scenarios at once.

The SAA technique also optimizes in expectation. It tries to approximate the true optimal solution $z^* = \min \mathbb{E} [f(x, \zeta)]$ using its sample average

$$
\hat{z}_k^* = \min_x \frac{1}{N} \sum_{i=1}^{N} \left[ f(x, \zeta^{(i)}) \right]
$$

where $\zeta^{(i)}$'s are i.i.d. and $k = 1, \ldots, K$ are the different instances of the problem solved. Using a large number of scenarios $N$ and solving numerous instances of the problem $k$, the average of the sample average $\bar{z}^* = \frac{1}{K} \sum_{j=1}^{K} \hat{z}_k^*$ becomes an unbiased estimator of $z^*$ due to the Central Limit Theorem (for a detailed explanation on the SAA method as well as convergence results, refer to Shapiro and Philpott (2007)). As $N$ and $k$ increase to $\infty$, the averages of $\hat{z}_k^*$ should improve and converge more to their true expected solution. Note, this method is somewhat similar to the one used by Krokhmal and Uryasev (2007) which differs in two important ways: first, they use scenario path grouping to make the problem more similar to a scenario tree, and second, they only solve a single instance of the problem instead of $k$ instances.

### 3.3 Static Model

Setting $f(x, \zeta)$ equal to $\text{LPM}_\nu$ results in $z^* = \min \mathbb{E} [\max (0, \Psi - C(x, u))^\nu]$ as an objective function which can be unbiasedly estimated solving multiple instances of

$$
\hat{z}_k^* = \min_{x, u} \frac{1}{N} \sum_{i=1}^{N} \left[ \max \left( 0, \Psi - C(x, u)^{(i)})^\nu \right) \right]
$$

(3.3)
Combining Equation (3.3) with the constraints described in Chapter 2 Equation (2.1) results in the following full model:

$$\min_{x,u} \text{LPM}_\nu(\Psi, C) = \frac{1}{N} \sum_{i=1}^{N} \left( \max \left( 0, \Psi - C(x, u)^{(i)} \right) \right)^\nu$$

(A)

s.t. \( x_0 = X \)
\[
x_T = 0
\]
\[
\sum_{t=0}^{T-1} u_t = X
\]
\[
\forall t = 0, ..., T - 1
\]
\[
u_t = x_t - x_{t+1}
\]
\[
u_t \geq 0 \quad \forall t = 0, ..., T - 1
\]
\[
x_t \geq 0 \quad \forall t = 0, ..., T - 1
\]

It is easy to see that Problem (A) is a convex optimization problem as the objective function is convex regardless of the value of \( \nu \) and all constraints are affine.

**Proposition 3.1.** Problem (A) is equivalent to the following problem with the variables \( u \in \mathbb{R}^{T-1} \), \( x \in \mathbb{R}^{T} \), and \( y \in \mathbb{R}^{N} \):

$$\min_{y,x,u} \text{LPM}_\nu(\Psi, C) = \frac{1}{N} \sum_{i=1}^{N} y^{(i)}^\nu$$

(B)

s.t. \( x_0 = X \)
\[
x_T = 0
\]
\[
\sum_{t=0}^{T-1} u_t = X
\]
\[
\forall t = 0, ..., T - 1
\]
\[
u_t = x_t - x_{t+1}
\]
\[
u_t \geq 0 \quad \forall t = 0, ..., T - 1
\]
\[
x_t \geq 0 \quad \forall t = 0, ..., T - 1
\]
\[
y^{(i)} \geq \Psi - C^{(i)}(x, u) \quad \forall i = 1, ..., N
\]
\[
y^{(i)} \geq 0 \quad \forall i = 1, ..., N
\]

where selecting \( \nu = 1 \) or \( \nu = 2 \) and using a linear or quadratic capture \( C^{(i)} \) results in Problem (B) becoming either a linear, quadratic, or a second order cone program.

All aforementioned forms of the convex optimization problem (B) may be solved efficiently using off-the-shelf solvers. Note, neither \( x_t \) nor \( u_t \) have a superscript \( i \) indicating that, as previously mentioned, the stochastic programs in Problems (A) and (B) result in an identical trading trajectory/schedule for all \( i = 1, \ldots, N \) scenarios.
interest of readability, only Problem (B) and its solutions are discussed\(^2\). To make the static solution adaptive, we overlay it with the shrinking horizon framework described in the following section.

### 3.4 Shrinking Horizon

The solution to Problem (B) at time \( t = 0 \) results in the trade schedule \( \{u_0, \ldots, u_{T-1}\} \). If the problem is only computed at time \( t = 0 \) then the changing nature of the problem is not taken into consideration. Although at time \( t = 0 \) the whole future sequences of the price processes, price volatilities, and market volumes are uncertain, at each subsequent time period the actual outcomes of these parameters are realized/revealed to the investor. Also note, that only action \( u_0 \) is actually executed at time \( t = 0 \); all other computed actions may be discarded or adjusted to take the newly available information into consideration. Therefore, the subsequent action at time \( t = 1 \) should take the first realizations of the parameters into account, subject to the uncertain developments of the future periods. In general, at the decision stage \( t \), the past realizations over periods \( 0, \ldots, t - 1 \) have been observed, and hence are exactly known, while the future outcomes of periods \( t, \ldots, T - 1 \) are still uncertain. Thus, if we define the information set available at time \( t \) as \( I_t \equiv \{(P_0, V_0, \sigma_0, u_0), \ldots, (P_{t-1}, V_{t-1}, \sigma_{t-1}, u_{t-1})\} \) then clearly, we know that when we choose the value of trade \( u_t \) we can access, at most, the information contained in \( I_t \). Thus, uncertainty is reduced at each decision stage.

In general, a Shrinking Horizon framework (similar to Skaf et al. (2010), Busseti and Boyd (2015), and Calafiore (2015)) is applied. Starting at time \( t = 0 \), i.e., before the beginning of the first period, the optimization problem is solved over the \( T \) periods ahead, and the first optimal trade \( u_0 \) is determined and implemented. Then, the actual market outcomes over the first period are observed and the performance of the action is measured against them. Then, at \( t = 1 \), the optimization is solved again over a reduced horizon of \( T - 1 \) periods, obtaining and implementing the here-and-now decision \( u_0 \) (which now replaces \( u_1 \) in the original trading schedule), and evaluating its performance against the real outcome of the market over the second period. Therefore, at each time \( t \), an optimization problem over a horizon of \( T - t \) periods is solved and only decision \( u_0 \), which replaces the trading decision \( u_t \) in the original trading schedule, is implemented. Please refer to Figure 3.2 for a visual representation of the Shrinking Horizon framework.

Thus, one can think of the Shrinking Horizon framework as being similar to a two-stage SPg problem where, at decision stage \( t \), \( u_t \) is the first-stage decision variable.

\(^2\)Everything mentioned in the context of Problem (B) also applies to Problem (A).
and the remaining decision variables $u_{t+1},...,u_{T-1}$ are the second stage decision variables. The Shrinking Horizon framework, in essence, reoptimizes the problem at each decision stage $t$ and only implements the first-stage decision variable. The remaining decision variables are then readjusted at each subsequent decision stage by optimizing the problem again subject to the previously executed first-stage decisions as well as the new realizations that are added to the information set $I_t$.

The Shrinking Horizon framework results in an overall strategy that is not truly multiperiod but rather mimics one by constantly updating the most immediate decision variable to incorporate information conveyed by current market conditions. Thus, this computationally tractable formulation of the problem results in an overall trading schedule that is reactive to the market without imposing any specific restrictions or assumptions on the problem’s parameters. Again, the investor is free to model the capture $C^{(i)}$ which every way they choose. Similarly, any benchmark $\Psi$ may be chosen. Problem (B) set in a shrinking horizon framework results in a data-driven, risk-averse, adaptive, quasi-multiperiod trading schedule.
Chapter 4

Case Study

In this chapter, a possible price process is developed to be used in the subsequent chapter’s numerical experiments. The process used in the experiments is for illustration purposes and thus, is rather simple and does involve some assumptions. It is important to stress that the framework proposed in the previous chapter is not dependant on the following price process. Users may choose any price process they wish.

4.1 Market Dynamics

Suppose the price of the security $P_t$ for $t = 0, ..., T$ follows a stochastic process that evolves according to some exogenous factors, such as volatility and drift, and one endogenous factor, market impact. That is the price changes according to the exogenous factors independently from the investor’s individual trades.

On the other hand, the trades translate into information that is released into the market. Other market participants naturally adjust their positions as they detect new volume entering the market. This market impact is divided into two types. Temporary Impact is the discount sellers need to provide the market in order to attract additional buyers. As the name implies, this effect is short-lived, and liquidity returns the following period when a new equilibrium price is established. Permanent Impact represents the information content that causes market participants to adjust their prices due to a new perceived fair value. The rationale is that each one of the investor’s trades, assuming they are rational, represents a view on the security being traded. Thus, as market participants observe sell orders their outlook is that the security is overvalued and they will lower their bid prices.
For any asset, let its price evolve according to the discrete arithmetic random walk
\[ P_t = P_{t-1} + \sigma_t \xi_t - g(u_{t-1}) \quad \forall t = 0, ..., T \] (4.1)
with a known initial asset price \( P_0 \). Here \( \sigma_t \) represents the volatility of the asset at time \( t \), \( \xi_t \overset{i.i.d.}{\sim} \mathcal{N}(0, 1) \), where \( \mathcal{N} \) is the Gaussian distribution, representing the uncertainty in the system at time \( t \), and \( g(u_{t-1}) \) is the permanent impact due to the trade decision taken in the previous period and executed between times \( t-1 \) and \( t \). It is noted that the arithmetic random walk described in Equation (4.1) may theoretically result in negative prices. Indeed, in long-term investment horizons or extremely volatile markets, the dynamics of the price process should be modelled as a geometric rather than arithmetic random walk. Yet, it is assumed that over short term horizons (< one day) the total fractional price changes will be small and the differences between arithmetic and geometric random walks are negligible.

Temporary impact is modelled by introducing a temporary price impact function \( h(u_t) \), which represents the temporary drop in price per share due to the trade undertaken. Thus, the actual price per share received on the sale at time \( t \) is
\[ \tilde{P}_t = P_t - h(u_t) \] (4.2)
where the effect of \( h(u_t) \) does not appear in the next market price \( P_{t+1} \).

