Optimal Trading Under Non-negativity Constraints Using Approximate Dynamic Programming

Shahin Abbaszadeh\textsuperscript{1}, Tri-Dung Nguyen\textsuperscript{1,2*}, Yue Wu\textsuperscript{1}

\textsuperscript{1}Management School, University of Southampton
\textsuperscript{2}School of Mathematical Sciences, University of Southampton

Abstract

In this paper we develop an extended dynamic programming (DP) approach to solve the problem of minimising execution cost in block trading of securities. To make the problem more practical, we add non-negativity constraints to the model and propose a novel approach to solve the resulting DP problem to near-optimal results. We also include time lags in the problem state to account for the autoregressive behaviour of most financial securities as a way of increasing problem sensitivity to variability of prices and information. The computation times achieved for the proposed algorithm are fast and allow for the possibility of live implementation. We demonstrate the benefits offered by the new approach through numerical analysis and simulation runs in comparison to the classic model without the non-negativity constraints.

Keywords: Dynamic programming, Finance, Optimisation, Stochastic processes.

1 Introduction

1.1 Background and Literature Review

The growth in equity trading in recent decades, which has been largely due to the ever-increasing amounts of funds available to institutional investors such as pension and mutual funds, has triggered an interest in more effective management of trading costs. These costs are often called transaction costs or execution costs, which include commissions, bid/ask spreads, opportunity costs of waiting, and price impact from trading. Loeb (1983) was among the first to recognise the importance of execution costs and discusses different aspects of trading cost with relation to capitalisation and spread of the stock and the funds available to the trader. Perold (1988), among others, documents that hypothetical portfolios or passive benchmarks constantly outperform the market and the portfolio manager by almost 20\% per year. Chan and Lakonishok (1993) argue that, since trading in equity markets is increasingly dominated by institutional investors, the implementation shortfall that Loeb (1983) and Perold (1988) discuss may be due to the costliness of executing individual trade transactions that result in a more costly overall execution of the order. This overall transaction cost prompts the traders to break their order down into smaller transaction units, which is then executed over a certain time period. A trading strategy that minimises the expected execution cost of the trade is defined as best execution strategy.

Chan and Lakonishok (1993) study the institutional effects of trade on equity prices. They argue that since trading in equity markets is increasingly dominated by institutional investors,
the importance of transaction costs is increasing. They offer three explanations for price changes triggered by large trade volumes: (i) short-run liquidity costs that arise when there is not an immediate buyer or seller, and the attempt to attract the buyer or seller translates to a price concession, (ii) imperfect substitution where there is no particular stock as a substitute and the buyer offers a premium for a large transaction, and (iii) information effects where the amount of trade reveals information about the trade that is incorporated into the subsequent prices. This and many other studies document portfolio managers’ inability to outperform various passive benchmarks, despite considerable effort to analyse and select stocks.

Bertsimas and Lo (1998) were the first to consider the use of dynamic programming in acquiring the best execution strategy. They argue that the act of trading affects price dynamics that will result in changes to the future trading costs (since the demand for financial securities is not perfectly elastic, the price impact of current trades, however small, can affect the course of future prices). They also observe that trading takes time as large trades need to be executed over numerous periods. They propose and solve the dynamic trading problem with the use of dynamic programming framework. Bertsimas and Lo (1998) offer an analytic solution to the transaction cost problem in the case of a single stock. They solve the problem using dynamic programming methods and offer numerical analysis based on simulation runs on a scenario representing a real problem. Bertsimas et al. (1999) apply a similar method to the single stock model presented by Bertsimas and Lo (1998) to address the portfolio case. They also develop a new price impact model and an approximate model to solve the problem with non-negativity constraints for a specific case.

Chakravarty (2001) suggests that the medium-sized trading done by institutions (which he argues is the optimal way if the trader intends to sell large blocks of shares and prevent big shocks to the market, i.e. adverse price impact) has a disproportionate effect on the cumulative price change compared to trading in big chunks. Alexander and Peterson (2007) study the effects of trade clustering on various parts of the market. They argue that the sizes of the clusters tend toward 100, 500 and 5000, which are mostly used by highly informed or stealth traders. They study the effects of clustering and block trading on the price of the underlying stock and conclude that the price impact from medium-sized clusters is much higher than the small- or large-sized clusters. Domowitz and Yegerman (2005) review some of the early algorithmic trading services and give a brief result as to their efficiency compared to manual and other types of trading considering parameters such as trade size and type of security, among others. Butenko et al. (2005) provide models and algorithms for the problem of liquidating a certain amount of securities with or without the consideration of risk as a factor in decision making. Engle and Ferstenberg (2006) propose an extension to the execution cost problem by incorporating the “risk return” trade-off into the problem. They achieve this by combining the transaction cost model with the portfolio planning problem.

