Revisiting Optimal Execution of Portfolio Transactions: A Dynamic Programming and Reinforcement Learning Approach

Master's Thesis in Economics & Finance

Jens Pedersen

University of Copenhagen

Department of Economics

June 2023

Abstract

The measurement and management of trading costs have gained renewed interest due to the tremendous growth in the equity market and the high degree of variance fueled by institutional investors. This thesis examines the optimal execution of portfolio transactions, focusing on the trading model proposed by Almgren and Chriss (2000). Four distinct approaches to solving the trading model are compared: analytical, numerical optimisation, dynamic programming and reinforcement learning. The analytical solution replicates the efficient frontier, optimal trajectories and implementation shortfall. Numerical optimisation results are consistent with Almgren and Chriss (2000). Risk-loving agents incur higher costs for greater variance, leading to exclusion from further analysis. The results of the dynamic programming approach align with the model intuition but the methodology is infeasible for large portfolios. The reinforcement learning approach captures model dynamics and demonstrates advantages at different risk-aversion levels. Empirical and distributional examples show reinforcement learning results differ across volatility and risk-aversion level. Contributions include the analytical solution for risk-neutral traders and novel approaches distinct from existing literature.

Acknowledgements

I would like to take this opportunity to express my deep gratitude to my supervisor, Professor Stefan Voigt, for his invaluable guidance and support throughout the process of writing my Master's thesis. Furthermore, I am thankful for the wealth of knowledge and skills I have gained through the courses facilitated by Stefan Voigt.

In addition, I would like to thank the love of my life, Julie, for her unwavering love and support during my Master's degree journey. Her encouragement and belief in me have been a constant source of strength and inspiration, and our son Erik and I are grateful for her presence in our lives.

Last but not least, I would like to thank friends and family, I have such amazing people in my life, and I could not have completed this without their unwavering support.

Table of Contents

1 Introduction

The tremendous growth in the equity market in the last two decades, driven by the assets of institutional investors such as mutual and pension funds, has sparked a renewed interest in the measurement and management of trading costs[1]. These costs - often called 'executions costs' because they are associated with the execution of investment strategies that include commissions, bid-ask spreads, the opportunity cost of waiting and the price impact from trading (see Bertsimas and Lo (1998) for further discussion), and they can have a substantial impact on investment performance. For example, Perold (1988) finds that a 'hypothetical' portfolio constructed according to the Value Lines ranking[2] with weekly rebalancing from 1965-1986 outperformed the market by almost 20 per cent a year, while the actual portfolio outperformed the market by 2.5 per cent a year in the same period, where the difference arose from execution costs. This *implementation shortfall* is surprisingly large and highlights the importance of execution-cost control, particularly for institutional investors whose trades often comprise a large fraction of the average daily volume of many stocks.

Bertsimas and Lo (1998) first described the best execution strategy as a dynamic trading strategy that provides the minimum expected cost of trading over a fixed period of time. However, their strategy ignores the volatility of the temporal optimisation procedure. Almgren and Chriss (2000) propose an adjusted framework, where the minimisation of the expected cost comes with a suitable penalty for the uncertainty of said costs. In this framework, they study the variance of trading costs in optimal execution, as it fits with the intuition that a trader's utility should include risk, and thus be *optimal*. For example, when trading in a highly illiquid volatile security there are two extreme trading strategies according to Almgren and Chriss (2000), namely, to trade everything immediately at a known but high cost, or trade in equal-sized baskets over a fixed period of time at a relatively lower cost. The latter strategy has lower expected costs, but this comes at the expense of uncertainty. The authors state that this specific uncertainty is inherently subjective, as it is a function of the trader's risk tolerance. The incorporation of risk into the study of optimal execution of portfolio transactions does not come without cost. In order to produce tractable analytical results, the authors work within a framework where the price dynamics follow an arithmetic random walk with independent increments and obtain results using static optimisation procedures. Following these limitations

[1] Trading costs in terms of stocks refer to the expenses incurred when buying or selling stocks in the financial markets.

[2] Value Lines ranking system is a system that predicts the relative price performance of approximately 1,700 stocks in the U.S.

and given the renewed importance of execution costs, this thesis revisits the framework presented by Almgren and Chriss (2000) by comparing their application to a dynamic programming and reinforcement learning approach.

This thesis examines the optimal execution of portfolio transactions based on the research conducted by Almgren and Chriss (2000). The exploration of this topic involves a comparative analysis using four distinct approaches to solve the trading model. Firstly, this work supports the analytical approach by solving the trading model, wherein the optimal execution for a risk-neutral trader is explicitly derived. Secondly, this work seeks to validate the numerical optimisation results presented by Almgren and Chriss (2000) through replication. By reproducing these results, the research aims to corroborate the findings and ensure the robustness of the thesis' reliability. Furthermore, this work investigates the dynamic programming approach to tackle the trading model, presenting an alternative methodology for its resolution. By leveraging the principles of dynamic programming, this approach aims to offer novel insights and potential enhancements to the execution strategies. Lastly, this thesis adopts a reinforcement learning approach as an innovative means of solving the trading model. By harnessing the power of machine learning techniques, this approach seeks to optimise the execution process through iterative learning. By meticulously exploring these four approaches, this thesis aspires to provide a comprehensive and multifaceted understanding of the optimal execution of portfolio transactions.

The main findings are: The derived analytical solution is able to replicate the efficient frontier, optimal trajectories and the implementation shortfall given the baseline parameters following Almgren and Chriss (2000). Moreover, the reliability of the implementation of the trading model is ensured, as the results from the numerical optimisation approach are identical to the results presented by Almgren and Chriss (2000). Additionally, it is found that the risk-loving agent is inherently non-optimal, as the trader will incur higher expected costs for a higher level of variance, and thus, the analysis continues without this risk level. In the subsequent section, this work finds that the novel approach to solving the trading model using dynamic programming yields results that are aligned with the model's intuition and the preceding findings. However, it is also found that for large portfolios the dynamic programming approach is infeasible due to the curse of dimensionality. Therefore, this work provides a short analysis of the temporary market impact parameter, which shows that increasing levels of temporary market impact have a significant effect on the optimal liquidation path. Moreover, this work finds that the innovative method of solving the model using

reinforcement learning is able to accurately capture the dynamics of the trading model, and that, in terms of ex-post performance evaluation the methodology is, in fact, able to generate results that show specific advantages given different levels of risk-aversion. Moreover, the empirical results are generally aligned with the results using the baseline parameters, however, the reinforcement learning approach has different characteristics for different levels of risk-aversion, which are attributed to the hypertuning of the underlying algorithm. Lastly, the distributional analysis shows that the reinforcement learning approach is able to minimise implementation shortfall significantly, but at a cost of variance.

This thesis contributes to the current literature in several ways. Firstly, it provides an analytical solution for the risk-neutral agent, which supports the findings of Almgren and Chriss (2000). Additionally, it introduces two new approaches to solving the trading model, distinct from the methods used by Almgren et al. (2005), Guéant and Lehalle (2013), Hendricks and Wilcox (2014), and Nevmyvaka et al. (2006). Specifically, this thesis incorporates the variance of execution in the reinforcement learning reward function, unlike Hendricks and Wilcox (2014), and proposes a dynamic programming solution for minimising implementation shortfall over a fixed time horizon, unlike Nevmyvaka et al. (2006). Finally, this thesis expands empirical and distributional research on optimal execution by conducting experiments with high and low-volatility stocks and modelling price innovations using a t-distribution, building upon the work of Gatheral and Schied (2011).

The thesis is divided into seven sections. In Section 2, this work examines the literature on optimal execution strategies with a focus on the seminal work by Almgren and Chriss (2000). Section 3 contains a comprehensive overview of the trading model and the theory behind the approaches to solving the trading model, namely, analytically, numerically, dynamically and using reinforcement learning. Moreover, Section 4 delves into the various methodologies employed for solving the trading model[3]. In Section 5, this work presents the results of the various methodologies and the empirical and distributional examples. Section 6 addresses issues regarding the theoretical and methodological framework and topics for further research. Lastly, Section 7 concludes the findings of the thesis.

[3]All material, including data, code and figures can be found at https://github.com/jeopedersen/optimalexecution

2 Literature Review

The efficient execution of portfolio transactions is a critical aspect of portfolio management. In order to minimise transaction costs and maximise investment performance, portfolio managers seek to develop optimal execution strategies. This literature review aims to provide an in-depth overview of the existing research on the topic of optimal execution of portfolio transactions, with a specific focus on the seminal work by Almgren and Chriss (2000) as a foundational article for further analysis.

I. Overview of Optimal Execution Strategies

Optimal execution strategies involve the careful planning and implementation of portfolio transactions to achieve the desired objectives while minimising market impact and transaction costs. Various approaches have been proposed in the literature to tackle this complex problem, including mathematical models, empirical studies, and algorithmic trading techniques. Optimal execution strategies can be broadly categorised into two types: deterministic and stochastic. Deterministic approaches aim to find an optimal execution schedule by considering factors such as market liquidity, transaction costs, and risk constraints. They often employ mathematical optimisation techniques, such as dynamic programming and quadratic programming. On the other hand, stochastic approaches incorporate uncertainty in the execution process by modelling the price dynamics and transaction costs as stochastic processes. This allows for a more realistic representation of market conditions and enables the incorporation of risk management strategies (Bertsimas and Lo, 1998).

II. Almgren and Chriss (2000)

Almgren and Chriss (2000) present a comprehensive framework for optimal execution of portfolio transactions. They introduce the concept of a permanent and temporary market impact, where permanent impact represents the long-term effect on the price due to the transaction, and temporary impact refers to the short-term price changes during the execution period. The authors propose a mathematical model to optimise the trade-off between these impacts, considering risk constraints, liquidity constraints, and transaction costs. Hence, following the above dichotomy the Almgren and Chriss (2000) would be categorised as a deterministic approach to optimal execution. The Almgren and Chriss (2000) model provides insights into how to optimally schedule trades and determine order sizes to achieve efficient execution. Their framework has been widely adopted and extended in subsequent research. For example, Guéant and Lehalle (2013) built upon the Almgren and Chriss (2000) model by incorporat-

ing both price risk and non-execution risk, i.e., the risk of providing an order in the market, but not knowing if the order is going to be executed, as an attempt to tackle the question of using limit orders.

III. Extensions and Applications

Since the publication of Almgren and Chriss (2000)'s seminal work, several researchers have built upon their foundation and proposed various extensions and applications. Notably, Almgren and Chriss themselves have contributed to further developments in subsequent articles. For instance, Almgren et al. (2005) extend the original model to incorporate price impact dynamics and stochastic volatility, adding more realism to the execution process. In addition to extensions of the original model, researchers have explored other aspects of optimal execution. Kuno and Ohnishi (2015) conducted an empirical study to study the execution performance of the risk-averse institutional trader with constant absolute risk-aversion (CARA) type utility by using the condition of no price manipulation defined in the risk-neutral sense. From this, they derive the unique explicit optimal execution strategy by using dynamic programming. Their findings highlight that the optimal execution strategy exists in the static class. Additionally, they propose conditions necessary to compare the performance of their price models.

IV. Empirical Studies

Empirical studies have sought to validate and refine the theoretical models proposed in the optimal execution literature. By analysing real-world data, researchers have examined the impact of different execution strategies on transaction costs and performance. These studies provide valuable insights into the practical implications of optimal execution strategies in different market conditions. For example, Gatheral and Schied (2011) extend the theoretical model of Almgren and Chriss (2000) by introducing an alternative choice of risk criterion and solving the Hamilton-Jacobi-Bellman (HJB) equation explicitly to find a closed-form solution for the optimal trade execution strategy assuming that the underlying unaffected stock price process is a geometric Brownian motion. Empirically, they study the optimal trading rates in two scenarios, one where the stock price is rising and one where the stock price is falling. Their analysis suggests that an explicit closed-form solution to the optimal execution problem in Almgren et al. (2005) exists when the underlying stock price follows a geometric Brownian motion.

V. Algorithmic Trading and High-Frequency Trading

The rise of algorithmic trading and high-frequency trading (HFT) has revolutionised the execution landscape. These advanced trading technologies leverage computational power to execute trades at lightning-fast speeds, aiming to exploit market inefficiencies and reduce execution costs. Researchers have examined the impact of algorithmic trading and HFT on market quality, liquidity provision, and transaction costs, shedding light on the challenges and opportunities in the optimal execution of portfolio transactions in the era of technological advancements. For instance, Brogaard et al. (2014) analysed the impact of HFT on execution costs using high-frequency trading data. Their findings suggest that HFT can reduce transaction costs by providing liquidity and improving price efficiency. Moreover, Colliard et al. (2022) delve into the examination of Algorithmic Market-Makers (AMMs) and their pricing strategies for a risky asset in scenarios where their clients possess private information regarding the asset's payoff, employing Q-learning algorithms. The research findings indicate that AMMs exhibit a learning capability to effectively handle adverse selection by adjusting their prices in response to observed trading activity, aligning with established economic theory. However, the study also reveals that AMMs impose a mark-up on trades, surpassing the competitive price, with this mark-up diminishing as the number of AMMs operating in the market increases. This investigation proves insightful and significant in shedding light on the decision-making process of a trader operating within the framework proposed by Almgren and Chriss (2000), particularly when acting upon private information. The study distinctly underscores the significance of temporary market impact for traders, and it is plausible that the findings of Colliard et al. (2022) could potentially contribute to an expanded version of the temporary market impact function within the Almgren and Chriss (2000) framework.

VI. Market Microstructure and Liquidity Modelling

Market microstructure plays a crucial role in the optimal execution of portfolio transactions. Understanding the dynamics of price formation, order book dynamics, and liquidity provision is essential for developing effective execution strategies. Researchers have examined different aspects of market microstructure, such as order flow, bid-ask spreads, and market impact, to gain insights into optimal execution. Empirical studies by Kyle (1985) and Amihud and Mendelson (1986) have provided foundational insights into market microstructure and liquidity modelling. These studies highlight the relationship between liquidity and transaction costs, as well as the impact of trading volume on price dynamics.

VII. Machine Learning and Artificial Intelligence

Machine learning and artificial intelligence (AI) techniques have gained significant attention in the field of optimal execution. These methods leverage vast amounts of data to identify patterns, predict price movements, and optimise execution strategies. Researchers have explored the application of machine learning algorithms, such as reinforcement learning and deep learning, to enhance execution performance. The first documented large-scale empirical application of machine learning algorithms, i.e., reinforcement learning, to the problem of optimised trade execution in modern financial markets was conducted by Nevmyvaka et al. (2006). In their paper, they setup the problem as a minimisation of implementation shortfall for a buy/sell programme over a fixed time horizon with discrete time periods. Their results show that reinforcement learning could improve execution efficiency by more than 50 per cent over more traditional methodologies. Moreover, Hendricks and Wilcox (2014) extended the Almgren and Chriss (2000) framework by exploring reinforcement learning as a technique to enhance existing analytical solutions for optimal trade execution with elements from the market microstructure. Empirically, Hendricks and Wilcox (2014) found that by training a learning agent to modify a volume trajectory based on the market's prevailing spread and volume dynamics, they could improve implementation shortfall by up to 10.3 per cent on average compared to the baseline model, based on a sample of stocks and trade sizes in the South African equity market.

VIII. Regulatory Considerations

Optimal execution of portfolio transactions is subject to various regulatory considerations. Regulators aim to ensure fair and orderly markets, transparency, and investor protection. Researchers have investigated the impact of regulations, such as best execution requirements and transaction cost analysis, on execution practices and market outcomes. Empirical studies by Brogaard et al. (2017) have examined the effectiveness of regulations in the HFT market. This study showed that HFTs' activities are harmful to liquidity during periods of high volatility, which is an important element in the Almgren and Chriss (2000) framework, during extremely volatile periods.

IX. Conclusion

The optimal execution of portfolio transactions remains a crucial aspect of portfolio management. The seminal work by Almgren and Chriss (2000) has laid a solid foundation for further research in this field. Subsequent studies have expanded upon their framework, exploring various extensions, conducting empirical analyses, and considering the implications of algorithmic trading, market microstructure, and regulatory considerations. As financial markets continue to evolve, the quest for efficient execution strategies will persist, driving ongoing research in this area and providing valuable insights for portfolio managers and investors alike. Despite the extensive research conducted on the seminal work by Almgren and Chriss (2000) regarding optimal execution of portfolio transactions, there remains a gap in the literature concerning a comparative analysis of the underlying methodologies. Specifically, there is a need to examine and compare the different approaches employed for computing the optimal execution of trades, including numerical optimisation, dynamic programming, and machine learning techniques such as reinforcement learning. This gap presents an exciting opportunity to determine the effectiveness and relative merits of these methodologies, and to ascertain whether one approach outperforms the others or if they are comparable in terms of achieving optimal execution.

3 Theoretical Framework

This section delves into two overarching theoretical themes. In Section 3.1 the trading model as proposed by Almgren and Chriss (2000) is thoroughly described and can be divided into three parts (i) the price dynamics, (ii) the capture and cost and (iii) the linearity conditions of the trading model. After addressing the specifics of the trading model, this work continues into the second theoretical theme. This specific theme is divided into four parts, namely, (i) how to define the efficient frontier and the explicit construction of the optimal strategies, i.e., an analytical solution in Section 3.2, (ii) a numerical optimisation approach in Section 3.3, (iii) a dynamic programming approach in Section 3.4 and (iv) a reinforcement learning approach in Section 3.5 for solving the trading model.

3.1 The Trading Model

In this section, this work introduces the trading strategy proposed by Almgren and Chriss (2000) and shows the underlying price dynamics of the corresponding trading model. Firstly, a formal definition of the trading strategy is provided, focusing on the execution of a sell program aimed at liquidating a single security. It is important to note that the definition and results for a buy program are completely analogous to the sell program, thereby offering a comprehensive understanding of both trading scenarios. Secondly, this work delves into the description of the price dynamics inherent in the trading model. By examining the intricate relationship between various market factors and price movements, a comprehensive understanding of the dynamics that drive the trading process is established. Thirdly, the study proceeds to explore the capture and cost of the trading trajectories. In doing so, it sheds light on the mechanisms involved in capturing the desired trading outcomes and highlights the associated costs incurred throughout the execution process. Lastly, the thesis discusses the linearity conditions of the trading model. By examining the assumptions and conditions that underpin the model's linearity, a comprehensive evaluation of the model's applicability and limitations is achieved.

Overall, this section provides a comprehensive overview of the Almgren and Chriss (2000) trading strategy, encompassing its definition, the description of price dynamics, analysis of capture and cost, and an exploration of the linearity conditions within the trading model.

3.1.1 The Definition of a Trading Strategy

In the trading strategy presented by Almgren and Chriss (2000), a seller would hold a block of X units of a security that the seller would want to completely liquidate before time, T. Dividing T into N intervals of length $\tau = T/N$, and defining discrete times as, $t_k = k\tau$, for $k = 0, \ldots, N$. Almgren and Chriss (2000) define the trading trajectory to be a list $x_0, \ldots, x_N$, where x_k is the number of units that the seller holds at time, t_k. The seller's initial holding is $x_0 = X$ and liquidation at time, T, requires that $x_N = 0$. A trading trajectory can be thought of as ex-post realised trades or as a plan of how to trade a block, X, securities. Thus, a seller can specify a strategy by the trade list, $n_1, \ldots, n_N$, where $n_k = x_{k-1} - x_k$ is the number of units that the seller will sell between times t_{k-1} and t_k. Thereby, we have that

$$x_k = X - \sum_{j=1}^{k} n_j = \sum_{j=k+1}^{N} n_j, \quad k = 0, \ldots, N$$

which states that the number of units that the seller plans to hold at time t_k, is given by the initial block of units, X, minus the sum of the trades in the trade list. For notional simplicity, the model describes all time intervals of equal length, τ, however, this restriction is not essential for the model. Thus, it would be possible to define a trading strategy by the continuous-time limit $N \to \infty, \tau \to 0$. Almgren and Chriss (2000) define a trading strategy to be a rule governing the n_k in terms of the information available at time t_{k-1}, i.e., the model distinguishes two types of strategies, namely, static and dynamic strategies. Static strategies are strategies that have predefined trajectories in advance of trading, which govern the determination of each n_k and only depends on the information available at time t_0. Dynamic strategies, conversely, depend on all information up to and including time t_{k-1}.

3.1.2 The Price Dynamics of the Trading Model

In this framework, the initial market value of the position, which is given as XS_0, i.e., the initial holding multiplied by the initial security price, evolves according to two exogenous factors, namely, volatility and drift, and one endogenous factor, namely, market impact. Volatility and drift are in this framework assumed to be the result of market forces that occur randomly and are independent of the sellers trading. On the other hand, market impact denotes when market participants begin to detect volume changes as a result of selling or buying, and the participants naturally adjust their bids (offers) downward (upward). The framework distinguishes between two kinds of market impact, namely, tempo-

rary and permanent. Temporary market impact refers to the imbalances in supply and demand caused by the sellers trading that lead to temporary price movements away from equilibrium. Permanent market impact refers to the permanent changes from the equilibrium price due to the sellers trading, i.e., remains for at least the entirety of the liquidation period (Almgren and Chriss, 2000).

In this trading model, the security price evolves according to the discrete arithmetic random walk

$$S_k = S_{k-1} + \sigma\tau^{1/2}\varepsilon_k - \tau g\left(\frac{n_k}{\tau}\right), \tag{3.1}$$

for $k = 1, \ldots, N$. Here σ represents the volatility of the security, ε_k are drawn from a distribution with zero mean and unit variance and $g(v)$ is the permanent market impact, which is a function of the average rate of trading, i.e., $v = n_k/\tau$. In equation (3.1), there is no drift term, which should be interpreted as the assumption that the seller has no information about the direction of the future price movement (Almgren and Chriss, 2000)[4].

The logic behind the aforementioned temporary price impact is such that if a trader plans to sell a certain number of units, n_k, between times, t_{k-1}, and, t_k, the trader may slice these units into smaller deals to locate the optimal points of liquidity. Therefore, if the total number of units is sufficiently large the execution price may steadily decrease between t_{k-1} and t_k, which in part is due to exhausting the supply of liquidity at each successive price level. In this framework, this effect is assumed to be short-lived and, in particular, liquidity returns after each period and a new equilibrium price is established. This is modelled by introducing $h(v)$, which is the temporary drop in average price per share caused by trading at the average rate v during one time interval. Thereby, the actual price per share received on sale k is

$$\tilde{S}_k = S_{k_1} - h\left(\frac{n_k}{\tau}\right), \tag{3.2}$$

however, the effect of $h(v)$ does not appear in the next market price at S_k. The functions in equations (3.1) and (3.2), i.e., $g(v)$ and $h(v)$ can be chosen to reflect any market microstructure that is only subject to certain natural convexity conditions[5] (Almgren and Chriss, 2000).