Putting Equation (4.1) into Equation (4.2) results in the capture of a trajectory, i.e., the sum of all trades \( u_t \) for the effective price \( \tilde{P}_t \) for \( t = 0, ..., T \):
\[ C = \sum_{t=0}^{T-1} u_t \tilde{P} \]
\[ = \sum_{t=0}^{T-1} u_t P_0 + \sum_{t=0}^{T-1} u_t \left( \sum_{i=0}^{t} (\sigma_i \xi_i - g(u_i)) \right) - \sum_{t=0}^{T-1} u_t h(u_t) \] (4.3)
\[ = XP_0 + \sum_{t=0}^{T-1} (\sigma_i \xi_i - g(u_i)) x_t - \sum_{t=0}^{T-1} u_t h(u_t) \]

### 4.2 Market Impact Functions

Similar to the majority of the literature related to optimal liquidation, the impact functions \( h(\cdot) \) and \( g(\cdot) \) are assumed to be linear. Let the permanent impact function be defined as
\[ g(u_t) \propto \gamma u_t \] (4.4)
where $\gamma \in \mathbb{R}^+$ is a proportionality constant with units ($/\text{share})/\text{share}$. Thus, for each $n$ shares sold, the security’s price is depressed in proportion to $n\gamma$. Unlike in Almgren and Chriss (2001), the temporary impact function $h(u_t)$ is assumed to be proportional to the volatility of the asset as well as inversely proportional to the traded volume in the period. This is consistent with Lehalle (2009), Labadie and Lehalle (2010), and Kissell (2013)’s $I^*$ Model. Let the temporary market impact function be

$$h(u_t) \propto \eta \frac{\sigma_t u_t}{V_t}$$  \hspace{1cm} (4.5)$$

where $V_t \in \mathbb{R}^+$ for $t = 0, \ldots, T - 1$ represents the number of shares of the stock traded by the whole market in interval $t$ and $\eta \in \mathbb{R}^+$ is a proportionality factor defined by the user. Plugging Equation (4.4) and Equation (4.5) into the capture Equation (4.3) results in

$$C = XP_0 + \sum_{t=0}^{T-1} \left( \sigma_t \xi_t - \gamma u_t \right) x_t - \sum_{t=0}^{T-1} \left( \eta \frac{\sigma_t u_t^2}{V_t} \right)$$

$$= XP_0 + \sum_{t=0}^{T-1} \sigma_t \xi_t x_t - \sum_{t=0}^{T-1} \gamma u_t x_t - \sum_{t=0}^{T-1} \left( \eta \frac{\sigma_t u_t^2}{V_t} \right)$$

$$= XP_0 + \sum_{t=0}^{T-1} \sigma_t \xi_t x_t - \sum_{t=0}^{T-1} \gamma (x_t - x_{t+1}) x_t - \sum_{t=0}^{T-1} \left( \eta \frac{\sigma_t u_t^2}{V_t} \right)$$  \hspace{1cm} (4.6)$$

$$= XP_0 + \sum_{t=0}^{T-1} \sigma_t \xi_t x_t - \frac{1}{2} \gamma \left( X^2 + \sum_{t=0}^{T-1} u_t^2 \right) - \sum_{t=0}^{T-1} \left( \eta \frac{\sigma_t u_t^2}{V_t} \right)$$

$$= XP_0 - \frac{1}{2} \gamma X^2 + \sum_{t=0}^{T-1} \sigma_t \xi_t x_t - \sum_{t=0}^{T-1} \left( \frac{\eta \sigma_t}{V_t} + \frac{1}{2} \gamma \right) u_t^2$$

where the term of the type $\sum_t \sigma_t \xi_t x_t$ represents the total effect of uncertainty, and the following term represents the effect of market impact on the final capture. It is interesting to note that a portion of the effect of market impact, $\frac{1}{2} \gamma X^2$, is constant and occurs irrespective of the trade sizes.
Chapter 5

Computational Experiments

The objective of the numerical examples is to validate the following facets of the proposed framework:

1. The performance against other strategies: How does the proposed framework perform compared to other common strategies (such as TWAP) as well as other more favoured, ideal strategies (such as VWAP)?

2. The performance of the adaptive model against the static model: Since a static model only takes the information set $I_0$ into account in its decision making, it should underperform an adaptive model which takes the most up-to-date information set $I_t$ into consideration.

3. The performance in adverse market conditions: Since the whole aim of the proposed framework is to protect a risk-averse investor against downside risk, its performance in adverse price movements will be of special interest.

To examine the performance and behaviour of Problem (B), experiments are carried out using the market dynamics described previously in Section 4.1. Stock orders are simulated using a static version of the problem, where the problem is only optimized once prior to trading. The results are compared to two variants of an adaptive version, both of which implement the shrinking horizon framework described in Section 3.4. In Section 5.1, the data exploration and clean-up process is described. In Section 5.2, the various models used to estimate the parameters used in the capture Equation (4.6) are described. Later, in Section 5.3, the process of trade execution in the simulations is explained. Finally, in Section 5.4, the results obtained are reported.
5.1 Data

Simulations are executed on data from the NYSE stock market, specifically, using Coca-Cola stock (“KO”), on the first half of the month of June 2017, which corresponded to 12 trading days. Raw Trade and Quotes (TAQ) data from Wharton Research Data Services (of the University of Pennsylvania) is used. The raw TAQ data is processed to obtain daily series of market volumes $V_t \in \mathbb{Z}^+$, where $V_t$ represents the total period volume for $t = 1, ..., T$, and average period prices $\bar{P}_t \in \mathbb{R}^+$, where $\bar{P}_t$ represents the periods’ VWAP for $t = 0, ..., T$, where $T = 390$ so that each period is one minute long (from market open at 9:30am to close 4:00pm). The data is cleaned by eliminating any trades that meet any of the following criteria:

- Correction Code (“TR_CORR”) greater than 1: trade data incorrect
- Sales Condition (“TR_SCOND”):
  - containing “4”: derivatively priced
  - equals “T or U”: extended hours trades
  - equals “V”: stock option trades
  - equals “Q,0,M, or 6”: opening or closing auction trades

5,686,793 trades remain representing continuous trading activity without considering market opening and closing, or any over-the-counter trades.

5.2 Model Parameter Estimation

To simplify the computational requirements for the adaptive framework, only intraday volatilities are updated whereas intraday volumes are kept constant. Using more sophisticated models for volatilities and volumes, and updating both parameters at each time period, i.e., using all new incoming information, is assumed to improve the solution performance significantly.

5.2.1 Volatility

Intraday volatility of equities has long been known to have pronounced periodic patterns. Wood et al. (1985) and Wood et al. (1985) were among the first to report the distinct “U-shape” that intraday return volatilities exhibit, i.e., they tend to be highest around the open and close of trading. It is worth noting that with the proliferation of algorithmic
trading, intraday volatilities have ceased following the historical “U-shape”. Leaving the initial price discovery process to algorithms that trade relatively small amounts at the open has caused opening period volatility to be higher than its historical levels. Thus, nowadays, intraday volatility is higher at the open than mid-day and only increases slightly into the close (Kissell (2013)).

The seminal works on modelling and estimating intraday volatilities by Andersen and Bollerslev (1997) and Andersen and Bollerslev (1998) showed that the time-of-day (diurnal) component that led to the distinct intradaily shape had to be explicitly accounted for in any representation of the volatility process. Both works decompose the intraday volatility of high-frequency asset returns into two multiplicative components, which are easily interpreted and estimated, namely, a daily component and a diurnal component. Thus, they combine the effect of a long memory factor and account for periodicity. The daily component is assumed to be stochastic, following a Generalized Autoregressive Conditional Heteroskedasticity (GARCH) process as described in Bollerslev (1986), a common method that is still widely used to analyze and estimate the time-varying, autoregressive conditional variance of daily asset returns, by academics and practitioners alike. The diurnal component is specified to be deterministic, i.e., it is assumed to be constant across days.

In this work, intraday volatility is estimated using the Multiplicative Generalized Autoregressive Conditional Heteroskedasticity (MGARCH) model proposed by Engle and Sokalska (2012). The MGARCH model expands on the frameworks proposed in Andersen and Bollerslev (1997) and Andersen and Bollerslev (1998) by including two intradaily components instead of just one: a deterministic diurnal pattern and stochastic intradaily GARCH process. Thus, MGARCH does not assume that the intraday volatility is time-invariant. The model has been shown to provide good forecasts in a variety of settings and outperform alternatives (see Sokalska (2012) and Diao and Tong (2015)). Additionally, the structure of the model allows for updating of the estimates by take advantage of new incoming information making the MGARCH model suitable for the shrinking horizon framework.

Let the VWAP of an asset at day $j$ and period $t$ be denoted by $\bar{P}_{j,t}$ where $t = 0, ..., T$ and $j = 1, ..., J$. The continuously compounded return $r_{j,t}$ is modelled as

$$
\begin{cases}
\ln \frac{\bar{P}_{j,t}}{\bar{P}_{j,t-1}} & \text{for } t \geq 1 \\
\ln \frac{\bar{P}_{j,0}}{\bar{P}_{j,0}} & \text{otherwise}
\end{cases}
$$

(5.1)

where $\bar{P}_{j,0}$ is the arrival price of day $j$. Engle and Sokalska (2012) assume the conditional variance of intraday asset returns to be a multiplicative product of daily, diurnal, and
stochastic intraday volatility components. They describe intraday equity returns by the following process:

\[ r_{j,t} = \sigma_{j,t} \varepsilon_{j,t} \]

\[ = \sqrt{h_j s_t q_{j,t}} \varepsilon_{j,t} \]

(5.2)

where \( h_j \) is the daily variance component for day \( j \), \( s_t \) is the diurnal (periodic) variance pattern for period \( t \), \( q_{j,t} \) is the unique intraday variance component for day \( j \) and period \( t \), with \( E[q_{j,t}] = 1 \), and \( \varepsilon_{j,t} \) is an error term with \( \varepsilon \sim N(0, 1) \).

The model described in Equation (5.2) requires exact characterization of its components. \( h_j \) is estimated using a GARCH(1,1) process as described in Bollerslev (1986). The diurnal variance component is estimated by deflating the squared intraday returns using the daily variance component. This allows for an estimation of the diurnal component that is completely free from the daily variance component’s shape. Thus, the diurnal component is estimated for each period as

\[ \hat{s}_t = \frac{1}{J} \sum_{j=1}^{J} \frac{r_{j,t}^2}{h_j} \]

(5.3)

Finally, the residual intraday volatility component is also modelled as a GARCH(1,1), which empirical evidence indicates is the most appropriate GARCH(\(p, q\)) process. Thus, the intraday volatility is estimated as

\[ q_{j,t} = \omega + \alpha \left( \frac{r_{j,t-1}}{\sqrt{h_j s_{t-1}}} \right)^2 + \beta q_{j,t-1} \]

(5.4)

where \( \omega \), \( \alpha \), and \( \beta \) are regression parameters of the variance equation for the intraday returns dampened by the daily and periodic components. Figure 5.1 shows a sample estimate of the volatility distribution at the outset of trading.