Almgren and Chriss (2001) examine the relation between the risk (different levels of liquidity and the trader’s idea of this) and optimal execution strategy. They obtain closed-form solutions for the optimal trading strategy for any level of risk-aversion from the trader. Almgren (2003) offers an update on the Almgren and Chriss (2001) modelling of the portfolio stock trading costs by introducing a more robust price impact model. He incorporates trading-enhanced risk, an additional volatility measure that corresponds to the change in price following the demand in a more rapid execution of large blocks (liquidity premium demanded by the market is not deterministic). Kissell et al. (2004) build on the work of Bertsimas and Lo (1998) and Almgren and Chriss (2001); they offer a more detailed breakdown of price change sources and analyse the causal relationship between these. They offer best execution strategy for three different scenarios: cost minimisation; balancing the risk vs cost; and price improvement. Subramanian and Sherali (2010) quantify the liquidity risk which corresponds to the difference between market price and realisation price from a trader’s position. They find an optimality condition for block trading.
He and Mamaysky (2005) offer an optimal execution strategy which is similar to that of Bertsimas and Lo (1998), but present a different price impact model based on Merton (1971). Huberman and Stanzl (2005) consider a linear price impact and argue that in a market without arbitrage opportunity, their proposed linear price impact model is optimal. They extend the work of Bertsimas and Lo (1998) by considering a risk-averse trader. Hasbrouck and Seppi (2001) study various price dynamics along with other issues related to liquidity such as focusing on the market as a whole and the effect of inter-company trades on the liquidity of a security. Kissell and Malamut (2006) introduce a framework that can accommodate different aspects of trading behaviour. A system that can act aggressively/patiently at times of favourable/unfavourable price changes would benefit investors greatly. The criteria also include the changes in the dynamics of price fluctuations as well as the different initial prices the algorithms can take.

1.2 Contributions and Paper Structure

Although Bertsimas and Lo (1998) obtain the closed-form solution and the resulting trade strategy is the optimal execution strategy, it does not take into account the non-negativity constraints which would lead to a short-selling situation when in practice it is not possible to do so. If the price change is severe in an otherwise normal trade scenario, during a buy operation it might be optimal to sell and vice versa, while in practice the trader will not be allowed to operate based on that insight. This shortcoming might not affect a large portion of daily trades, but since it is a significant probability when trading large volumes of securities in a volatile market, the expected cost of ignoring the non-negativity constraints is still considerable. We develop an approximate dynamic programming approach to circumvent this costly possible scenario.

Our first contribution to the literature is the inclusion of non-negativity constraints in the formulation. To the best of our knowledge, the inclusion of non-negativity constraints has not been treated in the relevant literature. This constraint, however, increases the complexity of this problem substantially, as dynamic programming does not lend itself readily to constraints. We present here an approximate dynamic programming approach that enables us to solve a large combination of problems to near-optimality through a generalised platform.

Secondly, we contribute to the literature through development of a bespoke Approximate Dynamic Programming method that combines both above contributions into a more complex problem. Our method offers a generalised platform through which a large combination of problems can be solved fast and near-optimally.

The rest of this paper is structured as follows: we first present the modelling approach and results outlined by Bertsimas and Lo followed by our proposed extensions to their model in section 2. We then propose an approximate dynamic programming method in section 3 that handles the added sign constraints to the problem. We present the results and insights gained through numerical analysis in section 4 followed by concluding remarks and a look at possible future directions for this research.

2 Models for Optimal Trade Execution

2.1 Basic Model

Consider a situation where a trader wants to buy a number $S$ of shares in $T$ consecutive periods of equal length. It is assumed that the price dynamics are known and the price $p$ at period $t$ follows an autoregressive process where it is related to the prices in the previous periods. It is also assumed that the effect of trade volume on the price is known to the trader. An additional information source is assumed, reflecting any complementary data the decision maker might use to infer the price behaviour. This additional source of information, denoted henceforth by $x_t$, might be an index of the market where the share is traded or expert knowledge available to the
trader.

Consider \( s_t \) to denote the current number of stocks available at each stage and \( u_t \) to be the decision variable which is the trade volume in the current period. Based on this information the decision maker wants to optimise the number of shares traded in each period in order to minimise the overall cost of execution of the \( S \) shares within the \( T \) sequential periods.

The general form of the basic problem (Bertsimas and Lo, 1998) is

\[
\min \ E \left[ \sum_{t=1}^{T} p_{t+1} u_t \right],
\]

s.t. \( \sum_{t=1}^{T} u_t = S \),

(1)

\( s_1 = S, \ s_{T+1} = 0 \),

\( s_{t+1} = s_t - u_t, \ t = 1, 2, ..., T - 1 \)

(2)

\( x_{t+1} = \rho x_t + \eta_t, \ t = 1, 2, ..., T \)

(3)

\( p_t = p_t + \beta x_t + a u_t + \epsilon_t, \ t = 1, 2, ..., T \)

(4)

where \( \eta_t \) and \( \epsilon_t \) are independent white noise processes with mean 0 and variance \( \sigma_\eta^2 \) and \( \sigma_\epsilon^2 \) respectively. \( a \) is assumed to be positive and \( \rho \) to be bounded between 1 and -1, and \( \beta \) is the effect of information on price dynamics and is a given at the start of the optimisation period. Constraints (1) and (2) explain the dynamics of the stock at hand and ensure that all the shares are executed during the \( T \) periods. Constraint (3) is to ensure that the number of stocks (to execute) in the coming period \( (s_{t+1}) \) is the current number of stocks available \( (s_t) \) minus the trade volume in the current period; and constraints (4) and (5) express the dynamics of information and price evolution accordingly. Finally, the objective function is the expected cost over the trading period.

Throughout this study the execution is assumed to be a buy operation. However, the results would be applicable to a sell operation without any loss in generality. The objective of such a problem would be to maximise revenue. Another point to be noted is that while the occurrence of negative prices in the model is possible, we implicitly assume that the parameters of the model are of such values as to make this unlikely. Thus we treat the model without such limitation.