[4]It should be noted that over long-term investment time scales or in highly volatile markets, one should consider the geometric rather than the arithmetic Brownian motion, i.e., scaling σ with S in equation (3.1). However, over short-term trading horizons, which is the interest of this trading model, the differences are negligible.

[5]See Ball (1976) for a rigorous analysis of convex functions.

3.1.3 The Capture and Cost of Trading Trajectories

In this framework profits, resulting from trading along a trajectory are defined by capture and cost. Capture is full trading revenue received upon completion of all trades along the predefined trajectory[6]. Capture is the sum of the product of the number of units that the seller can trade in each time interval multiplied by the effective price per share received on sale. This is represented as

$$\sum_{k=1}^{N} n_k \tilde{S}_k = XS_0 + \sum_{k=1}^{N} \left(\sigma \tau^{1/2} \varepsilon_k - \tau g\left(\frac{n_k}{\tau}\right) \right) x_k - \sum_{k=1}^{N} n_k h\left(\frac{n_k}{\tau}\right), \tag{3.3}$$

The first term in equation (3.3), represents the initial market value of the position and each additional term represents a gain or loss due to the market conditions. The term $\sum \sigma \tau^{1/2} \varepsilon_k x_k$ is the total effect of market variance, i.e., volatility. The permanent market impact, $-\sum \tau x_k g(\frac{n_k}{\tau})$, represents the loss, which is associated with the permanent price drop of selling a piece of the position. The temporary market impact, $\sum n_k h(\frac{n_k}{\tau})$, is the price drop due to selling in the kth period, but only affects the portion of the position sold in the kth period (Almgren and Chriss, 2000).

From equation (3.3), it is possible to deduce that the total cost of trading is given as the difference between the initial book value and the capture, i.e., $XS_0 - \sum n_k \tilde{S}_k$. This definition is equal to the ex-post measure of transaction costs used in performance evaluation, which was first mentioned by Perold (1988) as implementation shortfall. This metric serves as the primary transaction cost indicator and is minimised in order to maximise trading revenue.

In this framework, implementation shortfall prior to trading is a random variable. Thus, the expected shortfall is given as $E(x)$ and the variance is $V(x)$. From equation (3.3), the expected shortfall and the variance are given as

$$E(x) = \sum_{k=1}^{N} \tau x_k g\left(\frac{n_k}{\tau}\right) + \sum_{k=1}^{N} n_k h\left(\frac{n_k}{\tau}\right) \tag{3.4}$$

$$V(x) = \sigma^2 \sum_{k=1}^{N} \tau x_k^2 \tag{3.5}$$

The units of the expected shortfall are equal to the currency, \$, of the listed stock and the units of the variance, are the squared currency. The distribution of implementation shortfall is Gaussian if the ε_k are Gaussian; if N is sufficiently large it is in any case nearly Gaussian (Almgren and Chriss, 2000).

[6] In this framework, we do not consider any notions of carry or time value of money.

3.1.4 The Linearity Conditions of the Trading Model

In Almgren and Chriss (2000)'s framework, the optimal trajectories are computed using that the permanent and temporary impact functions are linear functions of the rate of trading[7]. From this, the linear permanent impact is given as

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

where γ is a constant with units of (\$/share)/share. With this form, each n unit the seller trades depresses the price per share by γn regardless of the time it takes to trade. This yields the following for equation (3.1)

$$S_k = S_0 + \sigma \sum_{j=1}^{k} \tau^{1/2} \varepsilon_j - \gamma(X - x_k)$$

Then summing the parts in the expected implementation shortfall, i.e., equation (3.4), the permanent impact term becomes

$$\begin{aligned} \sum_{k=1}^{N} \tau x_k g\left(\frac{n_k}{\tau}\right) = \gamma \sum_{k=1}^{N} x_k n_k \leftrightarrow \\ \gamma \sum_{k=1}^{N} x_k \left(x_{k-1} - x_k\right) = \frac{1}{2}\gamma \sum_{k=1}^{N} \left(x_{k-1}^2 - x_k^2 - (x_k - x_{k-1})^2\right) \leftrightarrow \\ = \frac{1}{2}\gamma X^2 - \frac{1}{2}\gamma \sum_{k=1}^{N} n_k^2 \end{aligned}$$

Likewise, for the temporary impact term, the form is given as

$$h(v) = h\left(\frac{n_k}{\tau}\right) = \epsilon \text{sgn}(n_k) + \frac{\eta}{\tau} n_k,$$

where sgn is the sign function[8]. The unit value of ϵ is \$/share and for η (\$/share)/(share/time).

Following Almgren and Chriss (2000), a reasonable estimate for ϵ is the fixed cost of selling, e.g., half of the bid-ask spread plus fees. However, it is difficult to estimate η, as it depends on the internal microstructures of the market. Therefore, it is especially in this term the trading model will expect non-linear effects to be most important, and thus, the approximation of the temporary market to be somewhat doubtful.

[7] The actual formulation of the trading model does not require linearity, however, it eases computation.

[8] The sign function is a function that returns the sign of a real number.

Noting that $h(v)$ is a quadratic cost model, as the total cost incurred by selling or buying n units in a single time unit is

$$nh\left(\frac{n}{\tau}\right) = \epsilon|n| + \frac{\eta}{\tau}n^2$$

With both linear cost models, i.e., the permanent and the temporary impact equations, the expectation for the impact costs becomes

$$E(x) = \frac{1}{2}\gamma X^2 + \epsilon\sum_{k=1}^{N}|n_k| + \frac{\tilde{\eta}}{\tau}\sum_{k=1}^{N}n_k^2 \tag{3.6}$$

where $\tilde{\eta} = \eta - \frac{1}{2}\gamma\tau$. Importantly, the expected impact costs, E, is a strictly convex function as long as $\tilde{\eta} > 0$. Moreover, it should be noted that if n_k all have the same sign, which is the case if it is a pure sell or pure buy programme, then $\sum|n_k| = |X|$ (Almgren and Chriss, 2000).

The most extreme trajectories of E and V with linear impact functions are: to sell at a constant rate and minimise variance with regard to transactions. The most obvious trajectory is to sell at a constant rate over the entirety of the liquidation period. Therefore, we have that

$$n_k = \frac{X}{N} \quad \text{and} \quad x_k = (N-k)\frac{X}{N}, \quad k = 1, \ldots, N \tag{3.7}$$

Following equations (3.4) and (3.5), we have that

$$\begin{aligned} E &= \frac{1}{2}XTg\left(\frac{X}{T}\right)\left(1 - \frac{1}{N}\right) + Xh\left(\frac{X}{T}\right) \leftrightarrow \\ E &= \frac{1}{2}\gamma X^2 + \epsilon X + \left(\eta - \frac{1}{2}\gamma\tau\right)\frac{X^2}{T} \end{aligned} \tag{3.8}$$

and

$$V = \frac{1}{3}\sigma^2 X^2 T\left(1 - \frac{1}{N}\right)\left(1 - \frac{1}{2N}\right) \tag{3.9}$$

This extreme trajectory minimises total expected costs, however, the variance may be large if T is long. Furthermore, we have that as the number of trading periods $N \to \infty$, $v = \frac{X}{T}$ remains finite and the expected costs and variance have finite limits (Almgren and Chriss, 2000).

The other extreme is to sell the entire position at the first step in time. Thus,

we have that

$$n_1 = X, \quad n_2 = \cdots = n_N = 0, \quad x_1 = \cdots = x_N = 0, \tag{3.10}$$

which results in

$$E = \epsilon X + \eta \frac{X^2}{\tau}, \quad V = 0 \tag{3.11}$$

This trajectory has the smallest variance, i.e., zero. If N is large and hence τ is short, then when the seller has the initial full portfolio, the price hit that the seller takes can be arbitrarily large (Almgren and Chriss, 2000).

3.2 The Efficient Frontier of Optimal Execution

To investigate the link between risk-aversion and optimal portfolio execution, Almgren and Chriss (2000) undertake a comprehensive analysis. They define and calculate optimal execution trajectories across different levels of risk-aversion, aiming to discern the interplay between these factors. Remarkably, their research reveals a singularly determined optimal execution strategy that emerges for each specified level of risk-aversion. this work extends the analysis by defining the efficient frontier and explicitly constructing optimal strategies in accordance with the framework presented by Almgren and Chriss (2000). By delineating the efficient frontier, which represents the optimal trade-off between risk- and implementation shortfall, this work sheds light on the range of viable strategies available to investors. Additionally, this work offers a meticulous exposition of the explicit methodology employed to construct optimal strategies, providing a practical and actionable approach to executing portfolio transactions based on Almgren and Chriss (2000)'s findings.

3.2.1 The Definition of the Frontier

Following Almgren and Chriss (2000) and economic intuition, a rational trader will always seek to minimise the expected implementation shortfall given the level of variance of the shortfall. A trader will prefer a strategy that provides the minimum error with regard to the estimate of expected costs. From this Almgren and Chriss (2000) define a trading strategy to be optimal if there is no strategy that has a lower variance for the same or lower level of expected transactions costs or, equivalently, no one strategy which has a lower level of expected transaction costs for the same or lower level of variance[9].

To construct efficient strategies Almgren and Chriss (2000) solve the constrained optimisation problem

$$\min_{x:V(x)\leq V_*} E(x) \tag{3.12}$$

The above constraint suggests that for a given maximum level of variance, $V_* \geq 0$, we find a strategy that has a minimum expected level of transaction costs. One should note that since $V(x)$ is convex, we have that the set, i.e., $\{V(x) \leq V_*\}$ is convex additionally since $E(x)$ is strictly convex, there exists a unique minimiser, $x_*(V_*)$.

[9]Importantly, the definition of optimality is the same for a strategy whether the strategy is dynamic of static. It is established by Almgren and Chriss (2000) that optimal strategies are in fact static regardless of the price dynamics.

Despite the traders preferred balance of risk- and return, we note that every solution, x, that has $V(x) \leq V_*$ has higher expected costs than $x_*(V_*)$, for the same and lower variance and thus, can never be optimal. Therefore, all possible optimal strategies are parameterised by a single variable, V_*, which represents all possible maximum levels of variance in transaction costs. This is exactly what Almgren and Chriss (2000) define as the efficient frontier of optimal trading strategies.

The constrained optimisation problem (3.12) can be solved by introducing the Lagrangian multiplier, λ, such that we have the following problem

$$\min_x (E(x) + \lambda V(x)) \tag{3.13}$$

From this, we can note that if $\lambda > 0$, then $E + \lambda V$ is strictly convex, and thus, $x^*(\lambda)$ has a unique solution, which varies with λ and creates the efficient frontier. Almgren and Chriss (2000) comments on the financial interpretation of λ, which they state as being a measure of risk-aversion, which is, how much the seller penalises variance relative to expected cost. Thus, for given values of λ, the constraint (3.13) can be solved by different numerical techniques depending on the functional form of $g(v)$ and $h(v)$. Almgren and Chriss (2000) solve the special case that the functions are linear.

3.2.2 The Explicit Construction of Optimal Strategies

Using equation (3.6) and equation (3.5), i.e., the expected cost and variance of the cost whilst assuming that n_j does not change sign, the combination $U(x) = E(x) + \lambda V(x)$ is a quadratic function with parameters $x_1, \ldots, x_{N-1}$ and it is strictly convex for $\lambda \geq 0$. Thereby, we have that

$$\begin{aligned}
&U(x) = E(x) + \lambda V(x) \leftrightarrow \\
&U(x) = \frac{1}{2}\gamma X^2 + \epsilon \sum_{k=1}^{N} |n_k| + \frac{\tilde{\eta}}{\tau} \sum_{k=1}^{N} n_k^2 + \lambda \left(\sigma^2 \sum_{k=1}^{N} \tau x_k^2 \right) \leftrightarrow \\
&U(x) = \frac{1}{2}\gamma X^2 + \epsilon \sum_{k=1}^{N} |\,(x_{k-1} - x_k)\,| + \frac{\tilde{\eta}}{\tau} \sum_{k=1}^{N} \left((x_{k-1} - x_k)^2 \right) + \lambda \left(\sigma^2 \sum_{k=1}^{N} \tau x_k^2 \right)
\end{aligned}$$

Thus, following Almgren and Chriss (2000), we can determine the unique global minimum by solving the first-order condition, i.e., setting its partial derivative equal to zero. First, we rewrite the expression $U(x)$ to ease the computation of

the partial derivative

$$U(x) = \cdots + \frac{\tilde{\eta}}{\tau}\sum_{k=1}^{N}\left((x_{k-1}-x_k)^2\right) + \lambda\left(\sigma^2\sum_{k=1}^{N}\tau x_k^2\right) \leftrightarrow$$
$$U(x)_j = \frac{\tilde{\eta}}{\tau}\left((x_{j-1}-x_j)^2\right) + \lambda\left(\sigma^2\tau x_j^2\right)$$

Now, we can calculate the partial derivative

$$\frac{\partial U}{\partial x_j} = 2\frac{\tilde{\eta}}{\tau}(x_{j-1}-x_j)(-1) + (x_j - x_{j+1}) + 2\lambda\sigma^2\tau x_j^2 \leftrightarrow$$
$$\frac{\partial U}{\partial x_j} = 2\tau\left(\lambda\sigma^2 x_j - \tilde{\eta}\frac{x_{j-1}-2x_j+x_{j+1}}{\tau^2}\right)$$

for $j = 1, \ldots, N-1$, we then have that $\frac{\partial U}{\partial x_j} = 0$ is equivalent to

$$\tilde{\kappa}^2 x_j = \frac{1}{\tau^2}(x_{j-1} - 2x_j + x_{j+1}) \tag{3.14}$$

where $\tilde{\kappa}^2 = \frac{\lambda\sigma^2}{\tilde{\eta}} = \frac{\lambda\sigma^2}{\eta(1-\frac{\gamma\tau}{2\eta})}$.

Following Almgren and Chriss (2000), it should be noted that equation (3.14) is a linear difference equation whose solution can be written as a combination of the exponentials $\exp(\pm\kappa\tau_j)$, where κ satisfies

$$\tilde{\kappa}^2 = \frac{2}{\tau^2}(\cosh(\kappa\tau) - 1)$$

Thus, the specific solution with $x_0 = X$ and $x_N = 0$ is a trading trajectory of the form

$$x_j = \frac{\sinh(\kappa(T-t_j))}{(\sinh\kappa T)}X, \quad j = 0, \ldots, N, \tag{3.15}$$

and with an associated trade list

$$n_j = \frac{2\sinh\left(\frac{1}{2}\kappa\tau\right)}{\sinh\left(\kappa T\right)}\cosh\left(\kappa(T - t_{j-\frac{1}{2}})\right)X, \quad j = 1, \ldots, N \tag{3.16}$$

where sinh and cosh are the hyperbolic sine and cosine functions and $t_{j-\frac{1}{2}} = (j-\frac{1}{2})\tau$. Moreover, the tildes on $\tilde{\eta}$ and $\tilde{\kappa}$ denote a $\theta(\tau)$ correction as $\tau \to 0$, then $\tilde{\eta} \to \eta$ and $\tilde{\kappa} \to \kappa$. These solutions originate from Grinold and Kahn (1990), who constructed combinations of expected excess returns and risk, i.e., an efficient frontier (Almgren and Chriss, 2000). In this framework, we have that $n_j > 0$ for each j as long as $X > 0$. This means that for a pure sell programme of an initial long position, the solution decreases monotonically from

its initial value to zero at a rate determined by the parameter κ. By definition, this would mean that the optimal execution of a sell programme never involves buying securities[10].

For a small step in time, τ, the approximate expression is given as

$$\kappa \approx \tilde{\kappa}\theta(\tau^2) \approx \sqrt{\frac{\lambda\sigma^2}{\eta} + \theta(\tau)}, \quad \tau \to 0 \tag{3.17}$$

Therefore, if the seller's trading intervals are short, κ^2 is essentially the ratio of the product of volatility and the risk-intolerance to the temporary transaction costs for the seller. The expectation and variance of the optimal strategy, for a given portfolio size, X, is thereby given as

$$E(X) = \frac{1}{2}\gamma X^2 + \epsilon X + \tilde{\eta} X^2 \frac{\tanh(\frac{1}{2}\kappa\tau)(\tau\sinh(2\kappa T) + 2T\sinh(\kappa\tau))}{2\tau^2\sinh^2(\kappa\tau)} \tag{3.18}$$

$$V(X) = \frac{1}{2}\sigma^2 X^2 \frac{\tau\sinh(\kappa T)\cosh(\kappa(T-\tau)) - T\sinh(\kappa\tau)}{\sinh^2(\kappa T)\sinh(\kappa\tau)} \tag{3.19}$$

Equations (3.18) and (3.19) reduce to equations (3.7 - 3.11) in the limits as $k \to 0, \infty$ (Almgren and Chriss, 2000).

[10] This can be the case if there is drift or serial correlation in the price movements.

3.3 A Numerical Optimisation Approach

In this section, the present study introduces sequential quadratic programming (SQP), a numerical optimisation method, as a means to address the static optimal strategies proposed by Almgren and Chriss (2000). The framework of SQP, as outlined by Judd (1998), serves as a powerful tool for solving these static optimisation problems effectively and efficiently. By incorporating SQP, this research offers a robust approach to determining the optimal strategies put forth by Almgren and Chriss (2000) in a numerical optimisation context.

3.3.1 The Theory of Sequential Quadratic Programming

Sequential quadratic programming is an iterative non-linear optimisation method, which takes advantage of the fact that there are special methods that can be used to solve problems that arise with quadratic objectives and linear constraints. SQP methods are used when the objective function and the constraints are continuous and differentiable twice[11] (Judd, 1998). The method SQP revolves around solving a sequence of optimisation sub-problems, each sub-problem optimises a quadratic model of the objective subject to a linearisation of the constraints. In the case of an unconstrained problem, the SQP formulation reduces to the well-known Newton-Raphson method. The Newton-Raphson method is a root-finding algorithm that seeks to locate points where the gradient of the objective function becomes zero, i.e., $f'(x) = 0$. By utilising second derivatives, SQP can efficiently handle unconstrained optimisation problems. When the problem includes equality constraints, the SQP method aligns with the Karush-Kuhn-Tucker (KKT) conditions. The KKT conditions are a set of first derivative tests used in nonlinear programming problems. By satisfying these conditions, SQP ensures that the solution satisfies both the objective function optimisation and the equality constraints. The key advantage of SQP lies in its ability to handle quadratic objectives and linear constraints efficiently. The quadratic models provide a local approximation of the objective function, capturing the curvature information necessary for effective optimisation. The linearisation of constraints enables the use of linear programming techniques, simplifying the optimisation process (Judd, 1998).

[11] A continuous function is not necessarily differentiable, but a differentiable function is necessarily continuous.

The sequential quadratic method solves the quadratic problem given as

$$\min_{x} \quad f(x) \tag{3.20}$$

$$\text{s.t} \quad h(x) \geq 0 \tag{3.21}$$

$$g(x) = 0 \tag{3.22}$$

The function (3.20) is to be minimised, the constraint (3.21) represents the equality constraints and the constraint (3.22) represents the inequality constraint. The Lagrangian for this problem is then defined as

$$\mathcal{L}(x, \lambda, \sigma) = f(x) - \lambda h(x) - \sigma g(x) \tag{3.23}$$

where λ and σ are Lagrange multipliers (Judd, 1998). In SQP, the Newton-Raphson method searches for the solution by iterating the following equation

$$\begin{bmatrix} x_{k+1} \\ \lambda_{k+1} \\ \sigma_{k+1} \end{bmatrix} = \begin{bmatrix} x \\ \lambda \\ \sigma \end{bmatrix} = \underbrace{\begin{bmatrix} \nabla^2_{xx}\mathcal{L} & \nabla h & \nabla g \\ \nabla h^T & 0 & 0 \\ \nabla g^T & 0 & 0 \end{bmatrix}^{-1}}_{\nabla^2\mathcal{L}} = \underbrace{\begin{bmatrix} \nabla f + \lambda_k \nabla h + \sigma_k \nabla g \\ h \\ g \end{bmatrix}}_{\nabla\mathcal{L}}$$

where ∇ is the gradient to the respective function. It should be noted that the matrix $\nabla^2\mathcal{L}$ is often non-invertible, i.e., singular, the Newton-Raphson step $d_k = (\nabla^2\mathcal{L}^{-1}\nabla\mathcal{L})$ cannot be computed directly. Thus, the SQP algorithm defines an appropriate search direction d_k at repeated $(x_k, \lambda_k, \sigma_k)$, as a solution for the SQ subproblem

$$\min_{d} \quad f(x_k) + \nabla f(x_k)^T d + \frac{1}{2} d^T \nabla^2_{xx}\mathcal{L}(_k, \lambda_k, \sigma_k) d \tag{3.24}$$

$$\text{s.t} \quad h(x_k) + \nabla h(x_k)^T d \geq 0 \tag{3.25}$$

$$g(x_k) + \nabla g(x_k)^T d = 0 \tag{3.26}$$

From this, the SQP algorithm starts by choosing the initial iteration, i.e., $(x_0, \lambda_0, \sigma_0)$, then computing $\nabla^2\mathcal{L}(x_0, \lambda_0, \sigma_0)$ and $\nabla\mathcal{L}(x_0, \lambda_0, \sigma_0)$. Furthermore, the quadratic programming subproblem is built and solved in order to find the next Newton-Raphson step direction, i.e., d_0, which is then used to update the parent problem iteratively using that $[x_{k+1}, \lambda_{k+1}, \sigma_{k+1}]^T = [x_k, \lambda_k, \sigma_k]^T + d_k$. This process is repeated for $k = 0, 1, 2, \ldots, K$ until the parent problem satisfies a convergence test (Judd, 1998).

3.4 A Dynamic Programming Approach

In this section, this work introduces the theory of dynamic programming, a powerful tool for addressing dynamic optimisation problems. The origins of dynamic programming can be traced back to the seminal works of Bellman (1954) and Bertsekas (1976), who laid the foundation for its application. Economists have also made valuable contributions to the field, with Sargent (1987) and Stokey and Lucas (1989) shedding light on dynamic programming's relevance in the realms of economics and finance. Moreover, this section provides a concise overview of a Markov Decision Process (MDP), widely employed in economic modelling. MPDs enable decision-making in the face of uncertainty over a predefined time horizon. Notably, Rust (1994) offers a comprehensive survey of the literature on MPDs, emphasising its significance in various economic contexts. Furthermore, this study delves into solving the Bellman equation, a central aspect of dynamic programming. By describing the techniques to solve the Bellman equation, this research provides a practical understanding of how dynamic programming can be applied to address complex optimisation problems effectively.

Overall, this section acquaints readers with the fundamental concepts of dynamic programming, explores its relevance in economics and finance, introduces the concept of Markov decision processes, and shows the process of solving the Bellman equation.