5.2.2 Volume

Equities’ intra-day volumes have historically exhibited a “horizontal-J” shape where the volume traded is only slightly more at the open before decreasing at mid-day, finally increasing again significantly into the close (Kissell, 2013). It plays a significant role in the trading process as a representation of liquidity. Yet, even though many algorithms have volume as an input (see Lehalle (2009), Kissell (2013)’s I* Model, Guéant et al. (2012), Busseti and Boyd (2015), and Guéant (2015) among others), the study of volume forecasting methods has been scarce. Publications dealing with intraday volume prediction are even rarer (Szücs, 2017).
Instead of estimating the intraday per period trading volume, the total daily volume is estimated using the Autoregressive Moving Average (ARMA) model described in Kissell (2013), and then the intraday volume portions are assumed to be equal to their historical median percentages. Although these intraday portions will not be very accurate, they will be able to provide an approximation of the intraday volume distribution.

The daily volume for day $j$, $V(j)$, is forecasted as

$$V(j) = \bar{V}_j(n) \times \text{DoW}(j) + \beta \times e(j - 1)$$  \hspace{1cm} (5.5)

where $\bar{V}_j(n)$ is the $n$-day moving average or moving median volume prior to day $j$ that minimizes the variation of the forecast error relative to the true daily volume, $\text{DoW}(j)$ is a measure of the weekly cyclical pattern of trading volumes of day $j$, $\beta$ is an autoregressive sensitivity parameter estimated via OLS regression analysis, and $e(j - 1)$ is the previous day’s volume forecast error.

Trading volumes are known to spike on special event days. These days include the ones when the Federal Open Market Committee meets, the days when stock market
(a) Volume estimation error std. deviation

(b) Volume estimation MAPE

Figure 5.3: Volume estimation error analysis

index futures, stock market index options and stock options expire simultaneously (commonly known as Triple Witching), and the days when company earnings are announced, among others. Thus, before estimating the daily volumes, outliers at the 95th percentile level were removed (see Figure 5.2).

To decide whether the trailing moving average or moving median is best used for $V_j(n)$, the error between the two estimates and the true volumes are compared for different look-back periods $n$. The measure for the look back period $n$ that minimized the error’s variation, measured by the error percentages’ standard deviation and their mean absolute percent error (MAPE) measured as $\frac{|\text{errors}|}{N}$ where the errors are calculated as $\frac{V(j) - V_j(n)_{\text{mean}}}{V(j)}$ for moving average and $\frac{V(j) - V_j(n)_{\text{median}}}{V(j)}$ for the moving median. As both plots in Figure 5.3 show, $V_j(10)_{\text{median}}$ provides the smallest estimation errors.

The reason for including the $\text{Day of Week}$ (DoW) effect in Equation (5.5) is that historically, a day of week pattern has been associated with market volumes. Specifically, equity trading volumes have been observed to be at their lowest on Mondays increasing on Tuesday and Wednesday, and decreasing on Thursday. Trading volumes on Friday depend on the asset. Recently trading has increased on Friday as portfolio managers have tried to avoid holding any weekend risk [Kissell (2013)]. For each week in the in-sample period, the percentage of actual volume traded on the day is compared to the average volume in the week. These daily portions for each week represent that week’s DoW factor. Similar to $V_j(n)$, the mean and median DoW factors as well as a baseline without the DoW factor were compared to the actual volumes to find out whether the mean or median portions are more suitable. The analysis of three alternatives’ errors, displayed in Table 5.1, show that the median DoW factor leads to the lowest errors as it dominates the two other alternatives in every metric except for the median error. Thus, $\text{DoW}(j)$ represents the median percentage that the asset gets traded on day $j$ of the week compared to its average volume in a week.
Table 5.1: Day of Week effect error analysis

<table>
<thead>
<tr>
<th>Moving 10-day Median</th>
<th>No DoW</th>
<th>Mean DoW</th>
<th>Median DoW</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean Error</td>
<td>-0.0196</td>
<td>-0.0189</td>
<td>0.0070</td>
</tr>
<tr>
<td>Median Error</td>
<td>-0.0125</td>
<td>-0.0069</td>
<td>0.0190</td>
</tr>
<tr>
<td>MAPE</td>
<td>0.2155</td>
<td>0.2147</td>
<td>0.2096</td>
</tr>
<tr>
<td>Error Std. Deviation</td>
<td>0.2742</td>
<td>0.2725</td>
<td>0.2659</td>
</tr>
</tbody>
</table>

Given the trailing median estimator, $V_j(n)$, and the median DoW factor, $\text{DoW}(j)$, the error term can be calculated as

$$ e(j) = V(j) - \bar{V}_j(10)_{\text{median}} \times \text{DoW}(j)_{\text{median}} $$

An OLS regression of the error term on its one-day lagged term is run, that is, for $e(j) = \alpha + \beta \times e(j - 1)$ the regression parameters $\alpha$ and $\beta$ are calculated. The full ARMA model described by Eq.(5.5) provides immense improvements on a simple 10-day moving median model as Table 5.2 shows. The ARMA model dominates the 10-day moving median model in all reported metrics and improves the correlation with the actual volumes from 0.3056 to 0.8470, a 177% improvement.

Finally, the intraday volume in (4.6) is calculated as

$$ V_{j,t} = \nu_t(n) \times V(j) $$

(5.6)

where $\nu_t(n)$ is the trailing n-day median per-period volume percentage for period $t$. Again, although $V_{j,t}$ will not be very accurate (with a total MAPE of 1.5618), it does provide a good indication of the intraday volume profile. Please refer to Figure 5.4 for an example of an estimated intraday volume distribution.

Table 5.2: Improvements in volume prediction

<table>
<thead>
<tr>
<th>Moving 10-day Median</th>
<th>ARMA Model in Eq. (5.5)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean Error</td>
<td>-0.0197</td>
</tr>
<tr>
<td>Median Error</td>
<td>-0.0125</td>
</tr>
<tr>
<td>MAPE</td>
<td>0.2157</td>
</tr>
<tr>
<td>Error Std. Deviation</td>
<td>0.2743</td>
</tr>
</tbody>
</table>
5.3 Simulation of Execution

5.3.1 Experimental Setup

For each day used to conduct the simulations, three sets of experiments were conducted: a static, an adaptive, and an aggressively adaptive version. For the static set of experiments, Problem (B) is solved only once at time $t = 0$. The chosen benchmark is the asset’s opening price, i.e., $\Psi = P^0 \times X_0$. The resulting trading schedule is then executed unaltered, i.e., the problem is not re-optimized and new incoming information during the trading horizon is ignored. In this set of experiments, the shrinking horizon framework is not implemented.

Conversely, both the adaptive and the aggressively adaptive set of experiments implement the shrinking horizon framework, thus taking advantage of new incoming information. At the beginning of each trading period $t$, the problem is (re-)optimized using the most current parameter estimates and only the ‘here-and-now’ decision variable $u_0$ is executed. The adaptive strategy always uses the experiment day’s opening price in the benchmark, i.e., $\Psi_t = P^0 \times X_t$, whereas the aggressively adaptive strategy selects a period’s benchmark according to the rules in Algorithm 1.

The main insight that motivates the aggressively adaptive strategy is that any losses against a high benchmark contain as a subset all losses against a lower benchmark.

\begin{algorithm}
\textbf{Algorithm 1} Setting benchmark in aggressively adaptive strategy
\begin{algorithmic}
  \If{$P_t \geq P^0$} \Comment Set $\Psi_t = P_t \times X_t$
  \Else \Comment Set $\Psi_t = P^0 \times X_t$
  \EndIf
\end{algorithmic}
\end{algorithm}
For example, if the initial (low) benchmark is set as $P^0 = $20, the current price is $P_t = $30, and there are $X_t = 100$ shares remaining to liquidate in the coming $T - t$ periods then the original benchmark, used in the adaptive strategy, would be $\Psi_t = 2000$. If, on the other hand, the current price is used, then the benchmark would be $\Psi_t = 3000$. A capture that lies between both benchmarks, say $C^{(i)} = 2500$, would be treated differently depending on the benchmark being used. If the initial benchmark of $\Psi_t = 2000$ is used, then the capture would be considered a gain and ignored by the minimization as it would lie above the benchmark. If the higher benchmark is used, then the same instance would count as a loss and be included in the LPM minimization.

Thus, any instances that are included in the minimization using a lower benchmark are also included when a higher benchmark is used, but not vice-versa. The use of the higher benchmark can be seen as a more risk-averse strategy than the adaptive strategy. Another way of viewing this strategy is that the more favorable the market conditions, the more risk averse the strategy becomes, since it views more instances, that would normally be viewed as gains, negatively. In fact, the better the period price, the more unfavorably said gains are viewed by the strategy, since compared to the higher benchmark they would count as greater losses. In other words, it would exhibit similar attitudes towards risk as an investor with increasing absolute risk aversion. Such an investor would prefer to spend some of the trade gains on market impact during favorable price movement to exposing more capital to adverse price movements in the remainder.

5.3.2 Problem Parameter Selection

All experiments assume that an investor has an initial position of $X = 100,000$ shares of Coca-Cola (“KO”). The objective of all experiments is to minimize the implementation shortfall, i.e., the average losses below the benchmark or $\text{LPM}_1(\Psi, C)$. Similar to Almgren and Chriss (2001), the market impact proportionality constants are assumed to be proportional to the bid-ask spread and inversely proportional to the volume. Thus, for $\gamma$ it is $1.5 \times \frac{\text{bid-ask spread}}{1\% \times V(j)}$ and for $\eta$ it is $\frac{\text{bid-ask spread}}{10\% \times V(j)}$. The bid ask spread is assumed to be 2 basis points of the period’s opening price, thus changing for each reoptimization. This assumption is similar to the one made in Busseti and Boyd (2015) and is reasonable for highly liquid stocks such as the one used in the experiments here.\footnote{Note, the assumption would not result in different results as long as the stocks used are liquid. Thus, the assumption should hold for all large-cap stocks. For smaller companies, the bid-ask spread would have to be increased to reflect the lack of trading and liquidity associated with this asset class.}
5.3.3 Number of Scenarios & Problem Instances Selection

No method for selecting the number of sample paths or the number of problem instances exists. An initial set of experiments with $N = 10,000$ and $K = 10$ was deemed to computationally intensive, especially given the fact that each day in the out-of-sample set has one static, 389 adaptive, and 389 aggressively adaptive optimizations, for a total of 779 optimizations periods. Through trial and error, it was concluded that the number of sample paths $N$ had a larger effect on computation times than the number of problem instances $K$ since multiple instances of the problem $m$ can be computed in parallel.