2.2 Extending the Basic Model

An important aspect of the basic model is the absence of non-negativity constraints. Based on the variance of the information variable and also the price, the optimal solution might suggest a negative trade (i.e. sale in a buy operation). In a real situation where the short-selling option is not available, a naive strategy is to change the negative trades to zero. However, such a solution would significantly reduce the intended benefits of the model - e.g. minimisation of transaction costs.

In the coming section we address these limitations by offering a flexible framework that provides near-optimal results while handling the added complexity of non-negativity constraints, and offer a generalised platform from which a large combination of problems can be solved fast and near-optimally.

We first extend the model in vector form and present the closed-form solution in line with the results of Bertsimas and Lo (1998). We augment the state space as

\[
\begin{bmatrix}
    p_{t+1} \\
    s_{t+1} \\
    x_{t+1} \\
    y_{t+1}
\end{bmatrix}
= \begin{bmatrix}
    1 & 0 & \beta \\
    0 & 1 & 0 \\
    0 & 0 & \rho \\
    A
\end{bmatrix}
\begin{bmatrix}
    p_t \\
    s_t \\
    x_t
\end{bmatrix}
+ \begin{bmatrix}
    a \\
    -1 \\
    0
\end{bmatrix}
\begin{bmatrix}
    u_t \\
    \eta_t
\end{bmatrix}
+ \begin{bmatrix}
    \epsilon_t \\
    \omega_t
\end{bmatrix}
\]

where \( A \) is a matrix that reflects the dynamics of the state variables in the current period. The objective function in this extended model becomes

\[
\min \ E \left[ \sum_{t=1}^{T} \right]
\]

subject to

(1)

(2)

(3)

(4)

(5)
The above formulation can be written as

\[ y_{t+1} = Ay_t + bu_t + \omega_t. \]

The cost at period \( t \) is

\[
g(y_t, u_t, \omega_t) = p_{t+1}u_t
\]

\[
= y_{t+1}^T e_1 u_t
\]

\[
= (Ay_t + bu_t + \omega_t)^T e_1 u_t
\]

\[
= y_t^T (A^T e_1) u_t + b^T e_1 u_t^2 + \omega_t^T e_1 u_t,
\]

where \( e_1 \) represents the first column of the identity matrix.

At the final stage (\( t = T \)), the value function is equal to the expected value of cost function with respect to \( \omega_T \). We substitute the decision variable \( u_t \) with \( s_T \), which is the only available decision at \( t = T \), and obtain

\[ u_T = s_T = e_1^T y_T. \]

Thus,

\[
V_T(y_T) = E[g(y_T, u_T, \omega_T)]
\]

\[ = E[y_T^T (A^T e_1) u_T + b^T e_1 u_T^2 + \omega_T^T e_1 u_T]
\]

\[ = y_T^T (A^T e_1) e_1 y_T + b^T e_1 (e_2 e_1^T) y_T
\]

\[ = y_T^T ((A^T e_1) e_1^T + b^T e_1 (e_2 e_1^T)) y_T
\]

\[ = y_T^T K_T y_T,
\]

where \( K_T = (A^T e_1) e_1^T + b^T e_1 (e_2 e_1^T) \).

For the second last period, i.e. stage \( (T - 1) \), we have

\[
V_{T-1}(y_{T-1}) = \min_{u_{T-1}} E[g(y_{T-1}, u_{T-1}, \omega_{T-1}) V_T(y_T)]
\]

\[ = \min_{u_{T-1}} E[y_{T-1}^T (A^T e_1) u_{T-1} + b^T e_1 u_{T-1}^2 + \omega_{T-1}^T e_1 u_T + y_T^T K_T y_T]
\]

\[ = \min_{u_{T-1}} E[y_{T-1}^T (A^T e_1) u_{T-1} + b^T e_1 u_{T-1}^2 + \omega_{T-1}^T e_1 u_T
\]

\[ + (Ay_{T-1} + bu_{T-1} + \omega_{T-1})^T K_T (Ay_{T-1} + bu_{T-1} + \omega_{T-1})]
\]

\[ = \min_{u_{T-1}} \left\{ (b^T e_1 + b^T K_T b) u_{T-1}^2 + y_{T-1}^T (A^T e_1 + A^T (K_T + K_T^T) b) u_{T-1}
\]

\[ + y_{T-1}^T K_T A y_{T-1} + E[\omega_{T-1}^T K_T \omega_{T-1}] \right\}.
\]

Since \( V_{T-1}(y_{T-1}) \) is a quadratic equation on \( u_{T-1} \) and we assume \( a > 0 \), we will have

\[ (b^T e_1 + b^T K_T b) > 0. \]

**Lemma 1.** \( \alpha_t = (b^T e_1 + b^T K_{t+1} b) > 0 \) is true for all \( t = 1, ..., T - 1 \).

**Proof.** We demonstrate this result in appendix A.1. \( \square \)

We acquire the optimal decision for period \( (T - 1) \) as

\[ u_{T-1}^* = \frac{-y_{T-1}^T L_{T-1}}{2\alpha_{T-1}}, \]

where \( L_{T-1} = A^T e_1 + A^T (K_T + K_T^T) b \), and \( \alpha_{T-1} = b^T e_1 + b^T K_T b. \)
We can then rewrite $V_{T-1}(y_{T-1})$ as
\[ V_{T-1}(y_{T-1}) = y_{T-1}^T K_{T-1} y_{T-1} + C_{T-1}, \]
where
\[ K_{T-1} = \left( A^T K_T A - \frac{L_{T-1} L_{T-1}^T}{4 \alpha_{T-1}} \right), \]
\[ C_{T-1} = E[\omega_{T-1}^T K_T \omega_{T-1}], C_T = 0. \]