3.4.1 The Theory of Dynamic Programming

The theory of dynamic programming was created in order to treat mathematical problems that arose from the study of various multi-stage decision processes. A multi-stage decision process can be described as a physical system whose state at any time, t, is determined by a set of quantities which are known as parameters, or state variables. These parameters may be determined in advance or determined by the process itself. These decisions are equivalent to transformations of the state variables, i.e., the choice of a decision is identical to the choice of a transformation. The outcome of the preceding decisions is then used to guide the choice of future decisions, with the sole purpose of the whole process to maximise some function of the parameters describing the final state (Bellman, 1954). In the literature, there are many examples of this specific process fitting, e.g., the planning of industrial production lines to the optimal asset allocation over the expected lifetime of a household (Lusardi et al., 2017).

3.4.2 Markov Decision Processes in Dynamic Programming

As mentioned before, dynamic programming can be used to study various multi-stage decision processes, one of these multi-stage decision processes is a Markov Decision Process. A MDP provides a broad framework for modelling sequential decision-making under uncertainty. A MDP is a given set of states, i.e., a Markov chain with rewards[12] for each potential state and action. More specifically, MPDs have two sorts of variables, namely, state variables, s_t, and control variables, d_p, both of which are indexed by time, $t = 0, 1, 2, \ldots, T$, where this horizon may run to infinity. A decision-maker or agent can be represented by a set of conditions, e.g., (u, p, β), where $u(s_t, d_p)$ is a utility function representing the agent's preferences at time, t, $p(s_{t+1}|s_t,d_t)$, is a Markov transition probability representing the agent's subjective beliefs about the uncertainties of the future states and $\beta \in (0,1)$ is the rate at which the agent discounts utility in the future periods. Agents are assumed to act rational, i.e., they have to behave according to an optimal decision rule, $d_t = \delta(s_t)$ that solves $V_0^T(s) \equiv \max_\delta E_s\{\sum_{t=0}^T \beta^t u(s_t,d_t)|s_0 = s\}$ where E_s denotes the expectation with respect to the stochastic process $\{s_t, d_t\}$ that is induced by the decision rule δ (Rust, 1994). Dynamic programming provides a constructive procedure for computing δ using a value function V_0^T, as a way to decentralise a complicated stochastic and multi-stage optimisation problem into a sequence of simpler deterministic or static optimisation problems.

In order to solve a MDP using dynamic programming for both a finite-horizon and an infinite-horizon problem, this work briefly describes the numerical methods used to solve these functional equations referring to Rust (1994) for a more in-depth review.

Definition 3.1
A discrete Markovian decision process consists of the following objects:

- *A time index* $t \in \{0,1,2,\ldots,T\}, \quad T \leq \infty$;
- *A state space* S;
- *A decision space* D;
- *A family of constraint sets* $\{D_t(s_t) \subseteq D\}$;
- *A family of transition probabilities* $\{p_{t+1}(\circ|s_t, d_t) : \mathcal{B}(S) \to [0,1]\}$[13];

[12] Often referred to as gains or losses.
[13] $\mathcal{B}$ is the Boral σ-algebra of measurable subsets of s, as Boral algebra is outside the scope of this work see Rust (1994) for a statement required regarding the regularity conditions for this problem.

- *A family of discount functions* $\{\beta_t(s_t,d_t) \geq 0\}$ *and single period utility functions* $\{u_t(s_t,d_t)\}$ *such that the utility functional* U *has the additive separable decomposition*

$$U(s,d) = \sum_{t=0}^{T} \left[\prod_{j=0}^{t-1} \beta_j(s_j,d_j) \right] u_t(s_t,d_t). \tag{3.27}$$

From the above, the agent's optimisation problem is to choose an optimal decision $\delta^* = (\delta_0, \ldots, \delta_T)$ to solve the following form

$$\max_{\delta=(\delta_0,\ldots,\delta_T)} E_\delta\{U(s,d)\} \tag{3.28}$$

In finite-horizon problems ($T < \infty$), the optimal decision can be computed using backward induction starting at the terminal period, T, i.e., the last period. Principally, the optimal decision at each time, t, can depend not only on the current state, s_t, but also on the entire previous history of the process, i.e., $d_t = \delta_t^T(s_t, H_{t-1})$, where $H_t = (s_0,d_0, \ldots, s_{t-1}, d_{t-1})$. However, using backward induction it becomes evident that the Markovian structure of the probability, p, and the additive separability of the utility, U, imply that the previous entire previous history is obsolete, as the optimal decision rule depends only on the current time, t, and the current state $s_t : d_t = \delta_t^T(s_t)$ (Rust, 1994).

To show that the optimal decision rule depends only on the current time, it can be shown that

$$\delta_T(H_{t-1},s_T) = \underset{d_T \in D_T(s_T)}{\operatorname{argmax}} U(H_{T-1}, s_T, d_T), \tag{3.29}$$

where U can be rewritten as

$$\begin{aligned} U(H_{T-1},s_T,d_T) &= \sum_{t=0}^{T} \left(\prod_{j=0}^{t-1} \beta_j\left(s_j, d_j\right) \right) u_t(s_t, d_t) \\ &= \sum_{j=0}^{T-1} \left(\prod_{j=0}^{t-1} \beta_j(s_j, d_j) \right) u_t(s_t, d_t) + \left(\prod_{j=0}^{T-1} \beta_j(s_j, d_j) \right) u_t(s_t, d_t) \end{aligned} \tag{3.30}$$

From equation (3.30), it is evident that the previous history, i.e., H_{t-1} does not affect the optimal decision of d_T in equation (3.29), as d_T only appears in the final term $u_T(s_t,d_T)$ on the right-hand side of equation (3.30). Furthermore, since H_{t-1} only affects final term by the multiplicative discount factor,

$\prod_{j=0}^{t-1} \beta_j(s_j, d_j)$, it then clear that δ_T depends only on s_T. Conclusively, the optimal decision at time, t, is then verified to only depend on the given state s_t. This specific decision rule, which states that the decision rule depends on the past history of the entire process only through the current state, s_t, is called *Markovian*. Additionally, it should be noted that the optimal decision rule is often a deterministic function of s_t, as randomisation can only reduce the expected utility if the optimal value of d_t in equation (3.29) is unique (Rust, 1994).

In terms of defining the value, we turn to the *value function*, which is a function that defines the expected discounted value of utility over the horizon assuming an optimal policy is followed in the future. In dynamic programming, the method of computing the value function is done recursively through the optimal decision rule and where the terminal period, V_T^T, and the optimal decision rule, δ_T are defined by

$$\delta_T^T(s_T) = \underset{d_T \in D_T(s_T)}{\operatorname{argmax}} U(s_T, d_T), \tag{3.31}$$

$$V_T^T(s_T) = \max_{d_T \in D_T(s_T)} U(s_T, d_T) \tag{3.32}$$

In the periods $t = 0, \ldots, T-1$, the value function and the optimal decision rule are recursively defined by

$$\delta_t^T(s_t) = \underset{d_t \in D_t(s_t)}{\operatorname{argmax}} \left(u_t(s_t, d_t) + \beta_{t-1}(s_{t-1}, d_{t-1}) \int V_{t+1}^T[s_{t+1}, \delta_{t+1}^T(s_{t+1})] p_{t+1}(ds_{t+1}|s_t, d_t) \right), \tag{3.33}$$

$$V_t^T(s_t) = \max_{d_t \in D_t(s_t)} \left(u_t(s_t, d_t) + \beta_{t-1}(s_{t-1}, d_{t-1}) \int V_{t+1}^T[s_{t+1}, \delta_{t+1}^T(s_{t+1})] p_{t+1}(ds_{t+1}|s_t, d_t) \right) \tag{3.34}$$

From equations (3.31) and (3.32), it is evident that at time $t = 0$ the value function represents the conditional expectation of utility over all future periods, and as, dynamic programming has recursively generated the optimal decision rule, δ^*, it follows that

$$V_0^T(s) = \max_{\delta} E_\delta \{ U(\tilde{s}, \tilde{d}) | s_0 = s \} \tag{3.35}$$

Following Rust (1994), the above results can be formalised as follows

Theorem 3.1

Given an MDP that satisfies certain weak regularity conditions (see Rust (1994)):

- *An optimal, non-randomised decision rule δ^* exists,*
- *An optimal decision rule can be found within the subclass of non-randomised Markovian strategies,*
- *In the finite-horizon case, i.e., $(T < \infty)$ an optimal decision rule, δ^*, can be computed by backward induction according to equations, i.e., recursions, (3.31) to (3.34),*
- *In the infinite-horizon case, i.e., $(T = \infty)$ an optimal decision rule, δ^*, can be approximated arbitrarily closely by the optimal decision rule, δ_N^*, to an N-period problem, which can be defined as*

$$\lim_{N\to\infty} E_{\delta_N^*}\{U_N(\tilde{s},\tilde{d})\} = \lim_{N\to\infty} \sup_{\delta} E_{\delta}\{U_N(\tilde{s},\tilde{d})\} = \sup_{\delta} E_{\delta}\{U_N(\tilde{s},\tilde{d})\} \tag{3.36}$$

From the above, an important note is that for the finite horizon problem, dynamic programming is equivalent to backward induction. Furthermore, one should note that equations (3.33) and (3.34) can be recursively reduced to

$$\delta(s) = \underset{d\in D(s)}{\operatorname{argmax}} \left[u(s,d) + \beta \int V(s')p(ds')|s,d) \right], \tag{3.37}$$

$$V(s) = \max_{d\in D(s)} \left[u(s,d) + \beta \int V(s')p(ds')|s,d) \right] \tag{3.38}$$

as the Markovian structure of the problem implies that the optimal decision rule and the corresponding value function are time-invariant. Furthermore, (3.38) is commonly referred to as the *Bellman Equation* and dynamic programming uses the value function $V_t(s)$, as a shadow price, i.e., a measure of sensitivity to the optimal value, in order to decentralise a complicated multi-stage optimisation problem into a sequence of simpler static optimisation problems. Lastly, it is shown that the decision at time, t, i.e., d_t, can depend on the entire history, but it is time separable in utility and the Markovian property of p imply that $d_t = \delta_t(s_t)$ (Rust, 1994).

3.4.3 The Solution for the Bellman Equation

To establish the existence of a solution to (3.38), i.e., Bellman's equation, we must assume the following regularity conditions: (i) $u(s,d)$ is jointly continuous

and bounded in (*s*,*d*) and (ii) $D(s)$ is a continuous correspondence[14]. Let $C(S)$ denote the vector space of all continuous bounded functions $f : S \to R$ under the supremum norm, $||f|| = \sup_{s \in S} |f(s)|$ i.e., the greatest absolute value of s in f for the finite vector space. Then $C(S)$ is a Banach space, i.e., a complete normed linear space[15]. From this, we can define an operator $\Gamma : C(S) \to C(S)$ by

$$\Gamma(W)(s) = \max_{d \in D(s)} \left[u(s,d) + \beta \int V(s')p(ds')|s,d) \right] \tag{3.39}$$

Equation (3.39) can be rewritten using operator notation as

$$V = \Gamma(V) \tag{3.40}$$

i.e., V is a fixed point of the mapping Γ. Furthermore, using the argument that

$$\begin{aligned} |V(s) - W(s)| &\leq \beta \int \max_{d \in D(s)} |V(s') - W(s')|p(ds')|(s,d) \\ &\leq \beta \sup_{s \in S} |V(s) - W(s)| \end{aligned} \tag{3.41}$$

and that since $0 < \beta < 1$, the only solution to equation (3.41) is $\sup_{s \in S} |V(s) - W(s)| = 0$, it is evident that for any given $V, W \in C(S)$, we have that

$$||\Gamma(V) - \Gamma(W)|| \leq \beta ||V - W|| \tag{3.42}$$

The operator that satisfies inequality (3.41) for some $\beta \in (0,1)$ is said to be a *contraction mapping* (Rust, 1994). Following Rust (1994) the contraction theorem is defined as

Theorem 3.2
If Γ *is a contraction mapping on a Banach space* B*, then* Γ *has a unique fixed point* V*.*

The unique fixed point is established through an argumentation similar to the inequality (3.41). This solution is a result of the completeness of the Banach space. Additionally, starting at any initial element of B, e.g., 0, the contraction property (3.42) implies that the following sequence of successive approximations forms of a Cauchy sequence in B

$$\{0, \Gamma(0), \Gamma^2(0), \Gamma^3(0), \dots, \Gamma^n(0), \dots\} \tag{3.43}$$

[14] For a formal definition of jointly continuous and bounded functions and correspondences see Horváth and Kokoszka (2012).

[15] A space is said to be complete if every Cauchy sequence converges to a point in said space.

Since the Banach space, B, is complete, the Cauchy sequence[16] converges to a point $V \in B$, so existence follows by showing that V is fixed point Γ. This is noted, as

$$V = \lim_{n\to\infty} \Gamma^n(0) = \lim_{n\to\infty} \Gamma[\Gamma^{n-1}(0)]\Gamma(V) \tag{3.44}$$

Thus, V is the required fixed point of Γ (Rust, 1994).

Moreover, it is important to show - in order to solve the Bellman equation - that given the single period decision rule, δ, defined in equation (3.33), the stationary and infinite-horizon policy δ^* does in fact constitute an optimal decision rule for the infinite-horizon MDP. This result follows by showing that the unique solution the optimal value, $V(s)$ to Bellman's equation coincides with the optimal value function V_0^∞ defined by

$$V_0^\infty(s) \equiv \max_\delta E_\delta \left\{ \sum_{t=0}^{\infty} \beta^t u(s_t,d_t) | s_0 = s \right\} \tag{3.45}$$

Considering that approximating the infinite-horizon problem by the solution to a finite-horizon problem with value function

$$V_0^\infty(s) \equiv \max_\delta E_\delta \left\{ \sum_{t=0}^{T} \beta^t u(s_t,d_t) | s_0 = s \right\} \tag{3.46}$$

Since the utility, u, is bounded and continuous, $\sum_{t=0}^{T} \beta^t u(s_t,d_t)$, converges to $\sum_{t=0}^{\infty} \beta^t u(s_t,d_t)$ for any sequence, $s = (s_0 m, s_1, \dots)$ and $d = (d_0, d_1, \dots)$. Theorem (3.1) implies that for each $s \in S$, the finite-horizon value function, $V_0^T(s)$ converges to the infinite-horizon value function, $V_0^\infty(s)$

$$\lim_{T\to\infty} V_0^T(s) = V_0^\infty(s) \quad \forall s \in S \tag{3.47}$$

Furthermore, Theorem (3.1), i.e., the contraction mapping theorem, also implies that this sequence converges to V (since $V_0^T = \Gamma^T((0))$, so $V = V_0^\infty$. As V is the expected discounted value of utility under the optimal policy rule, the fact that $V = V_0^\infty$ immediately implies the optimality of δ^* (Rust, 1994). Similarly, a result can be proved under weaker conditions that allow $u(s,d)$ to be an unbounded function of the state variables. Unbounded utility functions arise in MDP problems, as a consequence of assumptions about the distribution of said state variables[17]. Thus, the contraction mapping theorem is no longer directly

[16] A Cauchy sequence is a sequence whose elements become arbitrarily close to each other as the sequence progresses.

[17] See Rust (1994) for unbounded utility functions.

applicable, instead, it has been proven through Blackwell's theorem that

Theorem 3.3
Given an infinite-horizon, time-homogeneous MDP that satisfies certain regularity conditions (see Rust (1994))

- *A unique solution V to Bellman's equation (3.38) exists, and it coincides with the optimal value function defined in (3.45),*
- *There exists a stationary, non-randomised, Markovian optimal control δ^* given by the solution to (3.37),*
- *There is optimal non-randomised, Markovian decision rule δ^*, which can be approximated by the solution δ_N^* to an N-period problem with utility function $U_N(s,d) = \sum_{t=0}^{N} \beta^t u(s_t,d_t)$:*

$$\lim_{N\to\infty} E_{\delta_N^*} = \lim_{N\to\infty} \sup_{\delta} E_{\delta}\{U_N(\tilde{s},\tilde{d})\} = \sup_{\delta} E_{\delta}\{U(\tilde{s},\tilde{d})\} = E_{\delta}^*\{U(\tilde{s},\tilde{d})\} \tag{3.48}$$

From theorem (3.3), i.e., Blackwell's theorem, it is shown that there exists a unique stationary Markovian decision rule, $\delta(s)$, and value function $V(s)$ defined as in equations (3.37) and (3.38) that solves the associated MDP problem (Rust, 1994).

3.5 A Reinforcement Learning Approach

In this section, this work describes the theory regarding the concept of reinforcement learning (RL), as a means for solving optimisation problems - and how RL differs from other machine learning algorithms. The history of RL has two main threads, both of which are long and rich, which were pursued independently before intertwining into modern RL. The first thread runs through some of the earliest work in terms of trial and error, originating in the psychology of animal learning, and the second concerns the problem of optimal control and its solution using value functions and dynamic programming. Thus, the origins of RL can be traced back to the seminal works of Bellman (1954) and Bertsekas (1976). Moreover, this section provides a concise overview of the elements of RL in MDPs, i.e., the agent-environment interaction, rewards and returns, state-value functions and policies and the optimality of these functions and policies. Lastly, this thesis describes the Deep Deterministic Gradient Policy (DDPG) algorithm, which is an algorithm that provides a practical and effective solution to optimisation to complex optimisation problems.

Overall, this section acquaints the reader with a fundamental understanding of the concept of RL and explores its elements in the MDP universe and describes the process of the DDPG algorithm.

3.5.1 The Theory of Reinforcement Learning

Reinforcement learning is a computational approach to learning what to do - how to map certain situations into actionable decisions - as to maximise a numerical reward signal. An important element of RL is that the learner, i.e., the agent responsible for learning and improving its decision-making capabilities, is not told which actions to take, but instead must discover which actions yield the most reward by trying them. In many cases, actions may not affect - not only the immediate reward - but also the next situation and, through that, all subsequent rewards. These two characteristics, i.e., trial-and-error search and delayed reward - are the two most important components in RL (Sutton and Barto, 2018).

There are many different types of learning algorithms, however, they all follow a process, which is depicted in Figure 1, i.e., there is an agent, an action, an environment, a reward and a state. A learning agent must be able to sense the environment and be able to take actions that affect the state. Furthermore, more, the agent must also have goals that relate to the state of the environment. Following the logic of MPDs introduced in Section 3.4, i.e., that the aspects of

a MDP include, state, action and reward, an MDP is a perfect problem to be solved through RL (Sutton and Barto, 2018).

Reinforcement learning differs from *supervised learning*, which is most prevalent in most current research in the field of machine learning. Supervised learning is learning from a training set of labelled examples provided by a knowledgeable external supervisor. The object of supervised learning is for the system to extrapolate, or generalise, its responses so that it acts in accordance with the situation not present in the training set. This type of learning is important, but not adequate in terms of learning from interaction. In interactive problems, it is often impractical to obtain examples of desired behaviour that are both correct and representative of all the situations in which the learner has to act. For example, in uncharted territory, where one would expect learning to be most beneficial, an agent must be able to learn from their own experience (Sutton and Barto, 2018). Additionally, RL also differs from *unsupervised learning*, which is related to finding a structure hidden in a collection of unlabelled data. RL differs from unsupervised learning, as the agent tries to maximise a reward instead of trying to find a hidden structure in the data. Uncovering structure in itself is not a RL problem with regard to maximising a reward signal, and thus, RL is defined as a third machine learning paradigm, alongside supervised learning and unsupervised learning (Sutton and Barto, 2018).

3.5.2 The Elements of Reinforcement Learning in Markov Decision Processes

I. Agent-environment interaction

In accordance with the RL terms in MDPs, the algorithm being taught is referred to as an agent, and the environment comprises every piece of knowledge that can be acted upon by the agent. In some cases, the agent has all possible information about the environment, i.e., the agent can assess the given environment with full information, e.g., in a tic-tac-toe game. In other cases, the agent does not have full information, e.g., optimal execution of a portfolio given how the market reacts. The agent interacts with the environment that the agent is placed in, and in each discrete time step, $t = 0, 1, 2, \ldots, T$, the agent, in turn, gets a representation of the environment state, $S_t \in S$, and thereby, choose an action, $A_t \in A$, based on the state. From this, a numerical reward is awarded to the agent, which the agent attempts to maximise over the course of the learning period. This process is depicted in Figure 1 (Sutton and Barto, 2018).

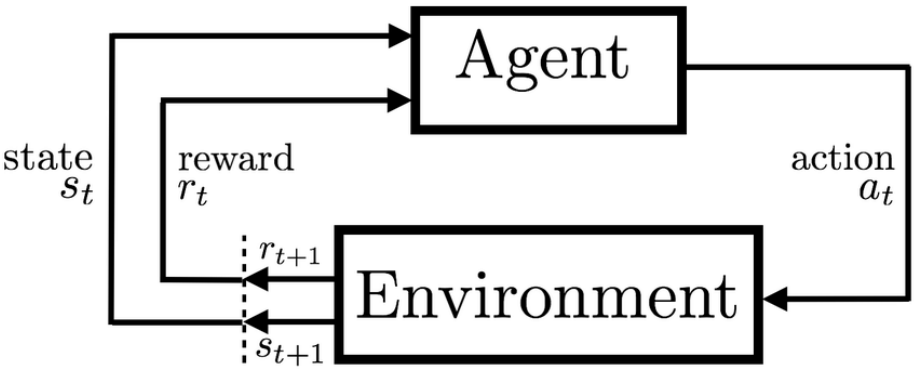

Figure 1: The agent–environment interaction in a Markov decision process (Sutton and Barto, 2018).

II. Rewards and returns

Moreover, an element of RL in MDPs is the reward, which is the primary basis for changing the decision of the agent. For example, if the agent receives a negative numerical reward for an action it can change a different action the next time it encounters the same state. This means that at each period, t, the agent is rewarded with a real number, $R_t \in R$. The agent will then maximise the cumulative rewards given as

$$G_t = R_{t+1} + R_{t+2} +, \ldots, R_t = \sum_{k=t+1}^{T} R_k \tag{3.49}$$

This cumulative reward structure is the standard approach in terms of agent-environment formulation, which is a natural stopping point at time, t, i.e., a finite horizon. Some agent-environment formulations may have that $T = \infty$, which would require some kind of discounting in order to solve for the optimal solution (for more see Sutton and Barto (2018)).

III. State-value functions and policies

The state-value function determines what actions are good in the long run, whereas the rewards as mentioned before only indicate what is good at the given time period, t. Future rewards are dependent on the actions that the agent undertakes throughout the entire learning period, and thus, the value functions are defined with regard to *policies*. A policy in RL, is a mapping, $\pi : S \times A \rightarrow [0,1]$, from states and actions to probabilities of each action, i.e., it thereby determines the agent's behaviour at all times. The agent's policy can be determined from some distribution of actions over states or from a lookup in a table, i.e., a table in which the maximum expected future rewards at each state

are computed. The state-value function under the policy, π, is the expected return of taking an action, a, in state, s, and following the policy, π

$$\begin{aligned} q_\pi(s,a) &= E_\pi[G_t|S_t s, A_t = a] \\ &= E_\pi\left[\sum_{k=0}^{T} R_{t+k+1}|S_t, = s, A_t = a\right] \end{aligned} \tag{3.50}$$

Equation (3.50) is denoted as the q_π action-value function for the policy, π (Sutton and Barto, 2018).