To ease the computational burden of the problem, the number of sample paths was reduced to $N = 2,500$, whereas the number of independent problem instances was increased to $K = 15$. To validate that the new problem configuration results in decisions that are significantly similar to the original decisions (of the problem with $N = 10,000$ and $K = 10$), the trajectories of two sets of static experiments were compared using a Granger-causality test. A Granger-causality analysis is a set of hypothesis tests that checks if a (nonlinear) time-series has casual links to another time-series, and may be used to test two time-series’ similarity. The appropriate Python function uses four different hypothesis tests to check for Granger-causality: a likelihood ratio test ($\text{lrtest}$), a sum of squares based chi-squared test ($\text{ssr_chi2test}$), a sum of squares based F-test ($\text{ssr_ftest}$), and a parameter F-test ($\text{params_ftest}$). All tests showed significant results with $\text{params_ftest}$ resulting in the largest p-value of $p = 1.9934e - 08$. The congruence of the results of all four hypothesis test, run with a max-lag of 1, indicate that the resulting trajectories of both problem configurations may be considered significantly similar.

Thus, for the final configuration of the experiments at each optimization period, a set of $K = 15$ independent instances of the problem are optimized, each containing a set of $N = 2,500$ sample paths. The final trading trajectory is then the average of the resulting 15 trading trajectories. Thus, each day used in the simulation of order executions 15 optimizations are conducted for each static experiment and ($388 \times 15 = 5820$) subsequent optimizations for each of the adaptive and aggressively adaptive experiments. Hence, the total resulting number of optimizations per day is 11,655.

5.3.4 In-sample Data Window

The experiments were conducted based on a “sliding window” approach: for each day used in the simulation of order executions, the various parameters were estimated on data from a “window” covering some preceding timeframe, since the most recent historical data is the most relevant for model calibration and the processes governing the
observations change over time. For daily parameters, a look-back window of 65 months was selected. Thus, the daily parameters required for the first day of experiments June 1st 2017, the daily GARCH(1,1) parameters for the daily component of volatility $h_1$ and the parameters for daily volume estimate $V(1)$, were estimated using daily data from 2012-01-01 to 2017-05-3. For diurnal/periodic parameters, such as the diurnal component of intraday volatility $s_t$ and the median per period volume percentages $v_t$, the window was set to the preceding 5 months. Finally, for intraday parameters representing randomness, such as the intraday volatility component $q_{j,t}$, the previous $390 \times n$ data points were used to estimate the three parameters for the intraday GARCH(1,1) parameters, representing the data of the previous $n$ days for each period. For each simulation day $j$, the number of days $n$ used to estimate $q_{j,t}$ was tuned manually prior to trading such that about 90% of the previous day’s intraday volatility values are captured. Also note that each scenario has its own unique volatility simulation. Thus, the simulations are updated each reoptimization, which has $N = 2500$ independent simulations of the volatility process.

5.4 Results

Problem (B) was written and solved in Python using the CvxPy (Diamond and Boyd (2016)) package utilizing the MOSEK solver (ApS (2017)). At the beginning of each day of simulation $j = 1, \ldots, 12$, Problem (B) is optimized and its results stored as the static solution. Afterwards, the problem is reoptimized for the upcoming 389 remaining periods. Finally, the results of both models are compared to

- **Benchmark ($\Psi$):** which is the arrival price for day $j$, $\bar{P}_0^j$, times the initial holdings, $X$

- **Time weighted average strategy (TWAP):** where the initial holdings $X$ are liquidated at a constant rate of $u_t = \frac{X}{T}$

- **Ex-post volume weighted average strategy (Ex-VWAP):** where the initial holdings are liquidated in proportion to the actual volume curve of day $j$. This strategy would violate non-anticipativity as it assumes perfect knowledge of the volumes being traded on day $j$. This unrealistic criterion is used in order to get a sense of how the proposed framework (with a simple case study) compares to an unrealistic, unattainable version of a strategy that is widely considered the gold standard in optimal execution problems.

---

2 Note that the methods developed by Berkelaar et al. (2005) to deal with convex MSP are based on techniques first proposed by the developers of the MOSEK solver and are used by the solver to enhance computation times.
Table 5.3: Improvement over benchmark aggregate results

<table>
<thead>
<tr>
<th></th>
<th>Static</th>
<th>Adaptive</th>
<th>Aggr. Adaptive</th>
<th>TWAP</th>
<th>Ex-VWAP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean</td>
<td>-1692.93</td>
<td>-1373.51</td>
<td>-796.29</td>
<td>-1040.17</td>
<td>-209.43</td>
</tr>
<tr>
<td>Median</td>
<td>-2585.53</td>
<td>-827.81</td>
<td>9.53</td>
<td>819.63</td>
<td>684.02</td>
</tr>
<tr>
<td>St. Dev.</td>
<td>7582.14</td>
<td>8825.35</td>
<td>8445.44</td>
<td>10185.88</td>
<td>9948.57</td>
</tr>
</tbody>
</table>

5.4.1 Aggregate Results

Table 5.3 shows the relevant statistics of the aforementioned strategies compared to the benchmark, also represented in Figure 5.6. First, as the negative means in Table 5.3 indicate, none of the strategies was able to outperform the benchmark on average. Additionally, Table 5.3 shows that both the static and adaptive strategies underperform both the TWAP and the Ex-VWAP strategies on average. The aggressively adaptive strategy, on the other hand, manages to outperform the TWAP strategy while underperforming the unattainable, unrealistic Ex-VWAP strategy. Additionally, it outperforms both the static and adaptive strategies on average. It is worth noting that the aggressively adaptive strategy is negatively skewed, which, combined with the positive median, indicates that its mode is positive as well. Thus, the majority of its captures outperform those of the benchmark. Finally, as indicated by the smaller standard deviations, it is clear to see that all variations of the proposed framework have more consistent performance compared to the benchmark than the TWAP and Ex-VWAP strategies.

Figure 5.5 supports the previous analysis. Figure 5.5 is similar to an efficient frontier plot in that it plots the different strategies’ yield against their volatility. The higher a datum is on the plot, the better its capture performed compared to the benchmark. The farther to the left a datum is, the smaller its capture’s standard deviation is. Thus, strategies in the upper left corner would represent strategies with high captures compared to the benchmark’s that performed more consistently and thus represent less risky strategies. As Figure 5.5 shows, all proposed strategies lie to the left of the TWAP and Ex-VWAP strategies indicating more consistent performance. They may thus be said to be less risky strategies. Furthermore, Figure 5.5 shows the point associated with the aggressively adaptive strategy lying higher than the two other variations of the proposed strategy and the TWAP strategy, but lower than the Ex-VWAP strategy. Thus, since the aggressively adaptive strategy lies higher and to the left of the TWAP strategy, it may be said that its performance dominates that of the TWAP strategy.

The outperformance of the adaptive and the aggressively adaptive strategy, on average, compared to the static strategy (as indicated by the higher position in Figure 5.5) supports the hypothesis that a modular, reactive strategy would outperform a
static strategy. The improvement can be attributed to the use of the shrinking horizon framework. Similar to how the shrinking horizon framework added an aspect of adaptability the two adaptive strategies compared to the static strategy, the increased reactivity of the aggressively adaptive strategy compared to the adaptive strategy lead to its improved results.

5.4.2 Results by Market Condition

Next, the results are investigated by market condition. Define a day \( j \) as having positive market conditions as one with an average price, \( \bar{P}_j \), that is greater than its opening price, \( \bar{P}^0_j \). Conversely, a day \( j \) with negative market conditions is one one with \( \bar{P}_j < \bar{P}^0_j \). In the 12-day sample, half exhibit positive conditions, namely on June 1, 5, 7, 9, 14, and 15; the remaining days exhibit negative market conditions.

The results, broken down by market condition, are reported in Table 5.4, and summarized in Figure 5.7 and Figure 5.8. This distinction helps elucidate the models’ aggregate results. The two box-plots in Figure 5.7 show that the strategies perform differently in the two different regimes. On the one hand, in positive market conditions,
Figure 5.6: Improvement over benchmark captures

The upper and lower boundaries of each box represent the 75th and 25th percentiles respectively, and the red horizontal line inside the box depicts the median. The whiskers represent the minimum and maximum values within 1.5 times the interquartile range, whereas plus signs represent outliers, i.e., any datum more than 1.5 times the interquartile range away from the boxes.

all strategies have positive means and medians (see Table 5.4), i.e., they outperform the benchmark on average.

Figure 5.7(a) indicates that the static and adaptive strategies are the only strategies that have instances that underperformed the benchmark during positive market conditions. These two strategies also exhibit higher standard deviations than the other strategies. Their corresponding data points in Figure 5.8(a) lie to the right and below those of all other strategies. The aggressively adaptive strategy, on the other hand, performs relatively well during positive market conditions. It has the smallest standard deviation during positive market conditions as Table 5.4 and Figure 5.8(a) show and was able to outperform the benchmark on all days with positive market conditions as indicted by Figure 5.7(a). As far as the order with which the strategies perform relative to one another during positive market conditions, the Ex-VWAP performs the best, followed by the TWAP, aggressively adaptive, adaptive, and static strategies, in that order.