To find the closed-form solution for the case of extended state, we assume that
\[ V_{t+1}(y_{t+1}) = y_{t+1}^T K_{t+1} y_{t+1} + C_{t+1}. \]

Then, we have
\[
V_t(y_t) = \min_{u_t} \mathbb{E}[g(y_t, u_t, \omega_t)] + V_{t+1}(y_{t+1}) \\
= \min_{u_t} \mathbb{E}[y_t^T (A^T e_1) u_t + b^T e_1 u_t^2 + y_{t+1}^T K_{t+1} y_{t+1} + C_{t+1}] \\
= \min_{u_t} \mathbb{E}[y_t^T (A^T e_1) u_t + b^T e_1 u_t^2 + (A y_t + b u_t + \omega_t)^T K_{t+1} (A y_t + b u_t + \omega_t) + C_{t+1}] \\
= \min_{u_t} \left\{ (b^T e_1 + b^T K_{t+1} b) u_t^2 + y_t^T (A^T e_1 + A^T (K_T + K_t^T) b) u_t \\
+ y_t^T A^T K_{t+1} A y_t + E[\omega_t^T K_{t+1} \omega_t] + C_{t+1} \right\}. 
\]

The above quadratic equation holds for all $t$, resulting in the optimal solution
\[ u_t^* = -\frac{y_t^T L_t}{2 \alpha_t}, \]
where
\[
L_t = A^T e_1 + A^T (K_{t+1} + K_{t+1}^T) b, \\
\alpha_t = b^T e_1 + b^T K_{t+1} b, \\
K_t = A^T K_{t+1} A - \frac{L_t L_t^T}{4 \alpha_t}, \\
K_T = (A^T e_1) e_2^T + b^T e_1 (e_2 e_2^T), \\
C_t = E[\omega_t^T K_{t+1} \omega_t] + C_{t+1}, \\
C_T = 0. 
\]

Given the values of $\beta, \alpha$ and $\rho$, we can calculate $K_t, L_t$ and $C_t$ offline for all stages $t = 1, 2, ..., T$ by starting from last period and working backwards, after which the optimal solution is obtained in the forward stage. Algorithm 1 depicts the general steps of this procedure.

The closed-form solution arrived at here is compatible with the results of Bertsimas and Lo (1998).

2.3 Including Non-negativity Constraints

The basic model in Bertsimas and Lo (1998) does not include non-negativity constraints to avoid negative trade volumes when the variations in the prices or the market warrant negative trading. An optimal policy that contains a negative trade in a block-trade buying operation would be void in real trading situations since it is counter-intuitive and is not allowed in many systems (Bertsimas and Lo, 1998). The inclusion of non-negativity constraints would improve the performance of the trade regime substantially. However, adding non-negativity constraints to a dynamic programming problem increases the complexity of the problem considerably. Bertsimas and Lo (1998) provide a closed-form solution for the problem with non-negativity constraints.
Algorithm 1: Closed-form solution in vector form

**input:** $\beta$, $a$ and $\rho$

**output:** $u^*$

\[
K_T = (A^T e_1^2 + b^T e_1 (e_2 e_1^T))
\]

for $t = (T - 1) \rightarrow 1$ do

\[
\begin{align*}
L_t &= A^T e_1 + A^T (K_{t+1} + K_{t+1}^T)b \\
\alpha_t &= b^T e_1 + b^T K_{t+1}b \\
K_t &= A^T K_{t+1} A - L_t L_t^T / 4\alpha_t
\end{align*}
\]

$y_1 = y_0$

for $t = 1 \rightarrow T$ do

\[
\begin{align*}
L_t &= L_{t-1}^T \\
u_t^* &= -y_t^T L_t / 2\alpha_t
\end{align*}
\]

when price dynamics follow a special pattern:

\[
p_t = \theta p_{t-1} + ax_t u_t + \epsilon_t,
\]

\[
log x_t = log x_{t-1} + \eta_t.
\]

However, as Huberman and Stanzl (2005) conclude, a linear price impact model is the most representative formulation for price impact in most real-world situations. The above special form of price evolution formula concerns only a limited case and is not an appropriate substitute for price impact in practical situations. In this paper we assume a linear price evolution model such as (5).

If we assume that non-negativity must hold, the optimal trade size is decided by

\[
u_t^* = \max(0, \min(s_t, -\frac{y_t^T L_t}{2\alpha_t}))
\]

resulting in three different possible outcomes for each period. In the context of dynamic programming, after each stage in the backward progress through algorithm, the state space of the problem grows threefold (up to $3^T$ in the final stage), which results in considerably larger problems. We discuss the solution approach for handling this in the next section.