IV. Optimal functions and policies

Solving RL problems requires finding a policy, π, that maximises the cumulative reward described in equation (3.49). The optimal policy is defined by the action-value function and one optimal policy is then defined to be equal to or better than π' if the expected return is greater for π in all states and actions. From this, it can be defined that if $\pi \geq \pi'$ if $q_\pi \geq q'_p i$ for all $s \in S$ and $a \in A$, there is at least one policy that is better than or equal to all other policies, which is exactly the *optimal policy*. However, there may be more than one optimal policy, but they are all denoted by π_*. The optimal policies share the same optimal action-value function denoted q_*, and defined as

$$q_*(s,a) = \max_\pi q_\pi(s,a) \tag{3.51}$$

for all $s \in S$ and $a \in A(s)$. For the state-action pair, (s,a), this function gives the expected return for taking action, a, in state, s, and thereafter following an optimal policy.

3.5.3 Deep Deterministic Policy Gradient Agents

In the section above, it was established how an agent conducts observations and takes actions within an environment and in return it receives rewards. However, there are several types of RL algorithms that include the aforementioned elements in order to maximise the agent's long-term expected reward. These algorithms can be divided into three groups, namely, (i) Critic-Only, (ii) Actor-Only and (iii) Actor-Critic[18]. (i) is also known as the value-based method, i.e., it first finds the optimal value function and then derives an optimal policy from it. (ii) also known as the policy-based method searches directly for the optimal policy in the policy space. This is done by parameterising the family of policies over which the optimisation procedures can be used directly. (iii) combines the

[18]Critics - for a given observation and action, a critic returns the predicted discounted value of the cumulative long-term reward.

advantages of (i) and (ii), as in this method the critic learns the value function and uses it to determine how the agent's policy parameters should be changed throughout the learning period. In this case, the agent brings the advantage of computing continuous actions without the need for optimisation procedures on the value function, as the environment supplies the agent with knowledge of the agent's performance. The Actor-Critic method usually has good convergence properties, in contrast to Critic-Only methods (Sutton and Barto, 2018).

The Deep Deterministic Policy Gradients (DDPG) algorithm is one example of an Actor-Critic method. The DDPG algorithm is a model-free, off-policy[19] RL learning method. A DDPG agent is an agent that searches for an optimal policy that maximises the expected cumulative long-term reward. DDPG agents can be trained in environments with continuous or discrete observation spaces and continuous action spaces. Furthermore, during training a DDPG agent (i) updates the actor and critic properties at each time step, t, during learning, (ii) stores past experiences using a circular experience buffer[20]. The agent updates the actor and critic using a mini-batch of experiences randomly sampled from the buffer. (iii), the algorithm perturbs the action chosen by the policy using a stochastic noise model at each training step (Lillicrap et al., 2016).

[19] Off-policy learning algorithms evaluate and improve a policy that is different from the policy that is used for the action selection.

[20] An array of constant length used to store data in a continuous loop.

I. Actor and Critic Functions

To estimate the policy and value function, DDPG maintains four function approximators

Function Approximator	**Parameters**
Actor $\pi(S;\theta)$	The actor, with parameters θ, takes observation S and returns the corresponding action that maximises the long-term reward.
Target Actor $\pi_t(S;\theta_t)$	To improve stability, the agent periodically updates the target actor parameters θ_t using the latest actor parameter values.
Critic $Q(S,A;\phi)$	The critic, with parameters ϕ, takes observation S and action A as inputs and returns the corresponding expectation of the long-term reward.
Target Critic $Q_t(S,A;\phi_t)$	To improve stability, the agent periodically updates the target critic parameters ϕ_t using the latest critic parameter values.

Table 1: Function Approximators in DDPG (Lillicrap et al., 2016).

II. Training Algorithm

According to Lillicrap et al. (2016)), DDPG agents use the following training algorithm, where they update their actor and critic models at each time step:

- Initialise the critic, $Q_t(S,A;\phi_t)$,with random parameter values, ϕm and initialise the target critic parameters, ϕ_t with the same values, i.e., $\phi_t = \phi$,
- Initialise the actor, $\pi_t(S;\theta_t)$, with random parameter values θ, and initialise the target actor parameters θ_t with the same values, i.e., $\theta_t = \theta$,
- For each training time step:
    1. For the current observation, S, select an action $A = \pi(S;\theta) + N$, where N is stochastic noise from the noise model.
    2. Execute action, A, and observe the reward, R, and the next observation, S',
    3. Store the experience (S, A,R,S') in the experience buffer.
    4. Sample a random mini-batch of M experiences (S_i, A_i, R_i,S'_i) from the experience buffer.

5. If S_i' is a terminal state, set the value function target y_i to R_i. Otherwise, set it to

$$y_i = R_i + yQ_t(S_i', \pi_t(S_i'; \theta_t); \phi_t)$$

The value function target is the sum of the experience reward, R_i, and the discounted reward. To compute the cumulative reward, the agent first computes the next action by passing the next observation, S_i'm from the sampled experience to the target actor. The agent finds the cumulative reward by passing the next action to the target critic,

6. Update the critic parameters by minimising the loss, L, across all sampled experiences

$$L = \frac{1}{2M} \sum_{i=1}^{M} \left(y_i - Q(S_i, A_i; \phi)\right)^2 \tag{3.52}$$

7. Update the actor parameters using the following sampled policy gradient to maximise the expected discounted reward

$$\begin{aligned} \nabla_\theta &\approx \frac{1}{2M} \sum_{i=1}^{M} G_{ai} G_{\pi i} \\ G_{ai} &= \nabla_A Q(S_i, A; \phi) \quad \text{where} \quad A = \pi(S_i; \theta) \\ G_{\pi i} &= \nabla_\theta \pi(S_i; \theta) \end{aligned}$$

8. Here, G_{ai} is the gradient of the critic output with respect to the action computed by the actor network, and $G_{\pi i}$ is the gradient of the actor output with respect to the actor parameters. Both gradients are evaluated for observation, S_i.

9. Update the target actor and critic parameters depending on the target update method.

The above DDPG algorithm as presented by Lillicrap et al. (2016) is summarised in pseudo-code in Appendix 9.1. Moreover, the hyperparameters used to tune the DDPG agent are also presented in 9.1.

4 Methodological Framework

This section delves into various methodologies employed for solving the trading model proposed by Almgren and Chriss (2000). By addressing the challenge of deriving optimal trading strategies and associated performance metrics, this research contributes to the field of financial modelling and optimisation. The theoretical framework outlined in Section 3 serves as the foundation for the analysis and exploration of these methodologies. Firstly, drawing upon the theoretical framework outlined in Section 3.2, this work describes how to reproduce the findings documented by Almgren and Chriss (2000) analytically. Secondly, this work explores the utilisation of numerical optimisation, particularly sequential quadratic programming for solving the trading model based on the theoretical foundation from Section 3.3. Thirdly, this work proposes a dynamic programming methodology to solve the trading model based on the theoretical foundation outlined in Section 3.4. Lastly, this work describes the application of reinforcement learning as an alternative approach to solve the optimal execution problem based on the theoretical foundation presented in Section 3.5.

4.1 Solving the Trading Model Analytically

Drawing upon the theoretical framework outlined in Section 3.2 for the derivation of explicit optimal strategies, this work reproduces the findings documented by Almgren and Chriss (2000) in their work on the explicit construction of optimal strategies[21]. Specifically, this work focuses on replicating the optimal trading trajectory as shown in their research. The methodological results that is described in the forthcoming section has been implemented in Python in order to compute the optimal trajectories, the efficient frontier and the expected implementation shortfall.

In order to obtain an analytical solution for the trading model, this work employs equations (3.6) and (3.5), while assuming that the sign of n_j remains constant. This assumption leads us to infer that $U(x) = E(x) + \lambda V(x)$ takes the form of a quadratic function with respect to the control parameters, as elaborated upon in Section 3.2. It is crucial to note that the validity of the analytical solution is contingent upon the strict convexity of λ[22], that is, $\lambda \geq 0$. Guided by this

[21] It is important to note that Almgren and Chriss (2000) do not provide a definitive resolution to the problem of determining optimal strategies. Instead, they contribute by offering a framework or methodology for approaching the analytical solution of such strategies.

[22] The strict convexity condition is relaxed in the subsequent sections.

understanding, we aim to solve equation (3.15), which is expressed as follows

$$n_j = \frac{2\sinh\left(\frac{1}{2}\kappa\tau\right)}{\sinh\left(\kappa T\right)}\cosh\left(\kappa(T - t_{j-\frac{1}{2}})\right)X, \quad j = 1, \ldots, N$$

From 3.2, it is known that $\tilde{\kappa}^2$ can be expressed as

$$\tilde{\kappa}^2 = \frac{\lambda\sigma^2}{\tilde{\eta}} = \frac{\lambda\sigma^2}{\eta(1 - \frac{\gamma\tau}{2\eta})}$$

Therefore,

$$\tilde{\kappa} = \sqrt{\frac{\lambda\sigma^2}{\eta(1 - \frac{\gamma\tau}{2\eta})}} \tag{4.1}$$

The statement $n_j > 0$ holds true for every j as long as $X > 0$, according to the derivation. Consequently, in the case of a sell programme, the optimal strategy solution exhibits a monotonically decreasing behaviour from its initial value, with the rate of decrease determined by the parameter κ. Thus, if the trading intervals are short, κ, is the ratio of the product of volatility and the sellers, or buyers, risk tolerance to the temporary cost parameter, η. The expectation and variance of the optimal strategy are then computed using equations (3.18) and (3.19).

An important observation arising from the analytical solution is that when $\lambda = 0$, the optimal strategy n_j is not well-defined. This is due to the fact that the hyperbolic sine function, sinh, evaluates to zero, i.e., $\sinh(0) = 0$. Consequently, the optimal strategy would involve an undefined value, specifically $\frac{0}{0}$. To resolve this indeterminate form, we must apply L'Hôpital's rule[23] to the following

$$n_j = \frac{2\sinh\left(\frac{1}{2}\kappa\tau\right)}{\sinh\left(\kappa T\right)}\cosh\left(\kappa(T - t_{j-\frac{1}{2}})\right)X$$
$$n_j = \frac{2\sinh\left(\frac{1}{2}\kappa\tau\right)\cosh\left(\kappa(T - t_{j-\frac{1}{2}})\right)X}{\sinh\left(\kappa T\right)}$$

Applying L'Hôpital's rule we get the following

$$\lim_{\kappa\to 0}\frac{f(\kappa)}{g(\kappa)} = \lim_{\kappa\to 0}\frac{f'(\kappa)}{g'(\kappa)} \leftrightarrow$$
$$= \frac{2X\cdot(T - t_{j-\frac{1}{2}})\sinh((T - t_{j-\frac{1}{2}})\kappa)\sinh\left(\frac{\tau\kappa}{2}\right) + X\tau\cosh((T - t_{j-\frac{1}{2}})\kappa)\cosh\left(\frac{\tau\kappa}{2}\right)}{T\cosh(T\kappa)}$$

[23] L'Hôpital's rule is a mathematical theorem that enables the evaluation of limits involving indeterminate forms using derivatives.

where the functions $f(\kappa) = 2\sinh\left(\frac{1}{2}\kappa\tau\right)\cosh\left(\kappa(T - t_{j-\frac{1}{2}})\right)X$, which represents the numerator of n_j, and $g(\kappa) = \sinh(\kappa T)$, which represents the denominator of n_j. By applying L'Hôpital's rule, the expression can be evaluated as κ approaches 0 through the substitution of $\kappa = 0$ into the expression. Denoting the entire expression as $h(\kappa)$, its behaviour near $\kappa = 0$ can be examined.

$$h(\kappa)\Big|_{\kappa=0} = \frac{2X \cdot (T - t_{j-\frac{1}{2}})\sinh((T - t_{j-\frac{1}{2}}) \cdot 0)\sinh\left(\frac{\tau \cdot 0}{2}\right) + X\tau\cosh((T - t_{j-\frac{1}{2}}) \cdot 0)\cosh\left(\frac{\tau \cdot 0}{2}\right)}{T\cosh(T \cdot 0)}$$

Note that since $\cosh(0) = 1$ and $\sinh(0) = 0$, the expression simplifies further

$$h(\kappa)\Big|_{\kappa=0} = \frac{2X \cdot (T - t_{j-\frac{1}{2}}) \cdot 0 \cdot 0 + X\tau \cdot 1 \cdot 1}{T \cdot 1}$$

Simplifying the expression converges, we find that

$$h(\kappa)\Big|_{\kappa=0} = \frac{X\tau}{T} \tag{4.2}$$

This methodological result, i.e., equation (4.2), is in line with Almgren and Chriss (2000), as they state for $\lambda = 0$, which they call the *naïve* strategy, as this represents the optimal strategy corresponding to simply minimising expected transactions costs, i.e., $U(x) = E(x) + \lambda V(x)$, without regard to variance, and corresponds to a simple linear reduction of holdings over the trading period, which is exactly what the methodological result shows by evaluating the optimal strategy as κ approaches zero.

4.2 Solving the Trading Model using Numerical Optimisation

this work utilises numerical optimisation, specifically SQP, to determine the optimal liquidation strategy, efficient frontier and the expected implementation shortfall. The optimisation problem is formulated as follows, as described in Section 3.2 equation (3.13)

$$\min_{x} \quad E[x] + \lambda V[x]$$

In this formulation, x represents the difference between the initial book value and the capture, which is commonly referred to as the implementation shortfall

$$x = XS_0 - \sum n_k \tilde{S}_k$$

Here, XS_0 denotes the initial book value, and $\sum n_k \tilde{S}_k$ represents the sum of the estimated shares executed at different prices $\tilde{S}_k$.

To solve the optimisation problem, the thesis leverages the linear optimisation method available in the optimisation library of the Python package `'SciPy'`. This library provides efficient tools for solving optimisation problems, including linear programming. By employing the SQP algorithm and formulating the problem with the appropriate objective function and constraints, the thesis aims to find the optimal execution strategy that minimises both the expected implementation shortfall and the risk, as measured by the variance $V[x]$. The parameter λ serves as a weighting factor to balance the trade-off between these two objectives. The utilisation of numerical optimisation techniques, such as SQP, allows for a systematic approach to determining the optimal liquidation strategy. By incorporating the implementation shortfall and considering the trade-off between expected implementation shortfall and risk, this methodology provides a comprehensive framework for achieving optimal execution of portfolio transactions.

4.3 Solving the Trading Model using Dynamic Programming

In this work, the application of dynamic programming is employed to address the trading model proposed by Almgren and Chriss (2000) in their work on optimal transactions. This approach distinguishes itself from the two aforementioned methodologies, as Almgren and Chriss (2000) do not explicitly recommend this particular approach for deriving optimal trading strategies. The rationale behind adopting dynamic programming to solve the trading model stems from its capability to tackle problems that exhibit two essential characteristics: (i) problems that can be decomposed into subproblems, and (ii) subproblems that can recur multiple times with their solutions being stored and reused.

In Section 3.4, a comprehensive overview of the theory concerning MDPs in dynamic programming was presented. The framework of MDPs in dynamic programming is employed to address the optimal execution framework proposed by Almgren and Chriss (2000). This framework can be effectively modelled as a Markov decision process, as the problem has the characteristics of both in the subproblem structure and overlapping subproblems. Specifically, the objective of achieving optimal liquidation while minimising transaction costs can be decomposed into multiple trading periods, each constituting a distinct subproblem. Moreover, the presence of overlapping subproblems emerges from the identical nature of the problem across trading periods, enabling the value function to assess the progression of liquidation effectively. Furthermore, it is established that the Almgren and Chriss (2000) optimal transactions framework inherently exhibits termination, meaning that there is no concern regarding an infinite horizon. At the terminal period, the final reward, namely the liquidation cost, is unequivocally known. This characteristic renders dynamic programming an appropriate and well-suited technique for resolving the Almgren and Chriss (2000) framework, facilitating the determination of optimal trading strategies within the given framework.

To incorporate dynamic programming into the Almgren and Chriss (2000) framework, it becomes imperative to transform the problem from a mean-variance perspective, as encountered in numerical optimisation approaches, into the Bellman equation as outlined in Section 3.4. Within the context of implementation shortfall, specifically denoted as $x = XS_0 - \sum n_k \tilde{S}_k$, this work employs the constant absolute risk-aversion (CARA) utility function, which enables the quantification of risk-aversion. The CARA utility function is characterised by a constant absolute risk-aversion coefficient that remains invariant through-

out the trading period (Pratt, 1964). Commonly referred to as the Arrow-Pratt coefficient, it is defined as

$$\rho = -\frac{v''(x)}{v'(x)}$$

where $v(x)$ is the utility function. Consequently, setting the utility of zero equal to zero yields $v(x) = \frac{1}{\rho}(1 - e^{-\rho x})$[24]. It is pertinent to assume that the random variable x follows a normal distribution with mean μ and variance σ^2. Consequently, we obtain the following result:

$$E[v(x)] = \frac{1}{p}\left(1 - \exp\left(-\rho\left(\mu - \frac{\rho}{2}\sigma^2\right)\right)\right) \tag{4.3}$$

Notably, CARA with normally distributed risks exhibits linear preferences concerning the mean and variance of x. Consequently, the trader selects the trade that maximises (or equivalently minimises) the value of $\mu - \frac{p}{2}\sigma^2$. These preferences align precisely with mean-variance preferences (Pratt, 1964).

With the aforementioned considerations in place and recalling the assertion made in Section 3.1, i.e., the implementation shortfall follows a normal distribution, the analysis proceeds with the following CARA utility function for implementation shortfall

$$\min_n \exp\left(-\rho E\left(XS_0 - \sum_{k=1}^{n} n_k \tilde{S}_k\right) + \lambda V\left(XS_0 - \sum_{k=1}^{n} n_k \tilde{S}_k\right)\right) \tag{4.4}$$

Since the minimisation of equation (4.4) is equivalent to the minimisation of an expectation as shown in equation (4.3), this work reformulates the Almgren and Chriss (2000) optimisation problem to (4.4).

Considering the stock price, s, the inventory, x, the revenue of the liquidation, r, and the time t, the Almgren and Chriss (2000) leads to the following terminal condition

$$v(r,x,s,T-1) = \exp\left(-\rho\left[r + x\left(s - g\left(\frac{x}{\tau}\right)\right)\right]\right) \tag{4.5}$$

[24]This formulation is derived by setting $v(0) = 0$ and evaluating $p = 0$.

and the following Bellman equation

$$
\begin{aligned}
v(r,x,s,t) &= \min_{n} E(v^*, x^*, s^*, t+1) \\
r^* &= r + x\left[s - g\left(\frac{n}{\tau}\right)\right] \\
s^* &= s + \sigma\xi\sqrt{\tau} - \tau h(\frac{n}{\tau}) \\
x^* &= x - n
\end{aligned}
\tag{4.6}
$$

where ξ is a normally distributed innovation with mean zero and unit variance.

The implementation of the aforementioned Bellman equation presents computational challenges. This is primarily attributed to the multitude of potential values that the stock price, denoted as s, can assume, subsequently impacting the revenue, denoted as r, and inventory denoted as x. Specifically, considering any combination (x,t), which can be represented as a 2×2 matrix, there exists a vast range of feasible stock price values. Moreover, the tensor (x,t,s) encompasses all possible values for the revenue, leading to the creation of a higher-dimensional tensor, namely (x,t,s,r). Consequently, the implementation of this specific approach for the Almgren and Chriss (2000) trading model utilising dynamic programming encounters the *curse of dimensionality*, rendering it computationally infeasible[25]. In Figure 2 an illustration of the curse of dimensionality is presented.

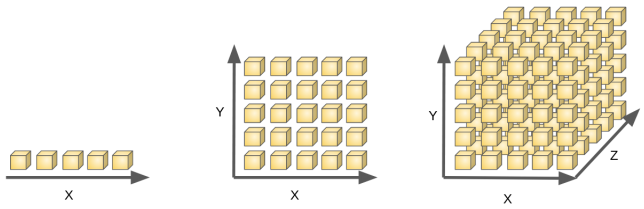

Figure 2: An illustration of the curse of dimensionality (Sutton and Barto, 2018).

There are many ways to solve the curse of dimensionality, however, in this work, the approach is to simplify the state space, v, to

$$
v(r,x,s,t) = u(x,t)\exp\left(-\rho(r+xs)\right) \tag{4.7}
$$

Here $u(x,t)$ is a simplified value function that only depends on x and t. Thus, the Bellman equation has now only two variables and the value function has been

[25] Additionally, the author finds this computational issue falls outside the scope of this work.

reduced from a 4-dimensional tensor to a 2-dimensional matrix[26]. From this reduction of the state space, this work implements the dynamic programming principles mentioned in Section 3.4 to the following terminal condition

$$u(x,T-1) = \exp\left(-\rho x g\left(\frac{x}{\tau}\right)\right) \tag{4.8}$$

and the following Bellman equation

$$u(x,t) = \min_{0<n<x} u(x-n,t+1)\exp\left(-\rho n g\left(\frac{n}{\tau}\right) + \rho(x-n)\tau h\left(\frac{n}{\tau}\right) + \frac{1}{2}\rho^2(x-n)^2\sigma^2\tau\right) \tag{4.9}$$

Defining the temporary market impact function, the permanent market impact function and the Hamiltonian equation, the Bellman equation is iteratively solved using Python. Thereby, enabling this work to solve the trading model using dynamic programming.

[26]This change of variables relies on the assumption that $r + xs$ is equal to the *fair value* valuation of the trader's portfolio at any time t during the entire trading process. This is specifically true for shorter time-horizons.

4.4 Solving the Trading Model using Reinforcement Learning

In this work, the application of reinforcement learning is employed to solve the trading model proposed by Almgren and Chriss (2000). As with dynamic programming, this approach also distinguishes itself from the solutions methodologies proposed by Almgren and Chriss (2000). The rationale behind adopting a reinforcement learning approach to solving the trading model is that reinforcement learning can combat some of the drawbacks that exist in dynamic programming, namely, completeness and the curse of dimensionality.

Reinforcement learning offers two advantages over classic dynamic programming. Firstly, agents learn online and continually adapt while performing a given task. Secondly, the methods of RL can employ function approximation algorithms to represent the knowledge, which allows the agent to generalise across the state space so the learning time scales a lot faster (Hendricks and Wilcox, 2014).