On the other hand, the box-plots in Figure 5.7(b) show that on days with
Table 5.4: Improvement over benchmark aggregate results by market condition
(+) positive market conditions (-): negative market conditions

<table>
<thead>
<tr>
<th>Strategy</th>
<th>Mean</th>
<th>Median</th>
<th>Std. Dev.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Static (+)</td>
<td>3610.15</td>
<td>1573.23</td>
<td>6675.85</td>
</tr>
<tr>
<td>Static (-)</td>
<td>-6996.01</td>
<td>-6505.18</td>
<td>3796.62</td>
</tr>
<tr>
<td>Adaptive (+)</td>
<td>5083.63</td>
<td>2476.45</td>
<td>6155.93</td>
</tr>
<tr>
<td>Adaptive (-)</td>
<td>-7830.65</td>
<td>-7151.31</td>
<td>5778.26</td>
</tr>
<tr>
<td>Aggr. Adaptive (+)</td>
<td>5645.12</td>
<td>4060.20</td>
<td>5462.08</td>
</tr>
<tr>
<td>Aggr. Adaptive (-)</td>
<td>-7237.71</td>
<td>-6909.78</td>
<td>5244.18</td>
</tr>
<tr>
<td>TWAP (+)</td>
<td>6886.06</td>
<td>4085.00</td>
<td>6019.37</td>
</tr>
<tr>
<td>TWAP (-)</td>
<td>-8966.39</td>
<td>-8978.94</td>
<td>6421.97</td>
</tr>
<tr>
<td>VWAP (+)</td>
<td>7584.31</td>
<td>6076.44</td>
<td>5642.82</td>
</tr>
<tr>
<td>VWAP (-)</td>
<td>-8003.18</td>
<td>-7183.57</td>
<td>6334.00</td>
</tr>
</tbody>
</table>

In negative market conditions, none of the strategies was able to outperform the benchmark. Table 5.4 confirms this observation, where, in negative market conditions, all strategies have negative means and medians. Due to the definition of negative market conditions, it is easy to see that if the majority of the prices are unfavourable then irrespective of liquidation, it will be difficult for a strategy to achieve a capture that is better than the benchmark’s. The order of the relative performance of the strategies is almost reversed to those observed during days with positive market conditions. For all reported measures, the upper and lower boundaries of each box represent the 75th and 25th percentiles respectively, and the red horizontal line inside the box depicts the median. The whiskers represent the minimum and maximum values within 1.5 times the interquartile range, whereas plus signs represent outliers, i.e., any datum more than 1.5 times the interquartile range away from the boxes.
the static strategy performs best, followed by the aggressively adaptive, adaptive, Ex-VWAP, and TWAP strategies, in that order. In Figure 5.8(b) the data points associated with all three variations of the proposed framework lie higher and to the left of the TWAP and Ex-VWAP’s. Thus, all three variations dominate both the TWAP and Ex-VWAP strategies. These results combined with those in Table 5.4 indicate that the three proposed strategies outperform the TWAP and Ex-VWAP strategies during negative market conditions, validating the use of the LPM as a risk measure. The objective of all three proposed strategies was to minimize the losses below the opening price, thus, all three strategies succeed since they perform as intended, posting better results than the TWAP and the Ex-VWAP during negative market conditions.

5.4.3 Model Behaviour

5.4.3.1 Adaptive Strategy Behavior

Although the adaptive strategy outperformed the static strategy on average, it did not manage to outperform the naïve TWAP strategy. To understand the difference in the performance of the adaptive strategy compared to the TWAP strategy, please refer to Figure 5.9 which shows the trading patterns during the first day of simulations on June 1st, 2017. The vertical lines indicate trades that are larger than the period’s TWAP order by 200 shares. The static strategy is “front-loaded” similar to the trajectory of
Almgren and Chriss (2001)’s static mean-variant trajectory: The strategy liquidates rapidly as early as possible in the program to reduce the risk that the natural price movement will move away from the benchmark.

Whereas the static strategy shows a stable, consistent trading trajectory, the adaptive strategy appears to be liquidating haphazardly. Figure 5.9 shows the adaptive strategy going through sporadic periods of increased trading activity. A closer look at the first half hour of trading, depicted in Figure 5.10, provides some insight into the framework’s behaviour.

Figure 5.10 shows that increased trading, represented by the dotted vertical lines, occur when the opening price for the period falls within a specific range. One way to explain the behaviour is that while the period’s opening price is below the dashed red line, marked as “Upper bound” in the figure, the strategy is “aggressive-in-the-money” (AIM) in the sense of Kissell and Malamut (2005). AIM strategies attempt to maximize the probability that the cost will be less than the benchmark execution strategy. Thus, AIM strategies accelerate execution in periods of favorable price movements, and trade less aggressively in times of adverse market conditions. When a period’s opening price is below the benchmark, the model views the market conditions as unfavorable and decreases trading to a rate hovering around the TWAP, i.e., the slowest rate it is allowed to trade at in order to still meet the problem’s constraint of liquidating all shares by the end of the liquidation horizon. It is trying to minimize its costs during that period by minimizing market impact as much as possible.
Above the benchmark, in what can be considered favorable market conditions, the model behaves as expected from an AIM strategy while the period’s opening price is below some threshold, which once crossed trading is reduced once again to the TWAP rate. In essence, above this threshold, the strategy turns “passive-in-the-money” (PIM). PIM strategies behave inversely to AIM strategies; they decelerate trading in favorable environments. The reasoning that leads to PIM strategies is the hope that the positive price movement will continue, and thus the decreased trading would allow investors to better participate in future gains. In the case of our adaptive strategy, the underlying reason might be different, namely, that above the threshold the capture function does not record enough losses compared to the benchmark and thus falls back to a default of TWAP trading. As a result, the adaptive strategy does not take advantage of very favorable price movements, i.e., movements above the threshold, which, if they occur at the outset of trading, are taken advantage of by the static strategy, albeit inadvertently.

The advantage of the static strategy is that the effects of any initial adverse price movements are mitigated by any overly-favorable initial movements. Additionally, if any downward price movements occur in the latter three-quarters of the trading day, then the static strategy is already liquidating amounts smaller than the TWAP strategy. Thus, the impact of such unfavourable conditions is reduced even further. As a result, the static strategy can outperform the adaptive strategy by many thousand dollars, such as on June 7th 2017 (depicted in Figure 5.11) when it outperformed it by $8839.91 by taking advantage of the initial pockets of favorable price movement.
5.4.3.2 Aggressively Adaptive Strategy Behaviour

Figures 5.12 and 5.13 show the behavior of the aggressively adaptive strategy. Unlike the adaptive strategy, the aggressively adaptive strategy does not increase its trading activity when the period price, $\bar{P}_{j,t}$, falls between some band. Rather all plots show a great deal of similarity between the strategy’s increased trading activity when $\bar{P}_{j,t} \geq \bar{P}^0_j$. Specifically, the increased trading has a consistent behaviour depending on the time of the day. Both plots in Figure 5.12 indicate that the behaviour of the aggressively adaptive strategy in the first two hours of trading is different than the rest of the day.

Specifically, during the portion of the day where the static strategy is trading higher than the TWAP strategy, the aggressively adaptive strategy during its times of increased trading tries to follow the trajectory of the static trading strategy. This behavior is depicted in both Figures 5.12 and 5.13, showing the behaviour of the aggressively adaptive strategy in favorable and adverse market conditions, respectively.

Inspecting Figures 5.12(a) and 5.12(b), which depict days with periods of both favorable and adverse price movements, helps clarify the behaviour of the aggressively adaptive strategy. When $P_{j,t} \geq P^0_j$, the aggressively adaptive strategy sells roughly the same amount of shares as the static strategy in that period, i.e., both strategies have roughly the same $u_t$. Both strategies exhibit almost perfect correlation in their trading.
Conversely, when $P_{j,t} < P_{j}^{0}$, the aggressively adaptive strategy sells roughly the same amount of shares as the TWAP strategy.

When the static strategy is trading a lower number of shares than the TWAP strategy, i.e., its $u_t < \frac{X}{T}$, the behavior of the aggressively adaptive strategy during favorable price movements changes. Instead of following the trajectory of the static strategy during favorable price movements, it trades at a constant rate that is slightly higher than the TWAP rate. Inversely to its earlier behavior, it now follows the trajectory of
the static strategy during adverse price movements. Figures 5.12(a) and 5.12(b) support this observation. During a period adverse price movements on June 15th between 12:48-13:02, the aggressively adaptive strategy reduces the number of shares it is selling to roughly the same amount as the static strategy (refer to Figure 5.14). On June 1st, the period price dips below the opening price during three periods after mid-day, between 12:11-12:13, 12:13-12:16, and 12:18-12:32. Similar behaviour is observed in all three periods, where again the number of shares being liquidated is reduced to that of...
The only time when the aggressively adaptive strategy deviates from the behavior described above is during extremely favorable price movements (see Figure 5.15). On days where extremely favorable price movements occur early in the day, the strategy diverges from the static trajectory and trades a larger number of shares. On June 9th, for example, displayed in Figure 5.15(a), the aggressively adaptive strategy veers away from the static trajectory after the share price increases by $0.21 within 6 minutes. Following said price movement, the aggressively adaptive strategy starts trading at a rate of about 1400 shares per period until it has sold the remaining shares. Similar behavior can be seen on June 14th represented in Figure 5.12(b). Initially, the asset price exhibits mostly positive price movements, which the aggressively adaptive strategy reacts to as described earlier: it follows the trajectory of the static strategy during periods of positive price movement and reverts to the TWAP rate during negative ones. Similar to the previous example, the asset price goes through a period of extremely favourable price movement, albeit at a slower rate. A similar increase of $0.28 took place over an hour and twenty minutes. Thus, although the price movement was more favourable, it took much longer, which corresponded to a smaller deviation from the static strategy’s trajectory than that observed in the previous example.

Finally, a persistent characteristic of the aggressively adaptive strategy during positive market conditions is that it concludes trading before the end of the trading horizon. As mentioned earlier, the aggressively adaptive strategy behaves in a similar fashion to an investor with increasing absolute risk aversion. Thus, the more favorable the conditions, the more it trades in order to capture the positive price movements, even if the higher trading will lead to a slightly higher market impact. As a result of rapid liquidation during positive market conditions and the corresponding early termination,
the aggressively adaptive strategy might not be able to take advantage of late favorable price movements as was the case on both June 1st and June 15th in Figure 5.12.
Chapter 6

Discussion & Conclusion

In this thesis, a quasi-multi-period approach for optimal position liquidation in the presence of both temporary and permanent market impact is developed. The main contributions of this work to the existing literature are two fold. First, is the development of a quasi-multi-period model for optimal position liquidation that uses stochastic programming to deal with the problems inherent uncertainty and a Shrinking Horizon framework to avoid the dimensionality curse. The main framework is data-driven and does not assume any structure on the parameters. Thus, it can accommodate different models of the capture and all the parameters used therein, such as temporary and permanent market impact. Second, the LPM risk measure is implemented here for the first time while tackling the problem of optimal execution. Overall, the resulting program is modular, risk-averse, numerically-tractable, and assumption-free.

Three versions of the framework were tested using a simple model for the market dynamics: a static model that is only optimized prior to trading, an adaptive model that is constantly reoptimized to take new market information into account, and an aggressively adaptive model which is a more risk averse variation of the adaptive model. These three alternatives’ performances were measured against an implementation shortfall benchmark and compared to the performances of a TWAP strategy and an Ex-post VWAP strategy that violates non-anticipativity. The results of the experiments, conducted using real market data, show that

1. The adaptive and aggressively adaptive strategies were reactive to market conditions showing that although the problem was not truly multi-period, it was able to incorporate new market information in its decision making and react to it.

2. On average, the adaptive and aggressively adaptive strategies outperform the static strategy, validating the use of the Shrinking Horizon framework. The performance
of the aggressively adaptive strategy compared to the two other proposed strategies indicate that the more responsive/reactive a strategy is to market conditions, the better its performance on average.