### 3 Approximate Dynamic Programming

Approximate dynamic programming is a range of approximation tools that are devised to address the inherent problems of the dynamic programming framework in dealing with constraints. Classic approaches to dynamic programming are unable to deal with exponential growth of the computational requirements as the number of states increases. Unless the problems are defined to very restrictive assumptions, a simulation of the system is more easily constructed than a model. In this paper we employ a value function approximation method where we replace the piecewise value function which is resulted from the addition of non-negativity constraints with a single value function that best captures the characteristics of the three functions.

#### 3.1 An Approximated Value Function

In order to overcome the added complexity of the non-negativity constraints, we approximate the cost-to-go at period $(t+1)$ with a quadratic function,

\[
V_{t+1}(y_{t+1}) = y_{t+1}^T Q_{t+1} y_{t+1} + B_{t+1}^T y_{t+1} + C_{t+1},
\]
where $Q_t$ and $B_t$ are quadratic and linear coefficients of the value function during period $t$ respectively and $C_t$ is the constant term.

Following the Bellman procedure, we have

$$V_t(y_t) = \min_{u_t \leq y_t \leq s_t} \mathbb{E}[g(y_t, u_t) + V_{t+1}(y_{t+1})]$$

$$= \min_{u_t \leq y_t \leq s_t} \mathbb{E}[y_t^2(A^T e_1)u_t + b^T e_1 u_t^2 + w_t^2 u_t$$

$$(A y_t + b u_t + w_t)^T Q_{t+1}(A y_t + b u_t + w_t) + B_{t+1}^T (A y_t + b u_t + w_t) + C_{t+1}]$$

$$= \min_{u_t \leq y_t \leq s_t} \left[y_t^2 + b^T Q_{t+1} b|u_t|^2 + [y_t^T (A^T e_1 + A^T Q_{t+1} + Q_{t+1}^T) b] + B_{t+1}^T b|u_t$$

$$+ [(A y_t)^T Q_{t+1} (A y_t) + B_{t+1}^T A y_t + E(\omega_t^T Q_{t+1} \omega_t) + C_{t+1}].$$

If there were no constraints on $u_t$, then the optimal $u_t$ would be

$$u_t^* = -\frac{y_t^T L_t + \beta_t}{2\alpha_t},$$

where $L_t = (A^T e_1) + A^T (Q_{t+1} + Q_{t+1}^T) b, \beta_t = B_{t+1}^T b$, and $\alpha_t = b^T e_1 + b^T Q_{t+1} b$.

Considering the three possible outcomes based on where $u_t^*$ is situated in the state space, we will have three possible optimal $u_t$, each of which will yield different value functions

$$u_t = \begin{cases} 
0 & \text{if } -\frac{y_t^T L_t + \beta_t}{2\alpha_t} < 0, \\
S_t & \text{if } -\frac{y_t^T L_t + \beta_t}{2\alpha_t} > S_t, \\
-\frac{y_t^T L_t + \beta_t}{2\alpha_t} & \text{if } 0 \leq -\frac{y_t^T L_t + \beta_t}{2\alpha_t} \leq S_t.
\end{cases}$$

Let us define

$$U_1 = \{y | -\frac{y_t^T L_t + \beta_t}{2\alpha_t} < 0\},$$

$$U_2 = \{y | -\frac{y_t^T L_t + \beta_t}{2\alpha_t} > S_t\},$$

$$U_3 = \{y | 0 \leq -\frac{y_t^T L_t + \beta_t}{2\alpha_t} \leq S_t\},$$

to be mutually exclusive domains for $y$. If we replace the $u_{t,j}$ ($j = 1, 2, 3$) in $V_t(y_t)$ with the above results, we will have the following optimal cost-to-go at each stage $t$ ($t = 1, 2, ..., T$).

**Case j=1 :** $y \in U_1$

$$V_t(y_t) = y_t^T (A^T Q_{t+1} A) y_t + B_{t+1}^T A y_t + E(\omega_t^T Q_{t+1} \omega_t) + C_{t+1}.$$  

**Case j=2 :** $y \in U_2$

$$V_t(y_t) = y_t^T (A^T Q_{t+1} A + L_t e_2^T + e_2 \alpha_t e_2) y_t + [B_{t+1}^T A + \beta_t e_2^T] y_t + E(\omega_t^T Q_{t+1} \omega_t) + C_{t+1}.$$  

**Case j=3 :** $y \in U_3$

$$V_t(y_t) = y_t^T (A^T Q_{t+1} A - \frac{3L_t L_t^T}{4\alpha_t}) y_t + [B_{t+1}^T A - \frac{3L_t^2}{2\alpha_t}] y_t + E(\omega_t^T Q_{t+1} \omega_t) + C_{t+1} - \frac{3\beta_t^2}{4\alpha_t}.$$  

In all three cases, this results in the quadratic form

$$V_t(y_t) = y_t^T \Phi_t y_t + \Pi_t y_t + \Omega_t,$$

with $\Phi_{t,j}, \Pi_{t,j}$ and $\Omega_{t,j}$ being defined appropriately according to the three aforementioned cases.