As in the case of dynamic programming, we need to restate the optimal execution, i.e., the optimal liquidation problem, in terms of (i) states, (ii) actions and (iii) rewards. For (i) the optimal liquidation problem entails that the trader sells the entire portfolio within a given time frame, and thus, the vector of states must contain information about the time remaining, or equivalently, the number of trades remaining. In this work, the vector at time t_k is defined as follows

$$[r_k, m_k, i_k] \tag{4.10}$$

where

$$r_k = \log\left(\frac{\tilde{S_k}}{\tilde{S_{k-1}}}\right) \tag{4.11}$$

$$m_k = \frac{N_k}{N} \tag{4.12}$$

$$i_k = \frac{x_k}{X} \tag{4.13}$$

Here equation (4.11) is the log-return at time t_k, equation (4.12) is the number of trades remaining at time t_k normalised by the total number of trades and equation (4.13) is the remaining number of shares at time t_k normalised by the total number of shares. The log-returns capture information about the stock prices before time t_k, which then can be used to detect possible price trends. The number of trades and shares remaining allows the agent to learn to sell all

the shares within a given time frame[27].

For (ii), namely, the action space, the optimal liquidation problem requires the trader only to sell stocks, and thus, it is reasonable to define the action, a_k, to be the number of shares to sell at time, t_k. Even though DDPG has good convergence properties, interpreting the action directly as the number of shares to sell at each time step can lead to some problems if x_k is large, as the agent will need to produce actions with very high values. Therefore, this work interprets a_k as a percentage. In this case, the actions produced by the agent will only need to be between zero and one. Using this specific interpretation, the trader can determine the number of shares to sell at each time step using the following

$$n_k = a_k \times x_k \tag{4.14}$$

For (iii), namely, the rewards, this work utilises that the original problem is minimisation problem, i.e., $\min_x(U(x) = E(x) + \lambda V(x)$. Therefore, at each time step, this work computes the utility using equations (3.4) and (3.5) for the remaining time and inventory while the risk parameter, λ, is held constant. Thus, the optimal trading trajectory is computed at time t as follows

$$R_t = \frac{U_t(X_t^*) - U_{t+1}(X_{t+1}^*)}{U_t(x_t^*)} \tag{4.15}$$

From equation (4.15), the reward is defined, and normalised in terms of the overall utility in order to train the DDPG agent.

The DDPG algorithm is trained in a simple simulated trading environment. The environment simulates stock prices that follow a discrete arithmetic random walk and the permanent and temporary market impact functions are linear functions of the rate of trading following Almgren and Chriss (2000). The DDPG algorithm is trained and the simulation environment is created using Python. From this, this work is able to solve the trading model using reinforcement learning.

[27]One may note, as in the case with dynamic programming, this state vector can hold many more state variables.

5 Results

This section of the thesis aims to present its findings using the methodological framework delineated in Section 4, incorporating the fundamental concepts and principles discussed in Section 3. Firstly, the parameters for the baseline trading models are introduced, providing a comprehensive understanding of the foundational elements upon which subsequent analyses are built. This initial step serves as a crucial benchmark for evaluating the effectiveness and reliability of the proposed methodologies. Subsequently, the results obtained through the application of the proposed methodologies are presented in chronological order, offering a clear depiction of their temporal evolution and progression. Lastly, this section of the thesis incorporates empirical examples and distributional cases to bolster the presentation of the findings. The inclusion of empirical examples allows for a practical illustration of the theoretical concepts discussed earlier, thereby grounding the research in real-world scenarios and enhancing its applicability. Furthermore, the utilisation of distributional examples enables the exploration of various statistical distributions, shedding light on the potential implications and limitations of the proposed methodologies in different contexts.

5.1 Parameters for the Baseline Trading Model

In order to solve the trading model whether it is analytically, numerically, dynamically or through reinforcement learning, a set of parameters must be chosen. As a point of departure, and for the purpose of replicating the results of Almgren and Chriss (2000), this work initially makes use of the following parameters. First, it assumed that we have a single stock with a current market price of $S_0 = 50$ and that the seller initially holds 1 million shares, i.e., an initial portfolio size of \$50 million. This stock has annual volatility of 30% and a 10% expected annual return, a bid-ask spread of $\frac{1}{8}$ and a daily trading volume of 5 million shares.

With a trading year of 250 days, this is equivalent to daily volatility of $0.3/\sqrt{250} = 0.019$ and an expected return of $0.1/250 = 4 \times 10^{-4}$. Thus, in order to obtain the absolute parameters, namely, σ and α, following Almgren and Chriss (2000), these results must be scaled by the stock price such that $\sigma = 0.019 \times 50 = 0.95$ and $\alpha = (4 \times 10^{-4}) \times 50 = 0.02$.

Over the period, if the seller held the initial position with no trading, the fluctuations in the stock price would be Gaussian with a standard deviation of $\sigma\sqrt{T} = 2.12\$/\text{share}$. The parameters for the temporary cost function is $\epsilon = \frac{1}{16}$,

i.e., the fixed part of the temporary costs will be one-half of the bid-ask spread. For η, it is assumed that for each 1% of the daily volume that the seller trades, they incur a price impact equal to the bid-ask spread. Thus, trading at a rate of 5% of the daily volume incurs a one-time cost on each trade of $\frac{5}{8}$. Under this assumption, $\eta = \frac{1}{8}/(0.01 \times 5 \times 10^6) = 2.5 \times 10^{-6}$. For the permanent costs, the price effects become significant when the trader sells 10% of the daily volume[28]. Therefore, supposing that significant means that the price depression is equal to one bid-ask spread, and the effect is linear for smaller and larger trading rates, then $\gamma = \frac{1}{8}/(0.1 \times 5 \times 10^6) = 2.5 \times 10^{-7}$. This parameter is a fixed cost independent of the optimal trajectory. The parameters are summarised in Table 2.

Table 2: Parameter Values for Baseline Case

Initial stock price:	$S_0 = 50\$/\text{share}$
Initial holdings:	$X = 10^6$ share
Liquidation time:	$T = 5$ days
Number of time periods:	$N = 5$
30% annual volatility:	$\sigma = 0.95(\$/\text{share})/\text{day}^{1/2}$
10% annual growth:	$\alpha = 0.02(\$/\text{share})/\text{day}$
Bid-ask spread $= \frac{1}{8}$:	$\epsilon = 0.00625\$/\text{share}$
Daily volume 5 million shares:	$\gamma = 2.5 \times 10^{-7}$ $\$/\text{share}^2$
Impact at 1% of market:	$\eta = 2.5 \times 10^{-6}(\$/\text{share})/(\text{share/day})$

Note: This table presents the parameter values for the baseline case following Almgren and Chriss (2000).

[28]This is a common rule of thumb, which is stated by Almgren and Chriss (2000).

5.2 Analytical Results

The efficient frontier for the analytical solution to the trading model is computed using an evenly spaced vector of risk tolerance levels, i.e., λ, from 2×10^{-4} to 0 with 200 intermediate values as well as the baseline parameters mentioned in Section 5.1. The resulting findings are graphically depicted in the subsequent plot

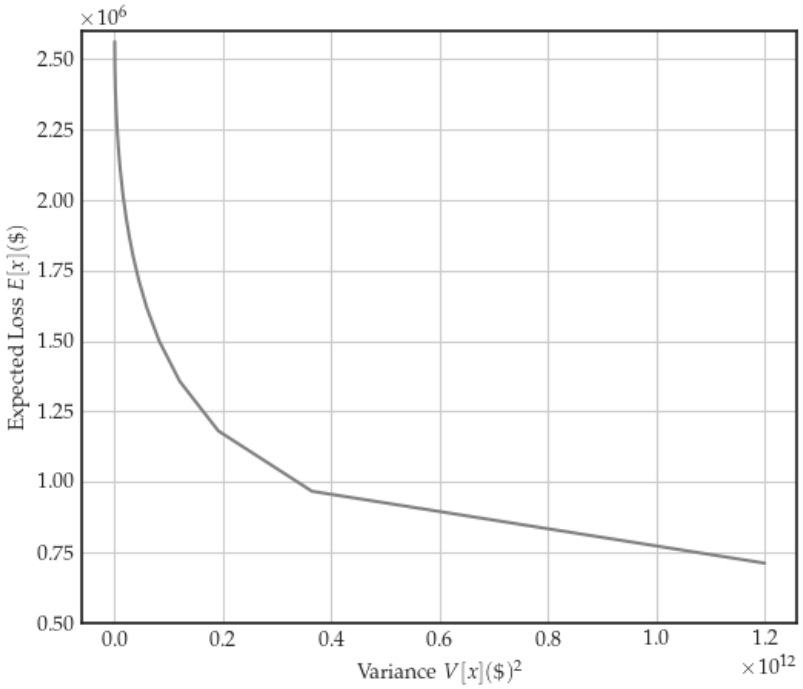

Figure 3: The efficient frontier (analytical). The parameters are as in Table 2. The area above the curve is a set of variances and expectations attainable by some time-dependent strategy. The curve is the efficient frontier.

From Figure 3, it is noted that the efficient frontier has accurate mean-variance properties, i.e., as the trader becomes more risk-neutral, the expected loss decrease, but at a cost of variance.

The optimal strategies for the analytical solution are computed for a vector of risk parameters, $r = [2\times10^{-4}, 2\times10^{-5}, 2\times10^{-6}, 2\times10^{-7}, 2\times10^{-8}, 2\times10^{-10}, 0]$[29]. Using the vector r results in the following optimal trajectories.

[29]Notably, the vector r has starting values 2×10^{-4}. This is due to the behaviour of the hyperbolic sine function, which for increasing κ explodes. See Appendix 9.2 for a graphical inspection.

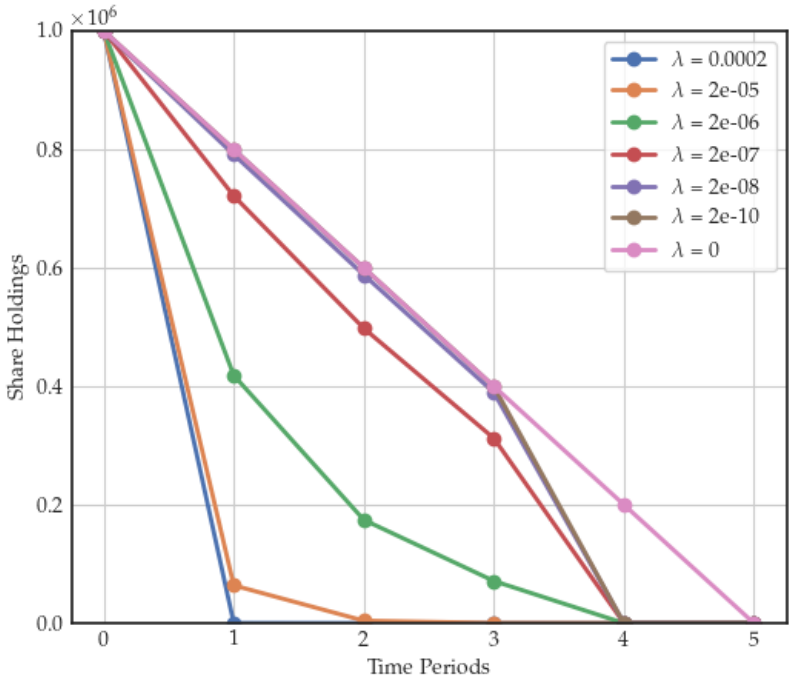

Figure 4: The optimal trajectories (analytical). The parameters are as in Table 2.

In Figure 4, seven distinct trading trajectories are shown, following the financial interpretation of λ, i.e., as a risk coefficient, it is clear that as the trader becomes more risk-averse as the trader wishes to liquidate the initial portfolio quicker in order to reduce exposure to volatility risk. For $\lambda = 0$, i.e., the naïve strategy, the analytical solution yields results in accordance with the interpretation of a risk-neutral trader, as the risk-neutral trader only focuses on minimising expected transaction costs with regard to variance. For the baseline case, this corresponds to a simple linear reduction of holdings over the trading period. These results are in exact accordance with Almgren and Chriss (2000), who states that for larger values of κ[30] while holding the trading time constant. Additionally, as a consequence of the analysis, different-sized portfolios of the same securities will be liquidated in exactly the same fashion, provided that it has the same trading horizon and the risk-aversion parameter is held constant. For example, a portfolio of size \$50 million will have exactly the same optimal liquidation path as a portfolio of size \$5 million.

[30]Increasing values of κ is a direct result of increasing values of λ following equation 4.1.

Following the analysis of the efficient frontier and the optimal strategies, it is convenient to look specifically at the expectation and variance of the implementation shortfall for different λ. Using the vector r, the following table shows the resulting expectation and variance

Table 3: Implementation Shortfall (analytical)

λ	$E[x]$(\$)	$V[x]$(\$)
2×10^{-4}	2.56194×10^{6}	0.00000×10^{12}
2×10^{-5}	2.29622×10^{6}	0.00318×10^{12}
2×10^{-6}	1.17952×10^{6}	0.19231×10^{12}
2×10^{-7}	0.86947×10^{6}	0.84506×10^{12}
2×10^{-8}	0.93834×10^{6}	1.12014×10^{12}
2×10^{-10}	0.95234×10^{6}	1.15958×10^{12}
0	0.71250×10^{6}	1.20000×10^{12}

In Table 3, a pattern is observed, which aligns with the findings presented in Figure 3. The results demonstrate notable differences between the highly risk-averse trader with $\lambda = 2 \times 10^{-4}$ and the risk-neutral trader with $\lambda = 0$. Specifically, the highly risk-averse trader exhibits an expected implementation shortfall of approximately \$2.6 million, whereas the risk-neutral trader's expected implementation shortfall amounts to approximately \$0.7 million. However, it is noteworthy that the variance for the risk-neutral trader surpasses that of the highly risk-averse trader by more than 95 million times in magnitude. Furthermore, an interesting finding is that the mean-variance structure seems to break down at $\lambda = 2 \times 10^{-7}$, as it has a lower expected shortfall and variance compared to $\lambda = 2 \times 10^{-8}$. This discrepancy can be attributed to computational considerations necessitated by ensuring the trader's complete liquidation of the entire position in the final period. This necessity leads to an increase in expected shortfall as the trader has to liquidate his entire position in the last period also resulting in an extreme increase in variance between the two risk parameters. This can also be seen in Figure 4. However, subsequent sections reveal that this phenomenon is not observed in other methodologies and can be considered an outlier.

Conclusively, this work has been able to replicate Almgren and Chriss (2000)'s trading model analytically and find that the efficient frontier, optimal trajectories and the ex-post portfolio evaluation measurement, implementation shortfall, coincide with suggested results in Section 4.1.

5.3 Numerical Optimisation Results

The numerical optimisation results differ somewhat from the analytical results, dynamic programming results and reinforcement learning results in terms of the reliability of the implementation of the trading model. As mentioned before, in the seminal paper by Almgren and Chriss (2000) the trading model is solved using numerical optimisation. Consequently, the accuracy of the implementation of the trading model in this work can be evaluated solely on the ability to reproduce their results using the optimisation technique. More specifically, the numerical optimisation can be seen as the benchmark of the methodologies if this work is able to replicate the findings of Almgren and Chriss (2000).

The efficient frontier for the numerical optimisation approach is computed using the same risk parameters as in Almgren and Chriss (2000), namely, (i) Trajectory A $= 2 \times 10^{-6}$, (ii) Trajectory B $= 0$ and (iii) Trajectory C $= -2 \times 10^{-7}$ and the parameters in Table 2. This produces the following plot

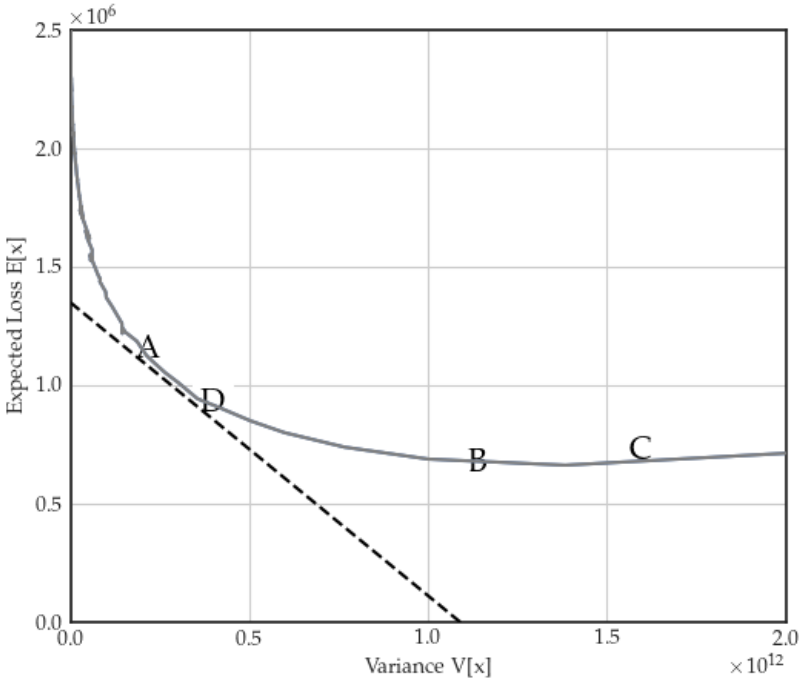

Figure 5: The efficient frontier (numerical optimisation). The parameters are as in Table 2. The area above the curve is a set of variances and expectations attainable by some time-dependent strategy. The curve is the efficient frontier. The straight dashed line illustrates a selection of a specific optimal strategy for $\lambda = 10^{-6}$, i.e., point D.

As in the case with the efficient frontier from the analytical solution, Figure 5 has accurate mean-variance properties until somewhere between the illustrated points B and C, as liquidation strategies here incur higher variance for the same, or higher, expected costs. This is in exact accordance with the findings of Almgren and Chriss (2000).

The optimal strategies for the numerical optimisation approach are computed using the aforementioned risk parameters. This yields the following plot

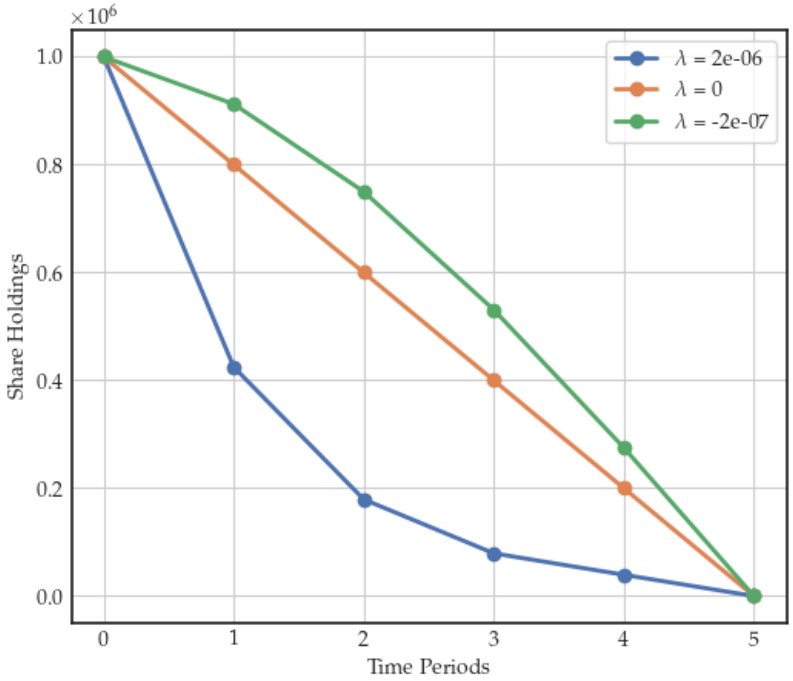

Figure 6: Optimal trajectories (numerical optimisation). The trajectories corresponding to the points are shown in Figure 5. A: $= 2 \times 10^{-6}$, B: $= 0$ and C: $= -2 \times 10^{-7}$

From Figure 6, three distinct trading trajectories are shown[31]. Trajectories A and B are identical to the optimal trajectories in the analytical solution for the same risk parameter, and additionally, it is equivalent to the findings of Almgren and Chriss (2000). However, as the analytical solution only holds for

[31]The rationale behind the selection of these particular three trajectories, as opposed to displaying the trajectories corresponding to equivalent risk parameters as depicted in Figure 4, can be attributed to their alignment with the risk parameters specified by Almgren and Chriss (2000) in their work on optimal transactions. By exclusively showcasing these trajectories, the emphasis is placed on reproducibility and maintaining consistency with Almgren and Chriss (2000) chosen risk parameters. Supplementary trajectories can be found in Appendix 9.3.

$\lambda \geq 0$, it is quite interesting to examine Trajectory C. This trajectory would only be chosen by a trader who is a *risk-lover*, and thereby, postpones selling and incurring both higher expected trading costs due to rapid sales towards the end of the trading period and higher variance during the extended period of holding the security.

Furthermore, following the analysis of the efficient frontier and optimal strategies for the numerical optimisation approach, this work turns to the expectation and variance of the implementation shortfall for the specific λ's. This results in the following table In Table 4, the findings in Table 3 are corroborated, as

Table 4: Implementation Shortfall (numerical optimisation)

λ	$E[x]$(\$)	$V[x]$(\$)
2×10^{-6}	1.14792×10^{6}	0.21882×10^{12}
0	0.66250×10^{6}	1.13906×10^{12}
-2×10^{-7}	0.71754×10^{6}	1.59562×10^{12}

the expected shortfall and variance of the expected shortfall are the same[32] as the analytical solution for the same risk parameter. As a consequence of the numerical optimisation approach, this work can now evaluate the implementation shortfall for the risk-loving trader. The numerical results in Table 4 show exactly what was shown in Figure 5, i.e., that the risk-loving trader will incur the higher expected cost for a higher level of variance compared to the risk-neutral trader. Thus, it is evident that being a risk-loving trader can *never* be an optimal strategy in this framework, as one can obtain lower expected costs at lower levels of variance. Hence, this work will only focus on $\lambda \geq 0$ in the following sections.

In conclusion, this work has been able to replicate Almgren and Chriss (2000)'s trading model using SQP and find that the efficient frontier, optimal trajectories and the ex-post portfolio evaluation measurement, implementation shortfall, coincide with Almgren and Chriss (2000)'s findings.

[32] With numerical differences of $\approx 2\%$.

5.4 Dynamic Programming Results

In this section, this work applies the methodology proposed in Section 4.3, in order to compute the efficient frontier, the optimal strategies and implementation shortfall as in the previous two sections.