3. During adverse market conditions, the three proposed strategies outperform the TWAP and the Ex-post VWAP strategies, supporting the use of the LPM as a risk measure.

4. During favorable market conditions, the three strategies underperformed the Ex-post VWAP strategy and only the aggressively adaptive strategy was able to outperform the naïve TWAP strategy on average. Additionally, as indicated by Figure 5.7(a), the aggressively adaptive strategy was able to consistently outperform the benchmark.

Although all three variations of the proposed framework performed well during adverse market conditions, their performance during favourable conditions was poor. Indeed, only the aggressively adaptive strategy was able to perform relatively well during friendly market conditions, even outperforming the naïve TWAP strategy on average. One reason might be that the risk measure used in the optimizations’ objective function only considered losses (which occurred mostly during unfavourable market conditions) and ignored gains completely. In that sense, the strategies performed as expected where the downside risk that occurs during negative conditions was hedged. A future research direction may be to examine the use of a utility function that attempts to maximize gains while hedging against downside risk in order to remedy the discrepancy in performance on days with favourable market conditions.

Overall, it seems that the aggressively adaptive strategy is the best suited for risk-averse investors as it has shown superior performance to its benchmark during positive market conditions and has outperformed all strategies except the static strategy on days exhibiting adverse market conditions. It is reactive to the market conditions. Contrary to the increases of the adaptive strategy, which can be almost four times as the static strategy’s largest trade (see Figure 5.9), the aggressively adaptive strategy’s modulations are more controlled and follow the trajectory of the static strategy. The fact that the aggressively adaptive strategy follows the trajectory of the static strategy is particularly interesting as both strategies do not communicate with one another. All re-optimizations throughout the day are independent of one another. Thus, although the aggressively adaptive strategy has more periods with increased trading rates than the adaptive strategy, each individual trade incurs a lower market impact. Additionally, the fact that the aggressively adaptive strategy trades in favorable time periods more often than the adaptive strategy reduces the overall effect of such a market impact on the aggressively adaptive strategy’s overall capture.
Finally, the strategies that performed best during positive and negative market conditions were the static and the Ex-post VWAP strategies, respectively. Both are highly dependent on the price movement at the beginning and end of trading, with the static strategy being front-loaded and the Ex-post VWAP strategy being extremely back-loaded (20% and 5% of the Ex-post VWAP strategy total trades occur in the final half hour and 2 minutes, respectively. Thus, the strategy would most likely have a much lower capture than the experiments indicate due to the extremely high market impact it incurs at the end of the trading day.). Yet, the aggressively adaptive strategy does not have such a dependency. Rather, it reacts to the market conditions. If the price is moving in its favor very rapidly at the beginning of trading, it will react and become very front-loaded. Otherwise, as Figure 5.12 shows, it takes advantage of favorable price movements whenever it can throughout the day.
Appendix A

Python Code

A.1 Imported Libraries

```python
%matplotlib inline
import numpy as np
import scipy as sp
import matplotlib.pyplot as plt
import pandas as pd
pd.set_option('display.width', 500)
pd.set_option('display.max_columns', 100)
from IPython import display
from datetime import time
import datetime
import time
import cvxpy as cvx
import quandl
import arch
import statsmodels as stm
import statsmodels.api as sm
import pickle
from tqdm import tqdm
from dask import compute, delayed
import dask.threaded
from copy import deepcopy
from IPython.core.debugger import Tracer
```

A.2 Data Exploration & Cleaning

```python
def cleanTicData(ticData):
    # correction code greater than 1: trade data incorrect
    ticData = ticData[ticData.TR_CORR <= 1]
    # Filter by sales condition:
```

51
# sales condition containing 4 : derivatively priced
# sales condition T or U : extended hours trades
# sales condition V : stock option trades
# sales conditions Q, O, M, 6 : opening and closing auction trades

chars = ['4', 'T', 'U', 'V', 'Q', 'O', 'M', '6']

wCodes = [any([n in i for n in chars]) for i in set(ticData.TR_SCOND)]
codes = [i for (i, v) in zip(set(ticData.TR_SCOND), wCodes) if v]
ticData = ticData[~ticData.TR_SCOND.isin(codes)]
ticData = ticData[['DATE', 'TIME_M', 'PRICE', 'SIZE']]
ticData.columns = ['Date', 'Time', 'Price', 'Volume']

# Set field types to string & concatenate
ticData['Date'] = ticData['Date'].astype(str)
ticData['Time'] = ticData['Time'].astype(str)
ticData['Timestamp'] = ticData['Date'] + ticData['Time']

# Create new column 'Timestamp' and set field type to datetime
ticData['Timestamp'] = pd.to_datetime(ticData['Timestamp'], format='%Y%m%d%H%M:%S.%f')
ticData['Timestamp'] = ticData['Timestamp'].values.astype('<M8[s]')

# Remove 'Date' & 'Time' columns
ticData = ticData[['Timestamp', 'Price', 'Volume']]
# Make 'Timestamp' column the index
ticData.index = ticData.pop('Timestamp')
# Price_i * Volume_i (to be used in aggregate table ahead)
ticData['Price x Volume'] = ticData['Price'] * ticData['Volume']

print(ticData.shape[0], " continuous trades remaining.")

return ticData

def summarizeTicData(ticData):
    ticSummary = ticData.groupby(pd.TimeGrouper(freq='Min')).sum()
ticSummary['P_VWAP'] = ticSummary['Price x Volume'] / ticSummary['Volume']
ticSummary = ticSummary[['Price x Volume', 'Volume', 'P_VWAP']]

    # Remove periods outside trade hours
    ticSummary = ticSummary.between_time(datetime.time(9, 30), datetime.time(16, 0), include_end=False)

    # Remove weekends
    ticSummary = ticSummary.iloc[(ticSummary.index.weekday <= 4), :]

    # Remove NYSE Holidays
    holidays = [datetime.date(2017, 1, 16), datetime.date(2017, 2, 20), datetime.date(2017, 4, 14), datetime.date(2017, 5, 29)]
    for i in holidays:
        ticSummary = ticSummary.drop(ticSummary.index[ticSummary.index.date == i])

    print('Total time periods:', ticSummary.shape[0])
ticSummary = ticSummary[['P_VWAP', 'Volume']]
ticSummary.columns = ['Price', 'Volume']

    return ticSummary
def retrieveDailyParams(Ticker, startDate="2012-01-01", endDate="2017-06-21"):
    quandl.ApiConfig.api_key = "***************"
    ticker = "WIKI/" + Ticker
dailyTic = quandl.get(ticker, start_date=startDate, end_date=endDate)
    returns = 100*dailyTic['Adj. Close'].pct_change().dropna()
    return dailyTic, returns

def summarizeTicData(ticData):
    ticSummary = ticData.groupby(pd.TimeGrouper(freq='Min')).sum()
    ticSummary[['P_VWAP']] = ticSummary[['Price x Volume']]/ticSummary['Volume']
    ticSummary = ticSummary[['Price x Volume', 'Volume', 'P_VWAP']]  
    # Remove periods outside trade hours
    ticSummary = ticSummary.between_time(datetime.time(9, 30), datetime.time(16, 0), include_end=False)  
    # Remove weekends
    ticSummary = ticSummary.iloc[(ticSummary.index.weekday <= 4), :]
    # Remove NYSE Holidays
    holidays = [datetime.date(2017,1,16),datetime.date(2017,2,20),datetime.date(2017,4,14),datetime.date(2017,5,29)]
    for i in holidays:
        ticSummary = ticSummary.drop(ticSummary.index[ticSummary.index.date == i])
    print('Total time periods:', ticSummary.shape[0])
    ticSummary = ticSummary[['P_VWAP','Volume']]  
    ticSummary.columns = ['Price','Volume']
    return ticSummary

A.3 Volatility Estimation

def dailyGARCH(returns, datesIn):
    am = arch.arch_model(returns, vol='Garch', p=1, q=0, o=1, dist='Normal');
    res = am.fit(disp='off');
    estimGARCHin = 0.1 * res.conditional_volatility[datesIn]
    forecast = res.forecast(horizon=1)
    return estimGARCHin, forecast

### Get Daily Opening Prices

# Get Daily Opening Prices

def getOpeningPrices(ticData, datesI, ticI, per):
    process = lambda i: ticData[str(i)].Price.iloc[0]
    values = [delayed(process)(i) for i in datesI]
    results = compute(*values, get=dask.threaded.get)
    P0 = pd.Series(results, index = ticI.iloc[(ticI.index.time == per[0])].index.copy())
### Calculate Log-Returns

```python
def calculateReturns(ticIn, P0, periods):
    calcRet = lambda i: np.log(ticIn.iloc[ticIn.index.time == periods[i]].Price / P0) if i == 0 else np.log(ticIn.iloc[ticIn.index.time == periods[i]].Price / ticIn.iloc[ticIn.index.time == periods[i-1]].Price.values)
    vals = [delayed(calcRet)(i) for i in range(390)]
    res = compute(*vals, get=dask.threaded.get)
    Rti = pd.concat(res)
    Rti.sort_index(inplace=True)
    return Rti
```

### Diurnal Variance

```python
def diurnalVariance(Rti, estimGARCHin, datesI, periods):
    tempS = pd.Series()
    for i in datesI:
        ht = estimGARCHin[str(i)]
        rt = Rti[str(i)]
        tempS = tempS.append(rt**2/ht)

    calcSi = lambda j: pd.Series(tempS[j].mean(), index=[j])
    vals = [delayed(calcSi)(j) for j in periods]
    results = compute(*vals, get=dask.threaded.get)
    si = pd.Series([[i[0] for i in results].index[[i.index[0] for i in results]]
                    results = None
                    vals = None
    return si
```

### Intraday Variance

```python
def intradayVariance(Rti, estimGARCHin, si, datesIn, periods, horizon=3900):
    Rti_ = Rti[-horizon:]
    zti = pd.Series(index=Rti_.index.copy())
    if np.size(datesIn) > 1:
        for t in datesIn:
            h = estimGARCHin[t]
            for i in periods:
                per = datetime.datetime.combine(t,i)
                if per in Rti_.index:
                    r = Rti_[per]
                    s = si[i]
                    zti.loc[per] = (r/pow(sqrt(h*s)))
    else:
        t = datesIn
        h = estimGARCHin[t]
```
for i in periods:
    per = datetime.datetime.combine(t, i)
    r = Rti[per]
    s = si[i]
    zti.loc[per] = (r / np.sqrt(h * s))

zti = zti.dropna()