We now approximate the value function based on the three distinct functions resulting from each possible $u_t$ by finding a quadratic function over the whole state space where the square distance from each of the above functions to it is minimum. For simplicity and clarity, we assume
a linear price impact with AR(1). These results can be extended to AR(m) without the loss in
generality. If we assume linear price formulation and no time lag in the price evolution, $A$ and $b$
are of the form
\[
A = \begin{bmatrix}
1 & 0 & \beta \\
0 & 1 & 0 \\
0 & 0 & \rho
\end{bmatrix}, \quad b = \begin{bmatrix}
a \\
-1 \\
0
\end{bmatrix}.
\]

**Lemma 2.** $\Phi_{t,j}$ and consequently $Q_t$ have the form
\[
\Phi_{t,j} = \begin{bmatrix}
0 & 1 & 0 \\
0 & d_1 & d_2 \\
0 & d_3 & d_4
\end{bmatrix}, \quad Q_t = \begin{bmatrix}
0 & 1 & 0 \\
0 & x_1 & x_2 \\
0 & x_3 & x_4
\end{bmatrix}.
\]

**Proof.** see appendix A.2. \qed

Under linear price impact, the approximate value function is of the form
\[
\tilde{V}_t(y_t) = y_t^T Q_t y_t + B_t^T y_t + C_t,
\]
where
\[
Q_t = \begin{bmatrix}
0 & 1 & 0 \\
0 & q_{t,1} & q_{t,2} \\
0 & q_{t,3} & q_{t,4}
\end{bmatrix}, \quad \text{and} \quad B_t = \begin{bmatrix}
0 \\
r_{t,1} \\
r_{t,2}
\end{bmatrix}.
\]

We need to find $(Q_t, B_t, C_t)$ that minimises the least squares equations, thus
\[
(Q_t, B_t, C_t) = \arg\min_{(Q_t, B_t, C_t)} \int (V_t(y_t) - \tilde{V}_t(y_t))^2 dy_t,
\]
where
\[
\int (V_t(y_t) - \tilde{V}_t(y_t))^2 dy_t = \\
\int \left[ y_t^T (Q_t - \Phi_{t,j}) y_t + (B_t^T - \Pi_{t,j}) y_t + C_t - \Omega_{t,j} \right]^2 dy_t \\
+ \int \left[ y_t^T (Q_t - \Phi_{t,j}) y_t + (B_t^T - \Pi_{t,j}) y_t + C_t - \Omega_{t,j} \right]^2 dy_t \\
+ \int \left[ y_t^T (Q_t - \Phi_{t,j}) y_t + (B_t^T - \Pi_{t,j}) y_t + C_t - \Omega_{t,j} \right]^2 dy_t.
\]

Figure 1 depicts a simplified graph that shows the approximated value function based on the
three value functions that are optimal in each region.

The quadratic coefficient $(Q_t - \Phi_{t,j})$ is
\[
Q_t - \Phi_{t,j} = \begin{bmatrix}
0 & 0 & 0 \\
0 & q_{t,1} - k_{t,j,1} & q_{t,2} - k_{t,j,2} \\
0 & q_{t,3} - k_{t,j,3} & q_{t,4} - k_{t,j,4}
\end{bmatrix},
\]
and the linear coefficient is
\[
B_t^T - \Pi_{t,j} = \begin{bmatrix}
0 \\
r_{t,1} - f_{t,j,1} \\
r_{t,2} - f_{t,j,2}
\end{bmatrix},
\]
where $k_{t,j,i}$ is the $i_{th}$ element of $\Phi_{t,j}$ corresponding to that of $Q$ and $f_{t,j,i}$ is the $i_{th}$ element of $\Pi_{t,j}$ corresponding to elements of $B_t$. The above results also render $p$, the first element of state
space, irrelevant for the rest of the calculations. Finally we have $C_t - n_{t,j}$ as the constant term.
Figure 1: The three regions (representing $U$ in cases $j = 1, 2, 3$) and approximation of the three value functions that are optimal in each region.

At each stage $t$, we need to solve the minimisation problem

$$(\{q\}, \{r\}, C) = \text{argmin}_{q, r, n} \int_y (V(y) - \tilde{V}(y))^2 dy,$$

where

$$\int_y (V(y) - \tilde{V}(y))^2 dy = \int_{U_1} ([s \ x] [q_1 - k_{1,1} \ q_2 - k_{1,2}] [s \ x] + \begin{bmatrix} r_1 - f_{1,1} & r_2 - f_{1,2} \end{bmatrix} [s \ x] + C - n_1)^2 dxds$$

$$+ \int_{U_2} ([s \ x] [q_1 - k_{2,1} \ q_2 - k_{2,2}] [s \ x] + \begin{bmatrix} r_1 - f_{2,1} & r_2 - f_{2,2} \end{bmatrix} [s \ x] + C - n_2)^2 dxds$$

$$+ \int_{U_3} ([s \ x] [q_1 - k_{3,1} \ q_2 - k_{3,2}] [s \ x] + \begin{bmatrix} r_1 - f_{3,1} & r_2 - f_{3,2} \end{bmatrix} [s \ x] + C - n_3)^2 dxds.$$

The three regions are obtained as follows:

- Case $j = 1 (U_1)$:

  $$-y^T L_t - \beta_t < 0 \Leftrightarrow y^T L_t > -\beta_t \Leftrightarrow \begin{bmatrix} l_2 \\ l_3 \end{bmatrix} > -\beta_t \Leftrightarrow$$

  $$l_2 s + l_3 x > -\beta_t \Leftrightarrow x > \frac{-\beta_t - l_2 s}{l_3}$$
• Case $j = 2$ ($U_2$):

$$-\left(\frac{y^T L_1 + \beta_i}{2\alpha_i}\right) > s_i \Leftrightarrow y^T L_1 < -\beta_i - 2\alpha_i s_i \Leftrightarrow \begin{bmatrix} s \ x \ \frac{l_2}{l_3}\end{bmatrix} < -\beta_i - 2\alpha_i s_i \Leftrightarrow$$

$$l_2 s + l_3 x < -\beta_i - 2\alpha_i s_i \Leftrightarrow x < -\frac{\beta_i - 2\alpha_i s_i - l_2 s}{l_3}$$