The efficient frontier for the dynamic programming approach is computed using an evenly spaced vector of risk tolerance levels as in Section 5.2, as it was previously established that the risk-loving trader behaves inefficiently according to mean-variance strategies. This produces the following plot

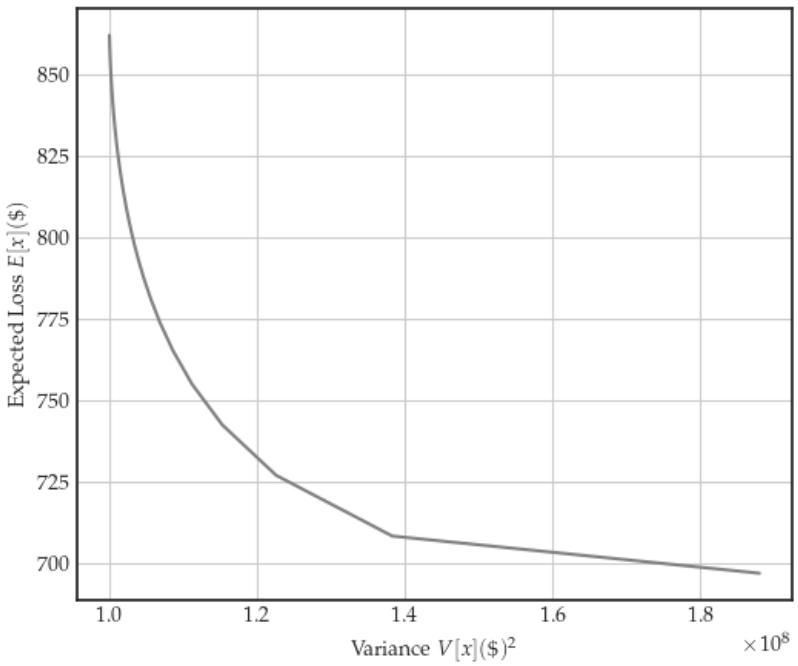

Figure 7: The efficient frontier (dynamic programming). The parameters are as in Table 2. The area above the curve is a set of variances and expectations attainable by some time-dependent strategy. The curve is the efficient frontier.

As with the two previous cases, it is evident that computing the efficient frontier following the methodology presented in Section 4.3 has accurate mean-variance properties. However, it should be noted that the expected loss and variance of the loss are scaled differently in this case. Specifically, this is due to the fact that the initial holding is now set to $X = 10000$, i.e., one-hundredth of the original size following Almgren and Chriss (2000). This is set to combat the curse of dimensionality.

The optimal strategies for the dynamic programming approach are computed using the same vector of risk parameters as in Section 5.1. This yields the following optimal strategies

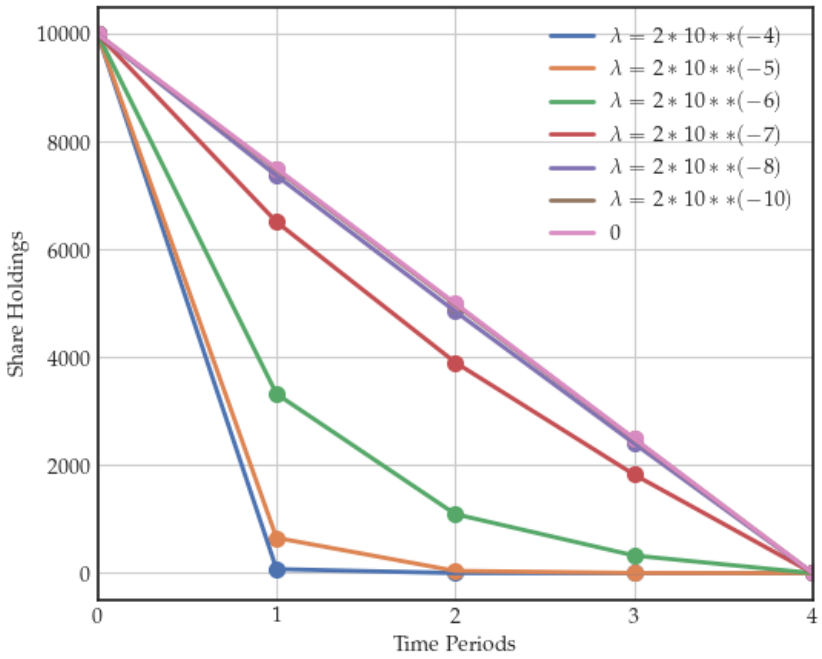

Figure 8: The optimal trajectories (dynamic programming). The parameters are as in Table 2.

From Figure 8, the findings from Figure 2 are corroborated, namely, that as the trader becomes more risk-averse the quicker the liquidation of the initial portfolio becomes. However, it does not seem as if a smaller initial holding, i.e., $X = 10000$, has an effect on the liquidation path, as was also argued in Section 5.2. This may seem contrary to the intuition that a large portfolio is effectively as liquid as a small portfolio. This is a consequence of the linear market impact assumption, i.e., η, which has the mathematical consequence that both variance and market impact scale quadratically with respect to the portfolio size. However, for large portfolios, it may be more reasonable to suppose that the temporary impact cost function has higher-order terms such that costs increase *superlinearly*, i.e., faster growth than linear growth, with trade size. Nevertheless, with non-linear impact functions, the general framework used in this thesis still applies, but it is not possible to obtain explicit exponential solutions as in the linear case. A practical solution to this problem would be to choose different values for the temporary impact parameter, η depending on the overall size of the initial holding.

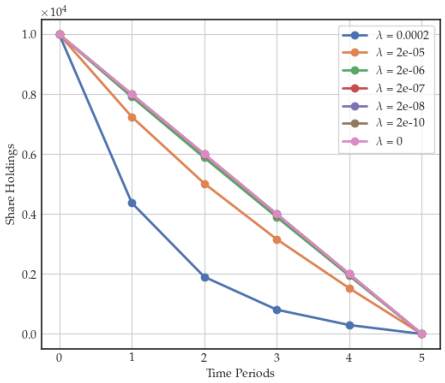

Figure 9: The optimal trajectories (dynamic programming). The parameters are as in Table 2. $X = 10{,}000$ and $\eta = 2 \times 10^{-4}$

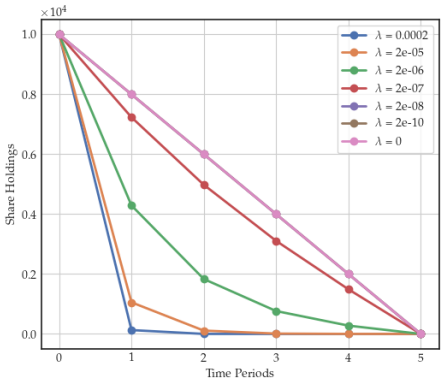

Figure 10: The optimal trajectories (dynamic programming). The parameters are as in Table 2. $X = 10{,}000$ and $\eta = 2 \times 10^{-6}$

In Figures 9 and 10, it is shown that increasing the effect of the temporary impact from selling does have a significant effect on the optimal liquidation path. As the effect increases and while holding the risk parameter constant, the trader will liquidate his position slower to minimise the temporary market impact[33].

Following the previous analysis of the efficient frontier and the optimal liquidation paths, this work turns to the expectation and variance of the shortfall for the dynamic programming approach. This results in the following table

Table 5: Implementation Shortfall (dynamic programming)

λ	$E[x]$(\$)		$V[x]$(\$)	
2×10^{-4}	862.18689	(2.52442×10^6)	1.00069×10^{-8}	(0.00000×10^{12})
2×10^{-5}	797.20766	(2.13713×10^6)	1.03522×10^{-8}	(0.00376×10^{12})
2×10^{-6}	708.17836	(1.10946×10^6)	1.38574×10^{-8}	(0.20137×10^{12})
2×10^{-7}	696.20896	(0.86027×10^6)	1.78564×10^{-8}	(0.85252×10^{12})
2×10^{-8}	696.73510	(0.91838×10^6)	1.86279×10^{-8}	(1.09810×10^{12})
2×10^{-10}	696.84285	(0.92949×10^6)	1.87245×10^{-8}	(1.13178×10^{12})
0	697.81504	(0.72057×10^6)	1.93754×10^{-8}	(1.21359×10^{12})

Note: Values in parentheses are scaled by a factor e, which is the ratio between the SQP expectation and variance and the DP expectation and variance for the implementation shortfall with an initial holding of $X = 10{,}000$.

[33] In Appendix 9.4, supplementary trajectories are provided that show the liquidation path for varying X and η.

In Table 5, the findings from Tables 3 and 4 are supported, as the expected shortfall and variance follow the same structure for the same risk parameters. Comparing the dynamic programming solution to the analytical solution, it is found that the expected shortfall is lower in all but the naïve strategy. However, the variance for the analytical solution is lower in all cases but $\lambda = 2 \times 10^{-8}$ and $\lambda = 2 \times 10^{-10}$. Compared to the numerical optimisation approach, i.e., SQP, dynamic programming has a higher expected loss at each level of risk-aversion. Nevertheless, the dynamic programming approach generates lower variance than the SQP solution.

Conclusively for the dynamic programming approach, this work has been able to solve Almgren and Chriss (2000)'s trading model. It is found that the dynamic programming approach can replicate the properties of the trading, i.e., creating an efficient frontier and offering optimal trajectories aligned with the model's intuition. In terms of ex-post evaluation, the dynamic programming approach differs from both the analytical and the numerical solution, as it yields a lower expected cost from trading in most cases against the analytical solution whilst generating a lower variance when compared to the numerical optimisation approach. Additionally and specifically for the dynamic programming approach, this work offers an interpretation of the temporary market impact coefficient, η, and shows that it has a significant effect on the optimal liquidation path. As a consequence of the significant effect on the optimal liquidation path by the temporary market impact parameter, and the computational difficulties for large X in dynamic programming, this thesis suggests that the dynamic programming approach is infeasible for this trading framework, even if, it produces similar results with mean-variance properties for small X.

5.5 Reinforcement Learning Results

In this section, this work applies the methodology proposed in Section 4.4, to compute the efficient frontier, the optimal strategies and implementation.

The efficient frontier for the reinforcement learning approach, i.e., the DDPG agent, is computed using the same risk parameters as in the analytical case, however, the expectation and variance are simulated 10,000 times to ensure convergence[34]. This yields the following plot

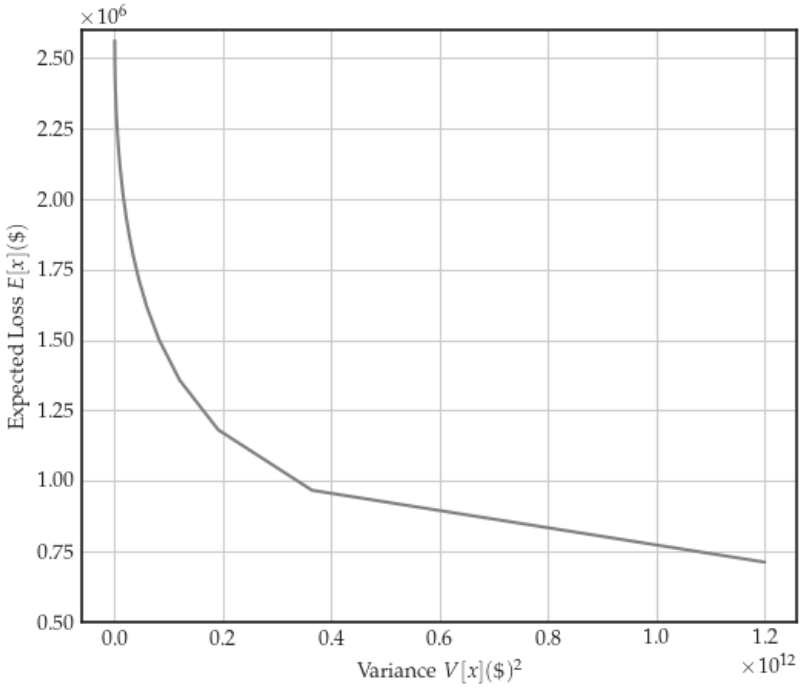

Figure 11: The efficient frontier (reinforcement learning). The parameters are as in Table 2. The area above the curve is a set of variances and expectations attainable by some time-dependent strategy. The curve is the efficient frontier.

Similar to the dynamic programming approach, the DDPG agent exhibits remarkable capabilities in replicating the mean-variance characteristics of the trading model after a sufficient number of iterations, specifically 10,000 iterations in this case. The DDPG agent leverages its learning capacity to accurately capture and emulate the fundamental mean-variance properties that underlie the trading model, enabling it to make informed decisions that balance risk and return effectively. However, a notable advantage of the DDPG approach becomes apparent when comparing it to the dynamic programming

[34]See Appendix 9.5 for visual depiction of the iterations.

method. In the case of DDPG, the learning process can be initialised with a significantly larger state space, specifically with $X = 1{,}000{,}000$, without encountering the curse of dimensionality problem. Unlike dynamic programming, which experiences challenges in scaling up due to the exponential growth of the state space, DDPG circumvents this issue by employing a parameterised policy representation and utilising neural networks for function approximation. This allows the DDPG agent to effectively handle and learn from high-dimensional state spaces, such as the large-scale trading model considered in this study.

The optimal strategies for the reinforcement learning approach are computed using the vector of risk parameters as in Section 5.1. This produces the following plot

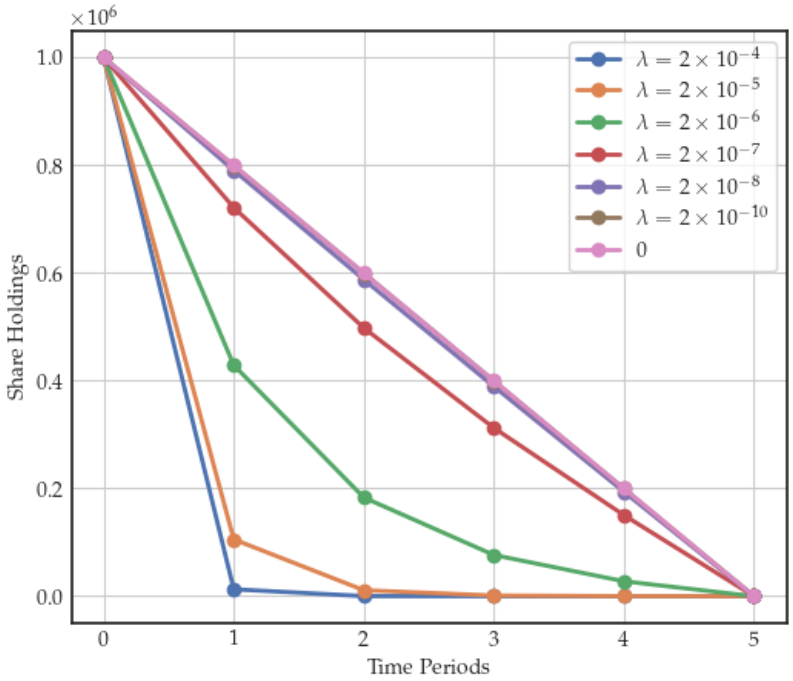

Figure 12: The optimal trajectories (reinforcement learning). The parameters are as in Table 2.

In Figure 12, the results substantiate the findings previously obtained in the three preceding sections, establishing a consistent pattern whereby increased risk-aversion among traders leads to accelerated liquidation processes. This observation decisively confirms that irrespective of the employed methodologies to solve the trading model, they all successfully capture the inherent characteristic of accelerated liquidation for traders with higher risk-aversion.

Turning to the implementation shortfall, this work computes the expectation and variance for the reinforcement learning approach with 10,000 iterations. The results are in the following table

Table 6: Implementation Shortfall (reinforcement learning)

λ	$E[x](\$)$	$V[x](\$)$
2×10^{-4}	2.50245×10^{6}	0.00015×10^{12}
2×10^{-5}	2.11089×10^{6}	0.00995×10^{12}
2×10^{-6}	1.14594×10^{6}	0.19773×10^{12}
2×10^{-7}	0.69526×10^{6}	0.78552×10^{12}
2×10^{-8}	0.66992×10^{6}	1.02543×10^{12}
2×10^{-10}	0.66954×10^{6}	1.05949×10^{12}
0	0.66954×10^{6}	1.05984×10^{12}

In Table 6, the findings obtained earlier are further supported, as both the expected shortfall and variance exhibit similar patterns. A comparison of the results for expectation and variance with the previous methodologies reveals that the DDPG agent yields the lowest expected shortfall for large values of λ. However, this advantage comes at the expense of increased variance in implementation shortfall for higher levels of risk-aversion. Conversely, for small values of λ (specifically $\lambda \leq 2 \times 10^{-7}$ and beyond), the reinforcement learner exhibits the lowest variance in implementation shortfall. Nonetheless, this reduction in variance does not come at the cost of expectation, as the analytical solution yields the highest expected shortfall for these values, except in the case of $\lambda = 0$ where dynamic programming produces the highest expected shortfall. It is worth noting that dynamic programming generating the highest value of expected shortfall, in this case, may be attributed to the fact that the actual value of λ is not precisely zero but an approximation[35].

In conclusion, the reinforcement learning approach has been able to accurately capture the dynamic of the Almgren and Chriss (2000) trading model. It is found that the DDPG agent can reproduce the mean-variance properties of the model, i.e., mimic the efficient frontier. Moreover, the reinforcement learner is also able to create optimal trajectories that are aligned with the theory of the model. In terms of ex-post performance evaluation, the DDPG agent is, in fact, able to generate results aligned with the other methodologies, but with specific advantages for different risk-aversion levels.

[35]Please refer to Appendix 9.5 for a table presenting comprehensive results from all four methodologies.

5.6 Empirical & Distributional Examples

In this section, the thesis introduces an empirical and distributional implementation of the trading model. Building upon the preceding analysis from Sections 5.2 to 5.5, which revealed comparable outcomes across the four distinct methodologies and due to computational and temporal constraints, the subsequent analysis in this work exclusively focuses on utilising the numerical optimisation and reinforcement learning approach.

5.6.1 Empirical Examples

To generate empirical examples, this work leverages Python's Yahoo Finance API to access historical data. The selection the of specific stocks for analysis is predicated on their respective levels of volatility. Therefore, this work employs the services of Alpha Query Inc.[36] to identify a pair of stocks exhibiting high and low volatility, respectively. Tesla, Inc. (TSLA) is chosen as the high volatility stock, while The Procter & Gamble Company (PG) is selected as the low volatility counterpart. Notably, the historical 180-day volatilities (Close-to-Close) for these stocks stand at 0.62 and 0.15, respectively.

In the case of real data, it is important to calibrate the parameter of the trading model based on the historical data of the stock. Specifically, the initial stock price, annual volatility, bid-ask spread, permanent impact parameter, and temporary impact parameter are subject to calibration. The subsequent table provides a concise summary of the modifications made

Table 7: Parameter Values for Empirical Examples

Initial stock price:	S_0 = Close price
Initial holdings:	$X = 10^6$ share
Liquidation time:	$T = 5$ days
Number of time periods:	$N = 5$
Annual volatility:	σ = Standard deviation of daily close price
Annual growth:	α = N/A
Bid-ask spread:	ϵ = Spread / 2
Permanent impact:	γ = Spread / 0.1 × Avg. Daily Volume
Temporary impact:	η = Spread / 0.01 × Avg. Daily Volume

[36] https://www.alphaquery.com/

In the following figures, Tesla's and PG's stock price and daily volume are depicted in the period from 1 January 2022 to 31 December 2022.

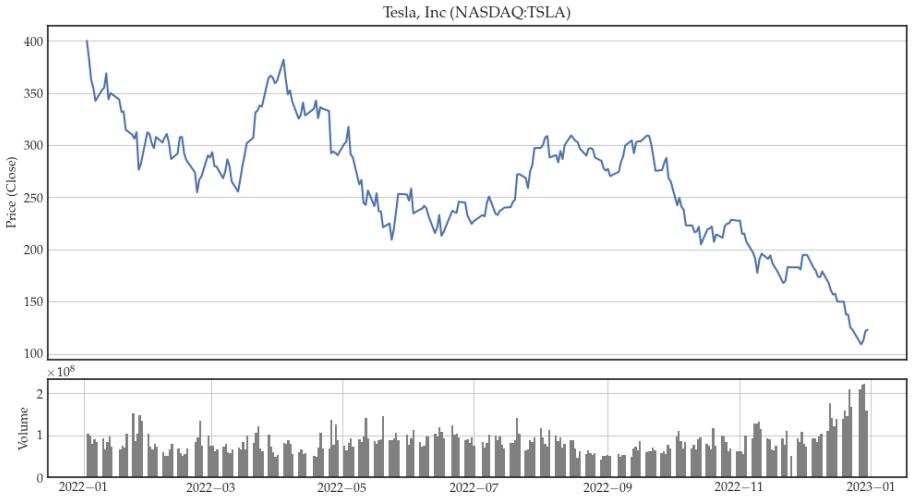

Figure 13: Tesla Inc. - Stock price & Daily volume. The parameters are as in Table 7.

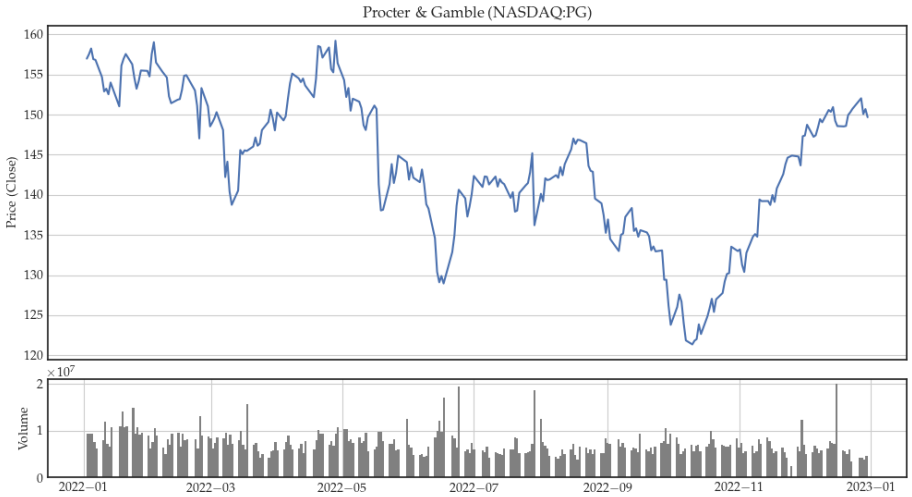

Figure 14: PG - Stock price & Daily volume. The parameters are as in Table 7.

Based on the analysis of Figure 13, it becomes apparent that TSLA exhibited a noticeable downward trajectory in its stock price throughout the entirety of 2022. The observed decline amounted to a significant reduction of over 50 per cent over the course of the year. Conversely, Figure 14 demonstrates a less definitive pattern in the stock price of PG. Instead of a clear trend, the data indicates a decline in the first half of the year, followed by a subsequent increase

that brought the stock price back to nearly the same level as at the beginning of the year by the second half of 2022. These findings underline the stark contrast between the performance of TSLA and PG. TSLA's substantial decrease in stock price suggests a period of notable volatility and downward pressure. It may reflect a range of factors such as market dynamics, company-specific developments, or broader industry trends. In contrast, the oscillating stock price of PG indicates a lack of a decisive trend over the given time frame. The initial decline followed by a recovery suggests a degree of equilibrium, potentially influenced by factors specific to PG's industry or company.