### Model Residual Volatility as a GARCH(1,1) process [Eq. (8)]
am2 = arch.arch_model(zti, mean='Zero', vol='Garch', p=1, o=0, q=1);
res2 = am2.fit(disp='off');

qti = res2.conditional_volatility.copy()

return res2, qti

def intradyVarianceForecastSimulation(resParams, qti, horizon = 390, nScen = 100, mean = 0):
sims = np.zeros([nScen, horizon])
qti_ = qti * np.ones([nScen, 1])

for i in range(horizon):
    zti_1 = np.random.normal(mean, qti_)
    qti = resParams['omega'] + resParams['alpha [1]'] * zti_1 ** 2 + qti * resParams['beta [1]']
    sims[:, i] = qti.T.copy()

return np.mean(np.transpose(sims), axis = 1)

def tuneVolatilityParams(i):
    a = 1
    c = 0

    if i == 0:
        days = 10
    else:
        days = 8
    elif i == 3:
        days = 8
    else:
        days = 14
    a = 5
    c = -1

    def tuneVolatilityParams(i):
        a = 1
        c = 0

        if i == 0:
            days = 10
        elif i == 1:
            days = 8
        elif i == 2:
            days = 8
        else:
            days = 14
        a = 5
        c = -1
        
        elif i == 3:
            days = 8
        else:
            days = 14
        a = 5
        c = -0.75

        elif i == 4:
            days = 14
        elif i == 5:
            days = 3
        elif i == 6:
            days = 3
        elif i == 7:
            days = 3
        elif i == 8:
The code block appears to be part of a method that estimates volatilities using GARCH models. It includes conditions based on the index `i`, which determine the parameters `a`, `c`, and `days` accordingly. The method returns these parameters along with `days`, `a`, and `c`.

```python
def estimTunedVolatilities(datesIn, returns, Rti, periods, i, tuneParam = None, totalHorizon = 390):
    if callable(tuneParam):
        days, a, c = tuneParam(i)

    # Find first & last day used for estimations
    lastDay = datesIn[-1]
    firstDay = lastDay.replace(year=lastDay.year-5)

    # Set returns window
    rets = returns[str(firstDay):str(lastDay)]

    # Estimate Daily GARCH Volatility
    estimGARCHin, forecast = dailyGARCH(rets,'2017')

    # Calculate Diurnal Variance
    si = diurnalVariance(Rti, estimGARCHin, datesIn, periods)

    # Calculate Intraday Variance
    res2, qti = intradyVariance(Rti, estimGARCHin, si, datesIn, periods, horizon = (days*390))

    # Simulate Intraday Variance
    simQti = intradyVarianceForecastSimulation(res2.params, qti[-1], horizon = totalHorizon, nScen = 100)**2

    # Calculate per period Volatility Estimate
    dailyvol = 0.1 * np.sqrt(forecast.variance.dropna().values[0][0])
    intraDayVols = a*simQti+c
    sigma = np.sqrt(pd.Series(intraDayVols, index = si.index)*si*dailyvol)

    return sigma
```

### A.4 Volume Estimation

```python
def removeVolOutliers(totalVolumeIn, alpha = 0.975):
    # sort volumes
    sortedVol = np.sort(totalVolumeIn.values)

    # Find upper & lower bounds for outliers
    index_U = round(sortedVol.shape[0]*alpha)
    index_L = round(sortedVol.shape[0]*(1-alpha))
    volumeCeiling = sortedVol[index_U]
```
volumeFloor = sortedVol[index_L]

# Prune Volumes
totalVolumeIn = totalVolumeIn[totalVolumeIn<volumeCeiling]
totalVolumeIn = totalVolumeIn[totalVolumeIn>volumeFloor]
return totalVolumeIn

def getDoW(totVolIn):
    # Find weeks with an average of more than 4 days per week
    weekVals = [(totVolIn.groupby(totVolIn.index.week).count()>20).values]

    # Find week values
    weeks = np.array(list(set(totVolIn.index.week)))
    years = np.sort(list(set(totVolIn.index.year)))
    index = [tuple((year,week)) for year in years for week in weeks]
    index = pd.MultiIndex.from_tuples(index, names=['year', 'week'])

    # Find day of week effect
    weekdayPortions = pd.DataFrame(index=index, columns=[i for i in range(5)])
    for year in years:
        totalVolYear = totVolIn[str(year)]
        for week in weeks:
            portions = totalVolYear[totalVolYear.index.week==week]
            # Only check weeks with more than 5 days
            if portions.shape[0] == 5:
                portions = portions/np.mean(portions)
            weekdayPortions.loc[(year,week)] = pd.Series(portions).values
    weekdayPortions = weekdayPortions.dropna()
dayOfWeekMedian = weekdayPortions.median()
return dayOfWeekMedian

def getARCoeffs(totVolIn, movMean, dayOfWeekMedian):
    # Calculate DoW modified MovMedian
    t = movMean.index
    modMovMedian = pd.Series(index=t.copy())
    for i in t:
        DoW = i.dayofweek
        modMovMedian.loc[i] = movMean.loc[i]*dayOfWeekMedian[DoW]
    errorsMovMedian = (totVolIn - modMovMedian).dropna()
    lagErrors = errorsMovMedian[1:].tolist()
    origErrors = errorsMovMedian[:-1].tolist()
    mod = sm.OLS(lagErrors, origErrors)
    resOLS = mod.fit()
    AR_coeff = resOLS.params[0]
return AR_coeff

def forecastDailyVolume(totVolIn, AR_coeff, MDV, dayOfWeek):
    t = totVolIn.index[-1].dayofweek
Appendix A

```python
DoW = dayOfWeek[t]
actualVolume = totVolIn[-1]
movMeanVol = MDV[-1]
predError = actualVolume - DoW*movMeanVol
return DoW*movMeanVol + AR_coeff*predError
```

```python
def getVolPortions(datesI, totVolIn, ticIn):
    volumePortions = pd.DataFrame(columns = [i for i in range(390)], index = datesI)
    for i in datesI:
        if i in totVolIn.index.date:
            portions = ticIn.Volume[str(i)]/totVolIn[str(i)]
            volumePortions.loc[str(i),:] = pd.Series(portions).values
    volumePortions = volumePortions.astype(float)
    return volumePortions
```

```python
def estimVolumePortions(datesIn, totalVolumeIn, ticIn):
    lastDay = datesIn[-1]
    firstDay = lastDay.replace(year=lastDay.year -5)
    totVolIn = totalVolumeIn[str(firstDay):str(lastDay)]
    totVolIn = removeVolOutliers(totVolIn, alpha = 0.975)
    MDV = totVolIn.rolling(center=False,window=10).median().dropna().iloc[:-1].shift()
    dayOfWeekMedian = getDoW(totVolIn)
    AR_coeff = getARCoeffs(totVolIn, MDV, dayOfWeekMedian)
    forcVol = forecastDailyVolume(totVolIn, AR_coeff, MDV, dayOfWeekMedian)
    volumePortions = getVolPortions(datesIn, totVolIn, ticIn)
    periodVol = (1/volumePortions.median().sum())*volumePortions.median() * forcVol
    return periodVol
```

A.5 Optimization Experiments

```python
def getP0_out(ticData, dayOut, ticOutIndex, periods):
    T=390
    ticsOut = ticData[str(dayOut)].Price
    process = lambda i,T: ticsOut.between_time(str(periods[i]),str(periods[i+1]))[0] if i != (T-1) else ticsOut.between_time(str(periods[i]),str(datetime.time(16)))[0]
    values = [delayed(process)(i,T) for i in range(T)]
    results = compute(*values, get=dask.threaded.get)
    P0_out = pd.Series(results, index = ticOutIndex.copy())
    return P0_out
```

```python
def runOptimization(N,T,X, P_Arr, sigma, periodVol, P_Arr_Init):
    x = cvx.Variable(T)
```
Appendix A

u = cvx.Variable(T)
z = cvx.Variable(N)
Psi = cvx.Parameter(sign="positive")
perm = 1.5*(2e-4 * P_Arr)/(0.01*periodVol.sum())
temp = (2e-4 * P_Arr)/(0.1*periodVol.sum())

constraints = [x[0] == X,
cvx.sum_entries(u) == X,
x >= 0,
u >= 0,
z >= 0]

for i in range(T-1):
    constraints.append((u[i] == x[i] - x[i+1]))
constraints.append((u[-1] == x[-1]))
eps = np.random.standard_normal(size=(N, T))
B = [sigma.values * e for e in eps]
d = X*P_Arr - 0.5*perm*(X**2)
coeff = (temp * (1+sigma.values/(periodVol.values/periodVol.sum())) + 0.5*perm)
quad = (coeff.T * cvx.square(u))
C = [d+b*x - quad for b in B]

constraints.extend([[(z[i] >= Psi - C[i]) for i in range(N)])
Psi.value = P_Arr_Init*X

obj = cvx.Minimize(cvx.sum_entries(z)/N)
prob = cvx.Problem(obj, constraints)
prob.solve(solver='MOSEK')

return u.value, temp, perm, coeff

A.5.1 Static Experiments

def runStaticExp(X,T,N,outOfSampleDays,numSigs,ticData,ticSummary,periods,datesIn,
datesOut,ko,returns,days = 7):
    volumes = []
    POS_Out = []
    tic = datetime.datetime.utcnow()
    datesOutTotal = datesOut[:outOfSampleDays]
    storedResults = dict.fromkeys(datesOutTotal)
    ticTotal = ticSummary[datesOut[:outOfSampleDays]]
    ticDatesTotal = np.unique(ticTotal.index.date)
    P0_total = getOpeningPrices(ticData, ticDatesTotal, ticTotal, periods)
    Rti_total = calculateReturns(ticTotal,P0_total,periods)
values = [delayed(getP0_out)(ticData, dayOut, ticSummary[str(dayOut)].index, periods) for dayOut in datesOutTotal]
results = compute(*values, get=dask.threaded.get)
P0s_Out = list(results)
P0_out_total = pd.concat(results)
print('Done estimating total parameters in:', datetime.datetime.utcnow() - tic)

for i in range(outOfSampleDays):
    print("i:",i)
    print("-------------")
tic = datetime.datetime.utcnow()

# Get Parameters
dayOut = datesOutTotal[i]
ticIn = ticTotal[datesIn[i]:datesOut[i]]
datesIn_updating = np.unique(ticIn.index.date)
P0 = P0_total[datesIn[i]:datesOutTotal[i]].index
Rti = Rti_total[datesIn[i]:datesOutTotal[i]]
P0_out = P0_out_total[str(dayOut)]

# Estimate Volumes
totalVolumeIn = ko.loc[:datesIn_updating[-1], 'Adj. Volume'].iloc[i:]
periodVol = estimVolumePortions(datesIn_updating, totalVolumeIn, ticIn)
volumes.append(periodVol.values)