• Case $j = 3$ ($U_3$):

$$0 \leq -\left(\frac{y^T L_1 + \beta_i}{2\alpha_i}\right) \leq S_i \Leftrightarrow -2\alpha_i S_i - \beta_i \leq y^T L_1 \leq -\beta_i \Leftrightarrow$$

$$-2\alpha_i S_i - \beta_i \leq \begin{bmatrix} s \ x \ \frac{l_2}{l_3}\end{bmatrix} \leq -\beta_i - 2\alpha_i s_i - l_2 s + l_3 x < -\beta_i \Leftrightarrow$$

$$-\beta_i - 2\alpha_i s_i - l_2 s \leq x < -\frac{\beta_i - l_2 s}{l_3},$$

where $l_2$ and $l_3$ are elements of $L_i$: $\begin{bmatrix} 0 & l_2 & l_3 \end{bmatrix}^T$.

Considering that both $x$ and $s$ have an additional range that they must comply with $x \in [-P_0, P_0]$ and $s \in [0, s_i]$, we have the following ranges to apply in the formulation. Let elements of $x$ and $s$ be parameters $a_1, a_2, b_1, b_2$ such that $x \in [a_1, a_2]$ and $s \in [b_1, b_2]$.

$$U_1 = \{(x, s) | x \in [a_1 = -\frac{\beta_i - l_2 s}{l_3}, a_2 = P_0], s \in [b_1 = 0, b_2 = s_i]\},$$

$$U_2 = \{(x, s) | x \in [a_1 = P_0, a_2 = -\frac{\beta_i - 2\alpha_i s_i - l_2 s}{l_3}], s \in [b_1 = 0, b_2 = s_i]\},$$

$$U_3 = \{(x, s) | x \in [a_1 = -\frac{\beta_i - 2\alpha_i s_i - l_2 s}{l_3}, a_2 = -\frac{\beta_i - l_2 s}{l_3}], s \in [b_1 = 0, b_2 = s_i]\}$$

Applying these ranges on (6) and expanding, we will have the following which can be used to obtain the optimal values for $Q, R$ and $C$.

$$\int_0^y (V(y) - \bar{V}(y)) dy = \int_0^{P_0} \int_{-\frac{\beta_i - l_2 s}{l_3}}^{P_0} F \ dx ds$$

$$+ \int_0^{P_0} \int_{-\frac{\beta_i - 2\alpha_i s_i - l_2 s}{l_3}}^{P_0} F \ dx ds + \int_0^{P_0} \int_{-\frac{\beta_i - l_2 s}{l_3}}^{\frac{P_0}{\beta_i - 2\alpha_i s_i - l_2 s}} F \ dx ds$$

where

$$F = ((q_1 - k_{j,1}) s^2 + (q_4 - k_{j,4}) x^2 + ((q_2 - k_{j,2}) + (q_3 - k_{j,3})) sx$$

$$+ (r_1 - f_{j,1}) s + (r_2 - f_{j,2}) x + C - n_j)^2.$$

The above equations are quadratic functions on $\{q, r, n\}$ and can be rewritten as $R^T MR + O^T R + N$, where $R$ is the vector $[q_1, q_2, q_3, q_4, r_1, r_2, C]$. $M$ is a symmetric matrix derived from the coefficients of the above formulation, and $O$ is the linear coefficient.

The resulting equation is of quadratic form and yields an optimal solution. An algorithmic representation of our approximate dynamic programming method can be viewed in Algorithm 2.

4 Numerical Analysis

Our numerical analysis consists of two studies. In section 4.1 we simulate the example used in Bertsimas and Lo (1998) to illustrate the comparative advantages gained through our approximate dynamic programming method with non-negativity constraints. We compare our method (ADP)
Algorithm 2: Approximate Dynamic Programming

input:  \( \alpha_t, \beta_t, L_t, \{Q^0_t, B^0_t, C^0_t\} \) for all \( t \)

output: \( \{Q^*_t, B^*_t, C^*_t\} \)

for \( i = 1 \to M \) do
  for \( t = (T-1) \to 1 \) do
    for \( j = 1 \to 3 \) do
      \( \{\Phi^i_{t,j}, \Pi^i_{t,j}, \Omega^i_{t,j}\} = F1(Q^{i-1}_{t+1}, B^{i-1}_{t+1}, C^{i-1}_{t+1}, j) \)
    end for
    \( \{Q^i_t, B^i_t, C^i_t\} = F2(\Phi^i_{t,j}, \Pi^i_{t,j}, \Omega^i_{t,j}) \)
    if \( VF\{Q^i_t, B^i_t, C^i_t\} < VF\{Q^{i-1}_t, B^{i-1}_t, C^{i-1}_t\} \) then
      for \( t = 1 \to T \) do
        \( \{Q^*_t, B^*_t, C^*_t\} = \{Q^i_t, B^i_t, C^i_t\} \)
      end for
    end if
  end for
end for