In the subsequent figures, the optimal trajectories for a trader holding Tesla and PG stock with the parameter values as in Table 7 are depicted for the numerical optimisation and the reinforcement learning approach.

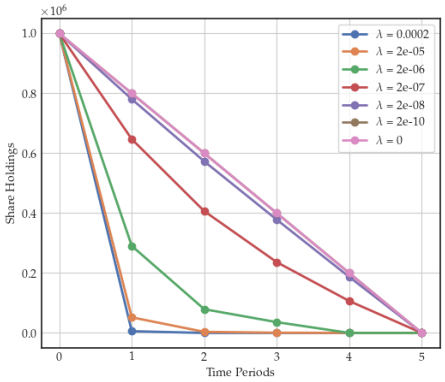

Figure 15: The optimal trajectories for TSLA (numerical optimisation). The parameters are as in Table 7.

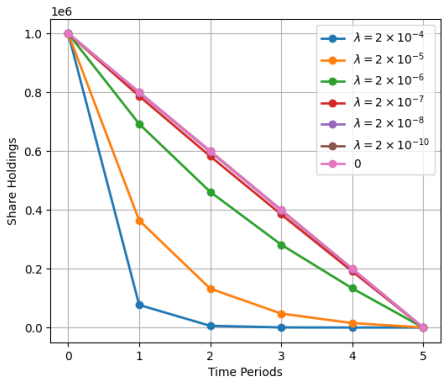

Figure 16: The optimal trajectories for TSLA (reinforcement learning). The parameters are as in Table 7.

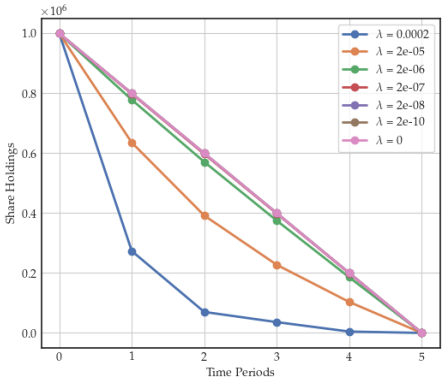

Figure 17: The optimal trajectories for PG (numerical optimisation). The parameters are as in Table 7.

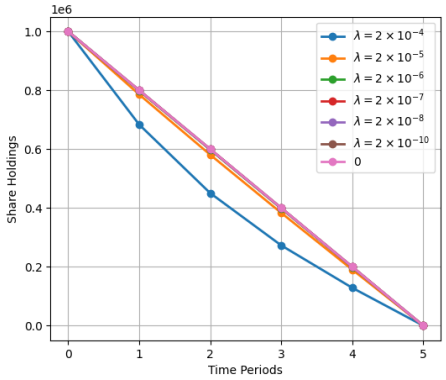

Figure 18: The optimal trajectories for PG (reinforcement learning). The parameters are as in Table 7.

The results obtained from Figures 15 and 18 indicate that, regardless of the stock's volatility level, the DDPG agent consistently exhibits a tendency to smoothen the liquidation of initial holdings across all levels of risk-aversion, i.e., λ. This observation suggests that the DDPG agent inherently places a lower value on risk compared to the numerical optimisation approach, as it converges to the naïve strategy, namely, liquidating in even baskets to minimise expected shortfall. Additionally, it is not surprising to find that the overall volatility of the stock significantly influences the optimal trajectories. Both the numerical optimisation approach and the reinforcement learning approach demonstrate that reduced volatility leads to a more linear trajectory for all risk levels. This outcome aligns with the expectation that as $V(x) \to 0$, the objective function simplifies to solely minimising $E(x)$, which has been shown to be achieved through a linear reduction of the initial holdings.

The tables below display the expected value and variance of the implementation shortfall for TSLA and PG. To achieve consistent outcomes for the reinforcement learning approach, a total of 10,000 iterations, i.e., learning episodes, were executed to attain converging results. See Appendix 9.7 for figures of learning episodes.

Table 8: Implementation Shortfall for TSLA (numerical optimisation & reinforcement learning)

	SQP		DDPG		SQP-DDPG(%)	
λ	$E[x]$(\$)	$V[x]$(\$)	$E[x]$(\$)	$V[x]$(\$)	$E[x]$(\$)	$V[x]$(\$)
2×10^{-4}	30.05824×10^{6}	0.05327×10^{12}	22.59538×10^{6}	0.00146×10^{12}	−24.82	−97.25
2×10^{-5}	28.71141×10^{6}	0.05341×10^{12}	19.16480×10^{6}	0.08407×10^{12}	−33.25	57.40
2×10^{-6}	23.33655×10^{6}	0.05817×10^{12}	12.95461×10^{6}	1.21248×10^{12}	−44.48	1984.37
2×10^{-7}	18.54950×10^{6}	0.08796×10^{12}	10.99942×10^{6}	3.53552×10^{12}	−40.70	3919.46
2×10^{-8}	17.95278×10^{6}	0.11270×10^{12}	10.92685×10^{6}	4.20896×10^{12}	−39.13	3634.65
2×10^{-10}	17.94192×10^{6}	0.11719×10^{12}	10.92592×10^{6}	4.29642×10^{12}	−39.10	3566.20
0	17.94192×10^{6}	0.11719×10^{12}	10.92592×10^{6}	4.29732×10^{12}	−39.10	3566.96

Note: The values in the column SQP-DDPG(%) denote the percentage difference between SQP-DDPG, i.e., a negative value is a reduction in the given estimate and a positive value is an increase in the given estimate.

Table 9: Implementation Shortfall for PG - (numerical optimisation & reinforcement learning)

	SQP		DDPG		SQP-DDPG(%)	
λ	$E[x]$(\$)	$V[x]$(\$)	$E[x]$(\$)	$V[x]$(\$)	$E[x]$(\$)	$V[x]$(\$)
2×10^{-4}	22.86745×10^{6}	0.00146×10^{12}	27.1032×10^{6}	0.02318×10^{12}	18.52	1487.67
2×10^{-5}	11.84754×10^{6}	0.00219×10^{12}	13.50905×10^{6}	0.25599×10^{12}	14.02	11589.04
2×10^{-6}	10.41311×10^{6}	0.00281×10^{12}	10.47344×10^{6}	0.59334×10^{12}	0.57	21015.30
2×10^{-7}	10.38352×10^{6}	0.00292×10^{12}	10.38984×10^{6}	0.67012×10^{12}	0.06	22849.31
2×10^{-8}	10.38322×10^{6}	0.00293×10^{12}	10.38887×10^{6}	0.67873×10^{12}	0.05	23064.84
2×10^{-10}	10.38322×10^{6}	0.00293×10^{12}	10.38886×10^{6}	0.67969×10^{12}	0.05	23097.61
0	10.38322×10^{6}	0.00293×10^{12}	10.38886×10^{6}	0.67970×10^{12}	0.05	23097.95

Note: The values in the column SQP-DDPG(%) denote the percentage difference between SQP-DDPG, i.e., a negative value is a reduction in the given estimate and a positive value is an increase in the given estimate.

Table 8 shows an average reduction in expected implementation shortfall for the TSLA stock of 37.22 % for DDPG agent. This result suggests that the reinforcement learning approach is more effective in terms of expectation than the numerical optimisation approach. However, this comes at a cost of variance, as the average increase in variance is 2374.54 %. This result is aligned with what was found in the optimal trajectories (Figures 15 to 18), namely, that the reinforcement learning approach places a lower value on risk compared to the numerical optimisation approach, and will try to liquidate holdings for a given risk-aversion coefficient closer to the naïve strategy. A further deep dive into the results shows that both the numerical optimisation and the reinforcement learning approach follow the inherent trading logic that as λ decreases expectation decreases and variance increases. Furthermore, an interesting observation is that for $\lambda = 2 \times 10^{-4}$ the variance estimate seems to suggest a reduction of 97.25 %, this estimate seems doubtful. An explanation for this estimate could stem from either or all of (i) randomness in the policy itself, i.e., randomness in the current environment observation to a probability distribution of the actions that must be undertaken by the agent, (ii) the initial state distribution of the environment or (iii) the transition dynamics of the environment, e.g., a high learning rate. These specific noises are oftentimes quite significant, which means that the gradient estimate from one trajectory can deviate far from the true gradient. In order to obtain a similar estimate of variance, as in the case of SQP, it was found that $\lambda = 2.5 \times 10^{-5}$ would generate a variance of 0.05964×10^{12}, which is roughly the same estimate.

Table 9, shows an average increase in expected implementation shortfall for the PG stock of 4.75 %. Thus, conflicting with the result for the high-volatility stock, the reinforcement learning approach is less effective in terms of expectation than the numerical optimisation approach. In spite of the poor, or equal, level of implementation shortfall, the DDPG agent does not seem to compensate for this in terms of variance, as the average increase in variance is 18028.77 % across all levels of risk-aversion.

Conclusively, for a high-volatility stock the reinforcement learning approach is able to minimise expected shortfall, but this comes at a cost of variance. Conversely, for a low-volatility stock, the reinforcement learning approach is at most able to equal the expected shortfall of the numerical optimisation approach, and for high levels of risk-aversion it performs significantly worse. These results suggest that the reinforcement learning approach inherently places less value on risk, and thus, tends to liquidate the portfolio holdings according to the naïve strategy, which by definition minimises expected shortfall, but without regard for variance. The overall results of the empirical analysis broaden the understanding of how the two methodologies perform when compared to the baseline case[37]. Specifically, the empirical test results suggest that the DDPG agent outperforms the numerical optimisation approach with regard to expected shortfall unless the volatility environment is low, which is aligned with the findings of Hendricks and Wilcox (2014). This is not particularly surprising, as the numerical optimisation approach first estimate and then optimise, which cannot correct any inaccuracy from estimation. In contrast, the DDPG agent learns while optimising, this means that for repeated interactions with the environment, the inaccuracy of prior estimates can be corrected and lead to a better control policy. This finding aligns with the prior research of Wang et al. (2023), who also find that their reinforcement learning algorithm outperforms when risk-aversion levels are high or when the volatility level is high.

[37]The baseline case can be regarded as an intermediate scenario situated between the two empirical cases with regard to volatility.

5.6.2 Distributional Examples

In the financial literature, when modelling, e.g., a security price authors often assume that the underlying variable follows a discrete arithmetic random walk and the innovations are drawn from a distribution with zero mean and unit variance, i.e., the standard normal distribution. In this work, according to equation (3.1), the security price evolves precisely according to this assumption. However, it has for a long time been known that this assumption is unrealistic, as plenty of statistical evidence has shown that markets experience extreme events more often than the normal distribution assumes (Kofman, 2015). Following McDonald (1996), this work employs Student's t-distribution (t-distribution), which is a continuous probability distribution that generalises the standard normal distribution. The t-distribution is like the normal distribution symmetric around zero and bell-shaped. However, for the t-distribution, the degrees of freedom, ν, define the amount of probability mass in the tails. For $\nu = 1$ the t-distribution is heavy-tailed and for $\nu \to \infty$ the t-distribution reduces to the standard normal distribution. For the distributional examples, this work models equation (3.1) where ε are drawn from a t-distribution with $\nu = 1$. Furthermore, the parameters are as in Table 2. As we have already established that the optimal trajectories are determined ahead of trading, and the parameters are as in the baseline case, it is only interesting to view the effect that the t-distributed innovations have on the expectation and variance of the implementation shortfall. In the following table, the expectation and variance for the numerical optimisation and reinforcement learning approach are presented. See Appendix 9.8 for figures of the learning episodes.

Table 10: Implementation Shortfall t-distributed innovations (numerical optimisation & reinforcement learning).

	SQP		DDPG		SQP-DDPG(%)	
λ	$E[x]$(\$)	$V[x]$(\$)	$E[x]$(\$)	$V[x]$(\$)	$E[x]$(\$)	$V[x]$(\$)
2×10^{-4}	2.50349×10^{6}	0.86633×10^{12}	2.43603×10^{6}	0.00021×10^{12}	-2.69	-99.97
2×10^{-5}	2.11800×10^{6}	0.87552×10^{12}	1.95352×10^{6}	0.00367×10^{12}	-7.76	-99.58
2×10^{-6}	1.13340×10^{6}	1.06598×10^{12}	0.40978×10^{6}	0.06803×10^{12}	-63.84	-93.61
2×10^{-7}	0.68764×10^{6}	1.63886×10^{12}	-1.02298×10^{6}	2.80461×10^{12}	-248.78	71.13
2×10^{-8}	0.66287×10^{6}	1.87100×10^{12}	-1.30223×10^{6}	5.12148×10^{12}	-296.45	173.72
2×10^{-10}	0.66250×10^{6}	1.90561×10^{12}	-1.30076×10^{6}	13.68729×10^{12}	-296.34	618.26
0	0.66250×10^{6}	1.90561×10^{12}	-1.38971×10^{6}	11.69195×10^{12}	-309.76	513.55

Note: The values in the column SQP-DDPG(%) denote the percentage difference between SQP-DDPG, i.e., a negative value is a reduction in the given estimate and a positive value is an increase in the given estimate.

In Table 10, it is found that on average the DDPG agent is able to reduce expected implementation shortfall by 175 % compared to the numerical optimisation approach. This result corroborates the findings in the previous sections, namely, that the reinforcement learning approach is better, or, at least as good as the numerical optimisation approach in terms of expectation. However, this again comes at a cost of variance, as the average increase in variance is 154.74 %. Again this is not surprising, as it has previously been established that DDPG agent inherently places less value on risk.

Furthermore, as $\lambda \to 0$, the numerical optimisation method decreases the expectation and increases the variance of the implementation shortfall. This aligns with the baseline scenario, where the expectation of the shortfall matches that of the shortfall in the case of t-distributed innovations, but exhibits a higher variance in the case of the t-distributed innovations. The higher variance is a direct consequence of the assumption of t-distributed innovations, and thus, the trading model still captures the mean-variance properties for this special, but more realistic, case of price movements. However, an intriguing finding emerges from the analysis, indicating that the DDPG agent exhibits the capability to generate a negative implementation shortfall, denoting a scenario where the realised execution price of a security exceeds the benchmark price at the time of the trade[38]. This observation may be attributed to the increased likelihood of extreme price movements, which are more prevalent due to the t-distributed innovations. On the other hand, this observation may be the result of the reward definition, which is a proxy for the objective function stated in the original framework by Almgren and Chriss (2000). However, it was previously established that the DDPG agent mimicked the results of the numerical optimisation approach in the baseline case, and consequently, it is not unreasonable to assume that the DDPG agent has the ability to learn from prior estimates, and thus, generate negative implementation shortfalls for some levels of risk-aversion.

[38]Practically, negative implementation shortfall can arise, e.g., if a trader puts in a sell order and the price of the stock increases.

6 Discussion

In this section, this work addresses issues regarding the theoretical and methodological framework and topics for further research. Firstly, a methodological consideration is described, namely, the choice of how to estimate the expected loss from trading. Following this, a discussion of the value of information, i.e., a theoretical consideration is discussed and suggestions for future work are described. Lastly, this work discusses the implementation of multiple securities in the trading model and suggests that this should be investigated in future research.

6.1 Ex-Post Portfolio Performance Evaluation

In this work, the ex-post portfolio performance evaluation is based on implementation shortfall. This measurement tool is purely cost-driven, and thus, the algorithm that seeks to minimise objective function can sometimes take a long time, as it is trying to strike the right balance between the market impact and timing risk. This often means that the algorithm will take as long as necessary to prevent significant market impact. Even though this is the preferred methodology by Almgren and Chriss (2000), one could argue for using the volume-weighted average price (VWAP) or the time-weighted average price (TWAP) as a means for portfolio evaluation. VWAP is an average price that accounts for the traded volume during a time period. This is computed as the traded value divided by the traded volume, i.e., $x_n \times S_n/v$, where v is the traded volume. TWAP is the average price that takes into account the time evolution of prices during one trading period. This method is a natural extension of order-slicing strategies, i.e., dividing a large order into smaller series of orders. A TWAP trade buys, or sells, uniformly through the trading period. Both VWAP and TWAP are impact-driven algorithms and are simple to calculate and computationally inexpensive to execute compared to implementation shortfall. However, this does come at a cost in terms of the consistency of the estimates, as the VWAP and TWAP rely on the trader being significantly influenced by spreading the trade over the trading period. If this is not the case, the trader could benefit from trading the stock immediately. In future research, it would be interesting to incorporate VWAP and TWAP as additional ex-post performance evaluations, and how sensitive the added benchmarks are to the risk-aversion level of the trader.

6.2 The Value of Information

In this thesis, the optimal execution has been discussed under the assumption that the price dynamics follow an arithmetic random walk with zero drift. Thus, past price evaluation provides no information about future price movements, hence, optimal execution trajectories can be statically determined regardless of the methodology. However, there are relevant ways that a random walk with zero drift may fail to correctly represent the underlying price process. Firstly, the price process may have drift. For example, if a trader has a strong directional view of the price process, the trader will try to incorporate this view into the liquidation strategy. Moreover, the price process may exhibit serial correlation, i.e., price movements in a given period provide nontrivial information regarding the next-period movement of the stock. Lastly, at the start of trading, it may be known that at some specific point in time, an event will take place whose outcome will cause a shift in the parameters governing the price process[39] For example, Brown et al. (1988), shows that events cause temporary shifts in both risk and return of individual securities, and these shifts depend upon the outcome of the event. In particular, stocks react more strongly to bad news than to good news. In future research, it would be interesting to incorporate these dynamics into the price process. This could be done by introducing three new information parameters separately, or together, to the underlying price process: (i) a drift term[40], (ii) including serial correlation in the innovations and (iii) and a parameter shift.

6.3 Multiple-Security Portfolios

In this thesis, the trading strategy is solved for a trader holding one security, but the framework can be extended to consider more general buy/sell programmes. However, this would require a vectorisation of the trading model, and thus also, the analytical solution. Almgren and Chriss (2000) briefly consider an example with only two securities. In this example, the first security is identical to the security in the single-security framework and the second security is more liquid and less volatile. Additionally, the two securities have a moderate amount of correlation. In this example, the trajectory of security 1 is almost identical to its trajectory in the absence of security 2. Moreover, they find that the interdependence of their trajectories increases as the correlation of the securities increase. In future work, it could be interesting to explore the multiple-security framework for the different methodologies.

[39]Such events could include quarterly and annual earnings announcements, dividend announcements and share repurchases.

[40]Note, if the drift is linear the price process evolves according to a Geometric Brownian Motion.

7 Conclusion

This thesis theoretically and empirically studies the optimal execution of portfolio transactions based on the trading model proposed by Almgren and Chriss (2000). It compares four approaches to solving the trading model, namely, analytically, using numerical optimisation, using dynamic programming and through reinforcement learning. This work derives the explicit analytical solution for the risk-neutral trader. Furthermore, this thesis replicates and validates the numerical optimisation results presented by Almgren and Chriss (2000). Moreover, this thesis solves the trading model using dynamic programming and reinforcement learning to offer alternative insights and potential enhancements. Lastly, this work provides empirical examples utilising Telsa Inc. and The Procter & Gamble Company stock data, and distributional examples using the t-distribution. Thereby, the thesis provides a comprehensive understanding of the optimal execution of portfolio transactions, in the methodological domain.

Following that a central feature of the analysis has been the construction of an efficient frontier, the visualisation of the optimal trajectories of trade and the expectation and variance of the implementation shortfall. This thesis has derived the analytical solution for the framework and has been able to replicate the efficient frontier, optimal trajectories and the implementation shortfall following the baseline parameters from Almgren and Chriss (2000). The numerical optimisation approach yielded results that were aligned with the findings of Almgren and Chriss (2000), and thus, ensured the validity of the implementation of the trading model. Moreover, for this approach, the risk-loving agent was found to be non-optimal, as the agent would incur a higher expected implementation shortfall for a higher level of variance. The dynamic programming approach showed results that are aligned with the two preceding methodologies, however, with the caveat that the methodology could not sustain large portfolios. Therefore, this thesis analysed the temporary market impact parameter and it was shown to have an effect on the optimal trajectories and the implementation shortfall. As a consequence of these factors, this work suggests that the dynamic programming approach is infeasible for the trading framework. Solving the trading model through reinforcement learning for the baseline case, it was found that the agent was able to accurately capture the mean-variance properties of the trading model. Both the efficient frontier and the optimal trajectories were aligned with the theory of the model. The reinforcement learning approach is able to generate the smallest expected shortfall for large values of risk-aversion, but comes at a cost of variance. The empirical analysis shows that the reinforcement learning approach is the best candidate in terms of the

expected shortfall in high-volatility stocks, but again at a cost of variance. However, for low-volatility stocks, the numerical optimisation approach performs better. The distributional examples show that t-distributed innovations lead to higher variance for the numerical optimisation and reinforcement learning approach. However, it is found that reinforcement learning methodology exhibits the capability to generate a negative implementation shortfall, this is concluded to be attributed to the definition of the reward function.

This thesis proposes to extend the research to include additional ex-post portfolio performance evaluation measurements, such as the volume-weighted average price and the time-weighted average price, to understand the sensitivity level of the approaches in terms of risk-aversion level. Moreover, it is proposed to incorporate the value of information with regard to the underlying assumption of the price process of the asset. It is suggested to add a drift term, add serial correlation to the innovation terms and a parameter shift. With these additions, it would be interesting to assess how the costs are affected and their significance for the overall results of the trading model. Lastly, it is recommended for future research to explore the different approaches using multiple securities.

8 References

Almgren, R., & Chriss, N. (2000). Optimal Execution of Portfolio Transactions. *Journal of Risk*, *3*, 5–39. https://doi.org/10.4236/jmf.2015.51001

Almgren, R., Thum, C., Hauptmann, E., & Li, H. (2005). *Direct Estimation of Equity Market Impact*.

Amihud, Y., & Mendelson, H. (1986). Asset pricing and the bid-ask spread. *Journal of Financial Economics*, *17*(2), 223–249. https://doi.org/10.1016/0304-405X(86)90065-6

Ball, J. M. (1976). Convexity conditions and existence theorems in nonlinear elasticity. *Archive for Rational Mechanics and Analysis*, *63*(4), 337–403. https://doi.org/10.1007/BF00279992

Bellman, R. E. (1954). *The Theory of Dynamic Programming*.

Bertsekas, D. P. (1976). *Dynamic Programming: Determninistic and Stochastic Models*. Academic Press.

Bertsimas, D., & Lo, A. W. (1998). Optimal control of execution costs. *Journal of Financial Markets*, *1*, 1–50.