# Estimate Volatilities
sigs = [estimVolatilities(datesIn_updating, returns, Rti, periods, days, totalHorizon = 390)[0] for t in range(numSigs)]

# Optimize
values = [delayed(runOptimization)(N, T, X, P0_out[0], sig, periodVol, P0_out[0]) for sig in sigs]
results = compute(*values, get=dask.threaded.get)

# Store Results
u_vals = pd.DataFrame([result[0].T.tolist()[0] for result in results]).T
storedResults[dayOut] = u_vals

# Reset Variables
values = None
results = None
ticIn = None
totalVolumeIn = None
P0 = None

print('Iteration ',i,' time:', datetime.datetime.utcnow() - tic)
print()
return storedResults, volumes, datesOutTotal, P0s_Out
Appendix A

A.5.2 Adaptive Experiments

```python
def runDynamicExp(X,T,N,outOfSampleDays,numSigs,ticData,ticSummary,periods,
datesIn,datesOut,ko,returns,tuneParam = None):
    print(f'Running Shrinking Horizon for {outOfSampleDays} days')
    tic = datetime.datetime.utcnow()
    volumes = []
    P0s_Out = []
    X_ = X
    datesOutTotal = datesOut[:outOfSampleDays]
    storedResultsStatic = dict.fromkeys(datesOutTotal)
    storedResultsDynamic = dict.fromkeys(datesOutTotal)

    ticTotal = ticSummary[:datesOut[(outOfSampleDays)]]
    ticDatesTotal = np.unique(ticTotal.index.date)
    P0_total = getOpeningPrices(ticData, ticDatesTotal, ticTotal, periods)
    Rti_total = calculateReturns(ticTotal, P0_total, periods)
    values = [delayed(getP0_out)(ticData, dayOut, ticSummary[str(dayOut)].index,
                                      periods) for dayOut in datesOutTotal]
    results = compute(*values, get=dask.threaded.get)
    P0s_Out = list(results)
    P0_out_total = pd.concat(results)
    print(f'Done estimating total parameters in:', datetime.datetime.utcnow() - tic)

    for i in range(outOfSampleDays):
        dayOut = datesOutTotal[i]
        print(f'Day Out :', dayOut)

        # Get Parameters
        ticIn = ticTotal[datesIn[i]:datesOut[i]]
        datesIn Updating = np.unique(ticIn.index.date)
        P0 = P0_total[datesIn[i]:datesOutTotal[i]].index
        Rti = Rti_total[datesIn[i]:datesOutTotal[i]]
        P0_out = P0_out_total[str(dayOut)]

        # Estimate Volumes
        totalVolumeIn = ko.loc[:datesIn Updating[-1], 'Adj. Volume'].iloc[i:]
        periodVol = estimVolumePortions(datesIn Updating, totalVolumeIn, ticIn)
        volumes.append(periodVol.values)

        u_dynamic = []

        for j in range(T-1):
            print(f'j: {j}, j')
            print('--------------')
            tic = datetime.datetime.utcnow()
            ind = datetime.datetime.combine(dayOut, periods[j])
            P_Arr = P0_out[j]
            T_ = T - j
            if j == 0:
                if callable(tuneParam):
                    days, a, c = tuneParam(i)
```
# Set Parameters

X = X_
P_Arr_orig = P_Arr
P_Arr_Init = P_Arr_orig

lastDay = datesIn_updating[-1]
firstDay = lastDay.replace(year=lastDay.year - 5)

# Set returns window
ret = returns[str(firstDay):str(lastDay)]

# Estimate Daily GARCH Volatility

estimGARCHin, forecast = dailyGARCH(rets, '2017')
dailyvol = 0.1 * np.sqrt(forecast.variance.dropna().values[0][0])

# Calculate Volatilities

si = diurnalVariance(Rti[j:], estimGARCHin, datesIn_updating, periods)
res2, qti = intradyVariance(Rti[j:], estimGARCHin, si, datesIntraday, periods, horizon = (days*390))
simQti = [intradayVarianceForecastSimulation(res2.params, qti[-1], horizon = T, nScen = 1500)**2 for t in range(numSigs)]
intraDayVols = [a*sim+c for sim in simQti]
sigs = [np.sqrt(pd.Series(sim, index = si[j:].index)*si[j:]*dailyvol) for sim in intraDayVols]

# Optimize

values = [delayed(runOptimization)(N,T,X,P_Arr,sig,periodVol[j:], P_Arr_Init) for sig in sigs]
results = compute(*values, get=dask.threaded.get)
res = deepcopy(results)
values = None
results = None

# Store Values

u_vals = pd.DataFrame([result[0].T.tolist()[0] for result in res]).T

if j == 0:
    storedResultsStatic[dayOut] = u_vals
    estimGARCHin = estimGARCHin.append(pd.Series().set_value(datetime.datetime.combine(dayOut, datetime.time(0,0,0)), dailyvol))
p1 = P0_out[j]
else:
    p1 = ticSummary[str(dayOut)].Price.iloc[j-1]

p2 = ticSummary[str(dayOut)].Price.iloc[j]
Rti = Rti.append(pd.Series().set_value(ind, np.log(p2/p1)))

# Update values using means

u = u_vals.mean(axis = 1)[0]
u_dynamic.append(u)

X = X - u

print('Iteration {}, time:'.format(datetime.datetime.utcnow()) - tic)
if np.round(X) == 0:
    break
print()
u_dynamic.append(X)
A.5.3 Aggressively Adaptive Experiments

```python
def runAggDynamicExp(X, T, N, outOfSampleDays, numSigs, ticData, ticSummary, periods, datesIn, datesOut, ko, returns, tuneParam = None):
    print('Running Shrinking Horizon for {} days'.format(outOfSampleDays))
    tic = datetime.datetime.utcnow()
    volumes = []
    P0s_Out = []
    X_ = X
    datesOutTotal = datesOut[:outOfSampleDays]
    storedResultsStatic = dict.fromkeys(datesOutTotal)
    storedResultsDynamic = dict.fromkeys(datesOutTotal)

    ticTotal = ticSummary[:datesOut[(outOfSampleDays)]]
    ticDatesTotal = np.unique(ticTotal.index.date)
    P0_total = getOpeningPrices(ticData, ticDatesTotal, ticTotal, periods)
    Rti_total = calculateReturns(ticTotal, P0_total, periods)
    values = [delayed(getP0_out)(ticData, dayOut, ticSummary[str(dayOut)].index, periods) for dayOut in datesOutTotal]
    results = compute(*values, get=dask.threaded.get)
    P0s_Out = list(results)
    P0_out_total = pd.concat(results)
    print('Done estimating total parameters in:', datetime.datetime.utcnow() - tic)

    for i in range(outOfSampleDays):
        dayOut = datesOutTotal[i]
        print('Day Out:', dayOut)

        # Get Parameters
        ticIn = ticTotal[datesIn[i]:datesOut[i]]
        datesIn_updating = np.unique(ticIn.index.date)
        P0 = P0_total[datesIn[i]:datesOutTotal[i]].index
        Rti = Rti_total[datesIn[i]:datesOutTotal[i]]
        P0_out = P0_out_total[str(dayOut)]

        # Estimate Volumes
        totalVolumeIn = ko.loc[:,datesIn_updating[-1],'Adj. Volume'].iloc[i:]
        periodVol = estimVolumePortions(datesIn_updating, totalVolumeIn, ticIn)
        volumes.append(periodVol.values)

        u_dynamic = []

        for j in range(T-1):
            print("j:",j)
            print("------------------")
            tic = datetime.datetime.utcnow()
            ind = datetime.datetime.combine(dayOut, periods[j])
```

```
\[ P_{\text{Arr}} = P_{0\text{out}}[j] \]

\[ T_\_ = T - j \]

if \( j = 0 \):
  if callable(tuneParam):
    days, a, c = tuneParam(i)
  # Set Parameters
  X = X_
  P_{\text{Arr\_orig}} = P_{\text{Arr}}

lastDay = datesIn\_updating[-1]
firstDay = lastDay.replace(year=lastDay.year-5)
# Set returns window
rets = returns[\text{str(firstDay)}:\text{str(lastDay)}]
# Estimate Daily GARCH Volatility
estimGARCHin, forecast = dailyGARCH(rets, '2017')
dailyvol = 0.1 * np.sqrt(forecast.variance.dropna().values[0][0])

if \( P_{0\text{out}}[j] < P_{\text{Arr\_orig}} \):
  P_{\text{Arr\_Init}} = P_{\text{Arr\_orig}}
else:
  P_{\text{Arr\_Init}} = P_{0\text{out}}[j]

datesIntraday = datesIn\_updating[-(days+1):]
# Calculate Volatilities
si = diurnalVariance(Rti[j:], estimGARCHin, datesIn\_updating, periods)
res2, qti = intradyVariance(Rti[j:], estimGARCHin, si, datesIntraday, periods, horizon = (days*390))
simQti = [intradyVarianceForecastSimulation(res2.params, qti[-1], horizon = T_, nScen = 1500)**2 for t in range(numSigs)]
 intraDayVols = [a*sim+c for sim in simQti]
sigs = [np.sqrt(pd.Series(sim, index = si[j:].index)*si[j:]*dailyvol) for sim in intraDayVols]

# Optimize
values = [delayed(runOptimization)(N, T_, X, P_{\text{Arr}}, sig, periodVol[j:], P_{\text{Arr\_Init}}) for sig in sigs]
results = compute(*values, get=dask.threaded.get)
values = None
results = None

# Store Values
u_vals = pd.DataFrame([result[0].T.tolist()[0] for result in res]).T

if \( j = 0 \):
  storedResultsStatic[dayOut] = u_vals
  estimGARCHin = estimGARCHin.append(pd.Series().set_value(datetime .datetime.combine(dayOut, datetime.time(0,0,0)), dailyvol))
p1 = P_{0\text{out}}[j]
else:
  p1 = ticSummary[\text{str(dayOut)}].Price.iloc[j-1]
p2 = ticSummary[\text{str(dayOut)}].Price.iloc[j]
Rti = Rti.append(pd.Series().set_value(ind, np.log(p2/p1)))
# Update values using means
u = u_vals.mean(axis = 1)[0]

u_dynamic.append(u)

X = X - u

print('Iteration ',j,' time:',datetime.datetime.utcnow() - tic)

if np.round(X) == 0:
    break

print()

u_dynamic.append(X)

storedResultsDynamic[dayOut] = u_dynamic

return storedResultsStatic, storedResultsDynamic, volumes, P0s_Out
Bibliography