Brogaard, J., Hendershott, T., & Riordan, R. (2017). High frequency trading and the 2008 short-sale ban. *Journal of Financial Economics*, *124*(1), 22–42. https://doi.org/10.1016/j.jfineco.2017.01.008

Brogaard, J., Hendershott, T., & Riordan, R. (2014). High-Frequency Trading and Price Discovery. *Review of Financial Studies*, *27*(8), 2267–2306. https://doi.org/10.1093/rfs/hhu032

Brown, K. C., Harlow, W., & Tinic, S. M. (1988). Risk aversion, uncertain information, and market efficiency. *Journal of Financial Economics*, *22*(2), 355–385. https://doi.org/10.1016/0304-405X(88)90075-X

Colliard, J.-E., Foucault, T., & Lovo, S. (2022). Algorithmic Pricing and Liquidity in Securities Markets. *SSRN Electronic Journal*. https://doi.org/10.2139/ssrn.4252858

Gatheral, J., & Schied, A. (2011). Optimal Trade Execution under Geometric Brownian Motion in the Almgren and Chriss Framework. *International Journal of Theoretical and Applied Finance*, *14*(3), 353–368. https://doi.org/https://doi.org/10.1142/9789814407892{_}0016

Grinold, R. C., & Kahn, R. N. (1990). *Active Portfolio Management: A Quantitative Approach for Providing Superior Returns and Controlling Risk* (Vol. 2).

Guéant, O., & Lehalle, C.-A. (2013). *General Intensity Shapes in Optimal Liquidation*. https://doi.org/https://onlinelibrary.wiley.com/doi/10.1111/mafi.12052

Hendricks, D., & Wilcox, D. (2014). A reinforcement learning extension to the Almgren-Chriss framework for optimal trade execution. *2014 IEEE Conference on Computational Intelligence for Financial Engineering & Economics (CIFEr)*, 457–464. https://doi.org/10.1109/CIFEr.2014.6924109

Horváth, L., & Kokoszka, P. (2012). *Inference for Functional Data with Applications* (1st ed., Vol. SSS, 200). http://www.springer.com/series/692

Judd, K. L. (1998). *Numerical methods in economics*. MIT Press.

Kofman, P. (2015). Fat Tails in Finance. In *Wiley encyclopedia of management* (pp. 1–5). John Wiley & Sons, Ltd. https://doi.org/10.1002/9781118785317.weom040038

Kuno, S., & Ohnishi, M. (2015). Optimal Execution in Illiquid Market with the Absence of Price Manipulation. *Journal of Mathematical Finance*, *05*(01), 1–14. https://doi.org/10.4236/jmf.2015.51001

Kyle, A. S. (1985). Continuous Auctions and Insider Trading. *Econometrica*, *53*(6), 1315. https://doi.org/10.2307/1913210

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., & Wierstra, D. (2016). Continuous control with deep reinforcement learning. http://arxiv.org/abs/1509.02971

Lusardi, A., Michaud, P.-C., & Mitchell, O. S. (2017). Optimal Financial Knowledge and Wealth Inequality. *Journal of Political Economy*, 431–477.

McDonald, J. B. (1996). 14 Probability distributions for financial models. https://doi.org/10.1016/S0169-7161(96)14016-5

Nevmyvaka, Y., Feng, Y., & Kearns, M. (2006). Reinforcement learning for optimized trade execution. *Proceedings of the 23rd international conference on Machine learning - ICML '06*, 673–680. https://doi.org/10.1145/1143844.1143929

Perold, A. F. (1988). The implementation shortfall: Paper versus reality. *The Journal of Portfolio Management*, (Spring), 4–9. https://jpm.pm-research.com/content/14/3

Pratt, J. W. (1964). Risk Aversion in the Small and in the Large. *Econometrica*, *32*(1/2), 122. https://doi.org/10.2307/1913738

Rust, J. (1994). STRUCTURAL ESTIMATION OF MARKOV DECISION PROCESSES. In *Handbook of econometrics* (pp. 3082–3143). https://doi.org/https://doi.org/10.1016/S1573-4412(05)80020-0

Sargent, T. J. (1987). *Dynamic Macroeconomic Theory*. Harvard University Press.

Stokey, N. L., & Lucas, R. E. (1989). *Recursive Methods in Economic Dynamics*. Harvard University Press.

Sutton, R. S., & Barto, A. G. (2018). *Reinforcement Learning: An Introduction* (2nd ed.). The MIT Press.

Wang, B., Gao, X., & Li, L. (2023). Reinforcement Learning for Continuous-Time Optimal Execution: Actor-Critic Algorithm and Error Analysis. *SSRN Electronic Journal*. https://doi.org/10.2139/ssrn.4378950

9 Appendices

9.1 Appendix - A

Algorithm 1 DDPG algorithm

Randomly initialise critic network $Q(s, a|\theta_Q)$ and actor $\mu(s|\theta_\mu)$ with weights θ_Q and θ_μ.
Initialise target network Q' and μ' with weights $\theta_{Q'} \leftarrow \theta_Q$ and $\theta_{\mu'} \leftarrow \theta_\mu$.
Initialise replay buffer R.
for episode = 1 to M **do**
    Initialise a random process N for action exploration.
    Receive initial observation state s_1.
    for $t = 1$ to T **do**
        Select action $a_t = \mu(s_t|\theta_\mu) + N_t$ according to the current policy and exploration noise.
        Execute action a_t and observe reward r_t and observe new state s_{t+1}.
        Store transition (s_t, a_t, r_t, s_{t+1}) in R.
        Sample a random minibatch of N transitions (s_i, a_i, r_i, s_{i+1}) from R.
        Set $y_i = r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1}|\theta_{\mu'}))|\theta_{Q'}$.
        Update critic by minimising the loss: $L = \frac{1}{N}\sum_i (y_i - Q(s_i, a_i|\theta_Q))^2$.
        Update the actor policy using the sampled policy gradient:
        $\nabla_\theta \mu J \approx \frac{1}{N}\sum_i \nabla_a Q(s, a|\theta_Q)|_{s=s_i, a=\mu(s_i)} \nabla_\theta \mu(s|\theta_\mu)|_{s_i}$.
        Update the target networks:
        $\theta_{Q'} \leftarrow \tau\theta_Q + (1-\tau)\theta_{Q'}$.
        $\theta_{\mu'} \leftarrow \tau\theta_\mu + (1-\tau)\theta_{\mu'}$.
    end for
end for

Table 11: Deep Learning Design

Parameter	**Value**	**Description**
Buffer Size	1×10^4	Replay buffer size
Batch Size	128	Minibatch size
γ	0.99	Discount factor
τ	1×10^{-3}	Soft update of target parameters
LR Actor	1×10^{-4}	Learning rate of the actor
LR Critic	1×10^{-3}	Learning rate of the critic
Weight Decay	0	L2 weight decay

9.2 Appendix - B

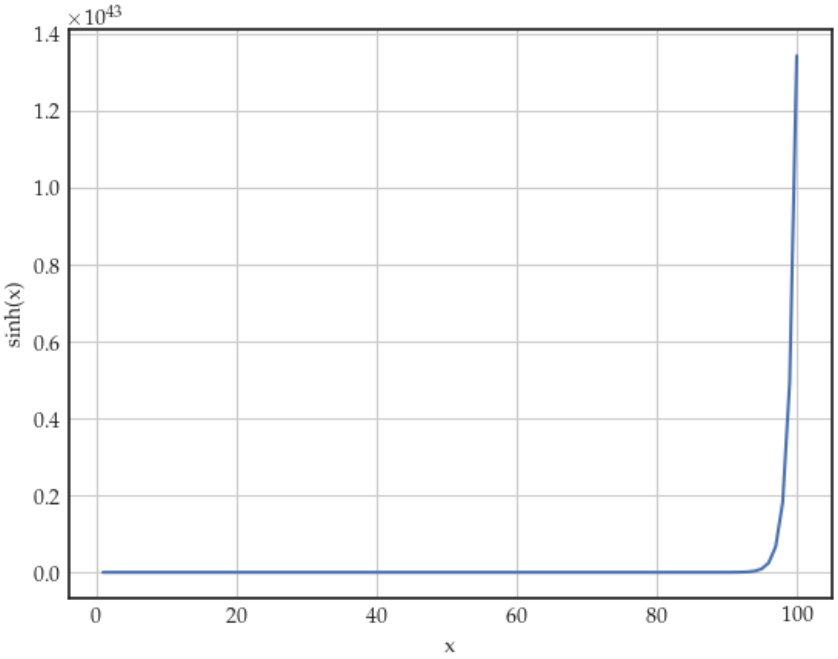

Figure 19: Evolution of $\sinh(x)$ for values between 1 and 100

9.3 Appendix - C

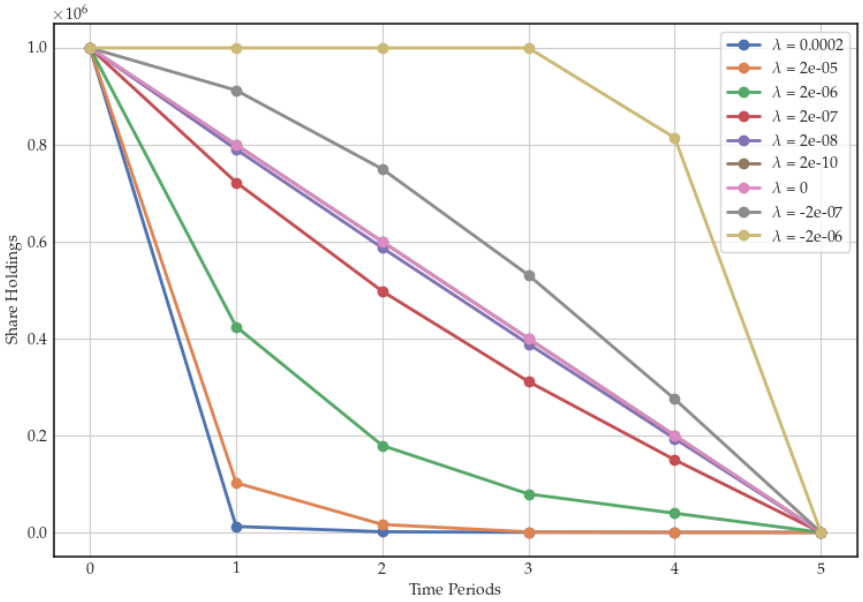

Figure 20: Optimal trajectories (numerical). Supplementary trajectories.

9.4 Appendix - D

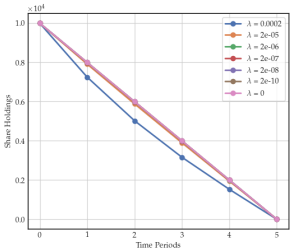

Figure 21: The optimal trajectories (dynamic programming). The parameters are as in Table 2. $X = 10{,}000$ and $\eta = 2 \times 10^{-3}$

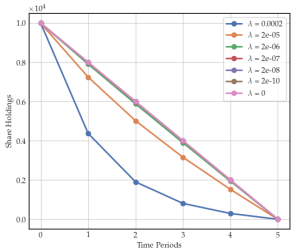

Figure 22: The optimal trajectories (dynamic programming). The parameters are as in Table 2. $X = 10{,}000$ and $\eta = 2 \times 10^{-4}$

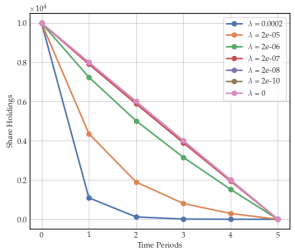

Figure 23: The optimal trajectories (dynamic programming). The parameters are as in Table 2. $X = 10{,}000$ and $\eta = 2 \times 10^{-5}$

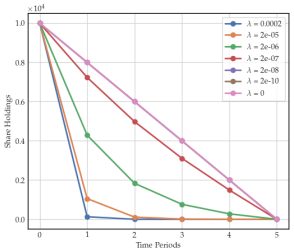

Figure 24: The optimal trajectories (dynamic programming). The parameters are as in Table 2. $X = 10{,}000$ and $\eta = 2 \times 10^{-6}$

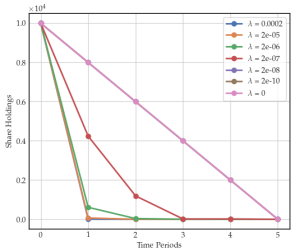

Figure 25: The optimal trajectories (dynamic programming). The parameters are as in Table 2. $X = 10{,}000$ and $\eta = 2 \times 10^{-7}$

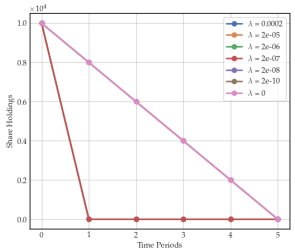

Figure 26: The optimal trajectories (dynamic programming). The parameters are as in Table 2. $X = 10{,}000$ and $\eta = 2 \times 10^{-8}$

9.5 Appendix - E

Table 12: Average Shortfall per Episode

Episode	Average Shortfall
100/1000	$2,061,800.44
200/1000	$2,559,876.16
300/1000	$2,562,500.00
400/1000	$2,562,500.00
500/1000	$2,562,500.00
600/1000	$2,562,500.00
700/1000	$2,562,500.00
800/1000	$2,562,500.00
900/1000	$2,562,500.00
1000/1000	$2,562,500.00

Table 13: Average Implementation Shortfall

Metric	Value
Average Implementation Shortfall	$2,512,167.66

9.6 Appendix - F

	Analytical		SQP		DP		DDPG	
λ	$E[x](\$)$	$V[x](\$)$	$E[x](\$)$	$V[x](\$)$	$E[x](\$)$	$V[x](\$)$	$E[x](\$)$	$V[x](\$)$
2×10^{-4}	2.56194×10^6	0.00000×10^{12}	2.50496×10^6	0.00000×10^{12}	2.52442×10^6	0.00000×10^{12}	2.50245×10^6	0.00015×10^{12}
2×10^{-5}	2.29622×10^6	0.00318×10^{12}	2.12007×10^6	0.00673×10^{12}	2.13713×10^6	0.00376×10^{12}	2.11089×10^6	0.00995×10^{12}
2×10^{-6}	1.17952×10^6	0.19231×10^{12}	1.14793×10^6	0.20559×10^{12}	1.10946×10^6	0.20137×10^{12}	1.14594×10^6	0.19773×10^{12}
2×10^{-7}	0.86947×10^6	0.84506×10^{12}	0.68821×10^6	1.00068×10^{12}	0.86027×10^6	0.85252×10^{12}	0.69526×10^6	0.78552×10^{12}
2×10^{-8}	0.93834×10^6	1.12014×10^{12}	0.66289×10^6	1.33299×10^{12}	0.91838×10^6	1.09810×10^{12}	0.66992×10^6	1.02543×10^{12}
2×10^{-10}	0.95234×10^6	1.15958×10^{12}	0.66250×10^6	1.38400×10^{12}	0.92949×10^6	1.13178×10^{12}	0.66954×10^6	1.05949×10^{12}
0	0.71250×10^6	1.20000×10^{12}	0.66250×10^6	1.38400×10^{12}	0.72057×10^6	1.21359×10^{12}	0.66954×10^6	1.05984×10^{12}

Table 14: Implementation Shortfall. The parameters are as in Table 2. 0.00000 estimates are not literal zero, but zero to the fifth decimal point.

9.7 Appendix - G

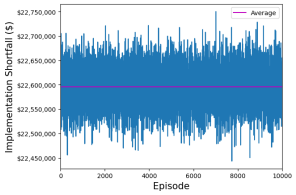

Figure 27: Implementation Shortfall TSLA (RL). The parameters are as in Table 7. $X = 10{,}000$ and $\lambda = 2 \times 10^{-4}$

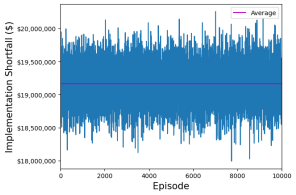

Figure 28: Implementation Shortfall TSLA (RL). The parameters are as in Table 7. $X = 10{,}000$ and $\lambda = 2 \times 10^{-5}$

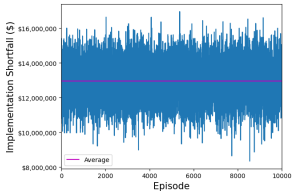

Figure 29: Implementation Shortfall TSLA (RL). The parameters are as in Table 7. $X = 10{,}000$ and $\lambda = 2 \times 10^{-6}$

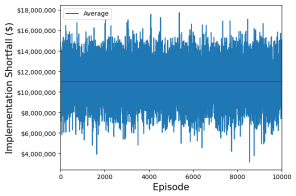

Figure 30: Implementation Shortfall TSLA (RL). The parameters are as in Table 7. $X = 10{,}000$ and $\lambda = 2 \times 10^{-7}$

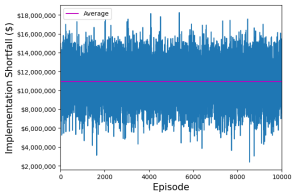

Figure 31: Implementation Shortfall TSLA (RL). The parameters are as in Table 7. $X = 10{,}000$ and $\lambda = 2 \times 10^{-8}$

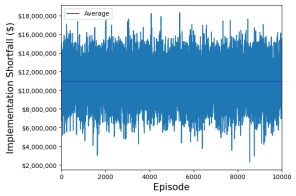

Figure 32: Implementation Shortfall TSLA (RL). The parameters are as in Table 7. $X = 10{,}000$ and $\lambda = 2 \times 10^{-10}$

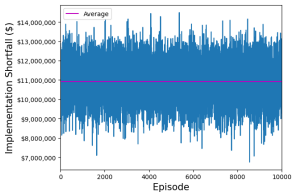

Figure 33: Implementation Shortfall TSLA (RL). The parameters are as in Table 7. $X = 10{,}000$ and $\lambda = 0$

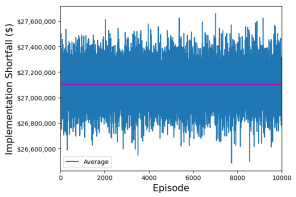

Figure 34: Implementation Shortfall PG (RL). The parameters are as in Table 7. $X = 10{,}000$ and $\lambda = 2 \times 10^{-4}$

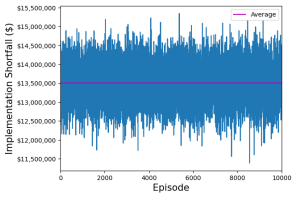

Figure 35: Implementation Shortfall PG (RL). The parameters are as in Table 7. $X = 10{,}000$ and $\lambda = 2 \times 10^{-5}$

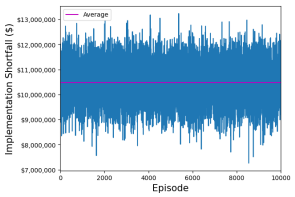

Figure 36: Implementation Shortfall PG (RL). The parameters are as in Table 7. $X = 10{,}000$ and $\lambda = 2 \times 10^{-6}$

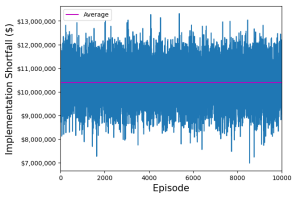

Figure 37: Implementation Shortfall PG (RL). The parameters are as in Table 7. $X = 10{,}000$ and $\lambda = 2 \times 10^{-7}$

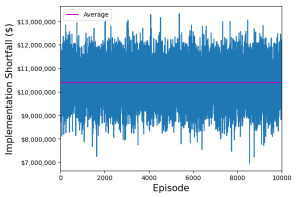

Figure 38: Implementation Shortfall PG (RL). The parameters are as in Table 7. $X = 10{,}000$ and $\lambda = 2 \times 10^{-8}$

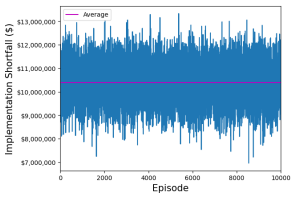

Figure 39: Implementation Shortfall PG (RL). The parameters are as in Table 7. $X = 10{,}000$ and $\lambda = 2 \times 10^{-10}$

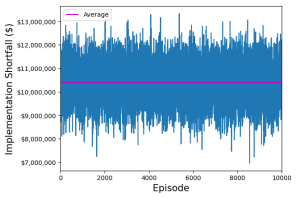

Figure 40: Implementation Shortfall PG (RL). The parameters are as in Table 7. $X = 10{,}000$ and $\lambda = 0$

9.8 Appendix - H

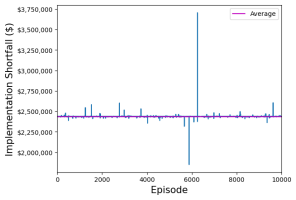

Figure 41: Implementation Shortfall t-distributed innovations (RL). The parameters are as in Table 2. $X = 10{,}000$ and $\lambda = 2 \times 10^{-4}$

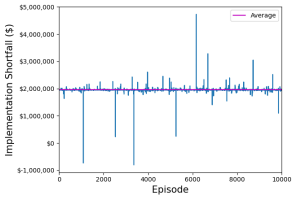

Figure 42: Implementation Shortfall t-distributed innovations (RL). The parameters are as in Table 2. $X = 10{,}000$ and $\lambda = 2 \times 10^{-5}$

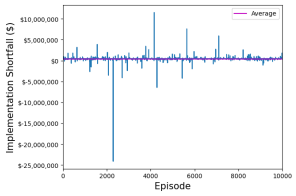

Figure 43: Implementation Shortfall t-distributed innovations (RL). The parameters are as in Table 2. $X = 10{,}000$ and $\lambda = 2 \times 10^{-6}$

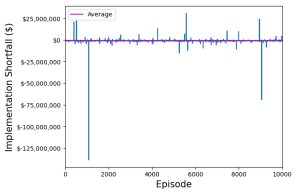

Figure 44: Implementation Shortfall t-distributed innovations (RL). The parameters are as in Table 2. $X = 10{,}000$ and $\lambda = 2 \times 10^{-7}$

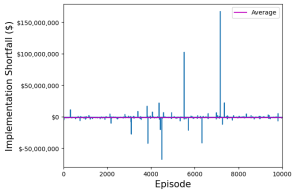

Figure 45: Implementation Shortfall t-distributed innovations (RL). The parameters are as in Table 2. $X = 10{,}000$ and $\lambda = 2 \times 10^{-8}$

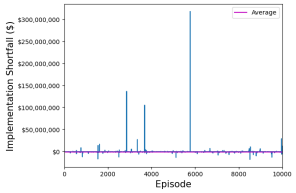

Figure 46: Implementation Shortfall t-distributed innovations (RL). The parameters are as in Table 2. $X = 10{,}000$ and $\lambda = 2 \times 10^{-10}$

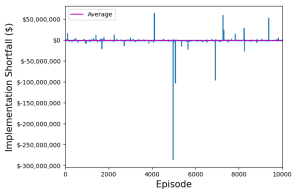

Figure 47: Implementation Shortfall t-distributed innovations (RL). The parameters are as in Table 2. $X = 10{,}000$ and $\lambda = 0$