\{\Phi, \Pi, \Omega\} = \text{Function } F1(\{Q, B, C, j\})

if \( j = 1 \) then
  \( \Phi = A^TQA \)
  \( \Pi = B^TA \)
  \( \Omega = E(\omega^T_t Q \omega_t) + C \)
else if \( j = 2 \) then
  \( \Phi = A^TQA + L_t e_1^T + \alpha_1 e_2^T \)
  \( \Pi = B^TA + \beta_1 e_2^T \)
  \( \Omega = E(\omega^T_t Q \omega_t) + C \)
else if \( j = 3 \) then
  \( \Phi = A^TQA - \frac{3L_t L_t^T}{4\alpha_2} \)
  \( \Pi = B^TA - \frac{3L_t L_t^T}{2\alpha_2} \)
  \( \Omega = E(\omega^T_t Q \omega_t) + C - \frac{3\beta^2}{4\alpha_2} \)
end if

\{Q, B, C\} = \text{Function } F2(\Phi_t, \Pi_t, \Omega_t)

\[
\arg\min_{\{(Q_t, B_t, C_t)\}} \int_{U_1} [y_t^T(Q_t - \Phi_{t,1})y_t + (B^T_t - \Pi_{t,1})y_t + C_t - \Omega_{t,1}]^2 dy_t \\
+ \int_{U_2} [y_t^T(Q_t - \Phi_{t,2})y_t + (B^T_t - \Pi_{t,2})y_t + C_t - \Omega_{t,2}]^2 dy_t \\
+ \int_{U_3} [y_t^T(Q_t - \Phi_{t,3})y_t + (B^T_t - \Pi_{t,3})y_t + C_t - \Omega_{t,3}]^2 dy_t
\]
Table 1: Execution time (milliseconds) of B&L and ADP algorithms for variable number of periods (T) and market volatility rates ($\sigma^2$ values)

<table>
<thead>
<tr>
<th>Period</th>
<th>$\sigma^2=0.01$</th>
<th>$\sigma^2=0.1$</th>
<th>$\sigma^2=1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>B&amp;L</td>
<td>ADP</td>
<td>B&amp;L</td>
<td>ADP</td>
</tr>
<tr>
<td>2</td>
<td>0.26</td>
<td>12.78</td>
<td>0.31</td>
</tr>
<tr>
<td>4</td>
<td>0.45</td>
<td>14.57</td>
<td>0.51</td>
</tr>
<tr>
<td>6</td>
<td>0.67</td>
<td>19.43</td>
<td>0.78</td>
</tr>
<tr>
<td>8</td>
<td>1.00</td>
<td>28.85</td>
<td>1.11</td>
</tr>
<tr>
<td>10</td>
<td>1.37</td>
<td>45.95</td>
<td>1.53</td>
</tr>
<tr>
<td>12</td>
<td>1.92</td>
<td>70.15</td>
<td>2.13</td>
</tr>
<tr>
<td>14</td>
<td>2.60</td>
<td>103.12</td>
<td>2.82</td>
</tr>
<tr>
<td>16</td>
<td>3.32</td>
<td>147.59</td>
<td>3.56</td>
</tr>
<tr>
<td>18</td>
<td>4.19</td>
<td>204.63</td>
<td>4.49</td>
</tr>
<tr>
<td>20</td>
<td>5.05</td>
<td>276.41</td>
<td>5.35</td>
</tr>
</tbody>
</table>

With their closed-form solution (B&L) and a naive strategy where the trade is divided into equal sizes to be executed over the trading horizon. In this case we test the efficiency of each optimal strategy with regard to different realisations of information variable $x$. Since the variable $x$ is the main driving factor in the volatility of prices in the scenario presented by Bertsimas and Lo (1998), we test the performance of the algorithms against different rates of variance in $x$.

In section 4.2 we simulate trading of a security from the London Stock Exchange on a specific date based on intra-day trade information of that day. This is to illustrate the effectiveness of our method in a practical setting and its advantage over the classic and naive methods that do not include the non-negativity constraints.

We code the algorithms in MATLAB and run all the experiments on a 64-bit Windows 7 workstation having 4GB of RAM and quad-core Intel CPU at 2.6GHz. All the experiments were very fast and as such we forego a detailed examination of the time performance. As a brief guideline, table 1 shows the time taken for a single run of each of the algorithms for various trading periods and market volatility rates.

### 4.1 A Simulated Example

The example involves execution (buy) of 100000 shares over $T=20$ periods. The current price is $50$. The parameters are set as follows: $a = 5 \times 10^{-5}$, $\beta = 5$, $\rho = 0.5$ and $\sigma^2 = (0.125)^2$.

For a full description of the values and the reasoning behind the choice of the values, interested readers are referred to Bertsimas and Lo (1998). In summary, $a$ is chosen to yield a price impact of $500000$ if the trader executes the 100000 shares in one transaction. The standard deviation of $\epsilon_t$ is calibrated to be one tick (12.5 cents) per period.

We assign different values to $\sigma^2$ (variance of the information evolution, a white noise process, representing the overall market volatility) in order to test the efficiency of the algorithms with regard to the market behaviour where it might drive the prices significantly up or down and render the non-negativity constraints relevant. For example, if the optimal strategy is to sell during a buy operation, where we do not apply the non-negativity constraints, the original method is likely to offer negative trades as part of its optimal solution. However, if the assumption is that short-selling is not possible, then the original algorithm would not be able to accommodate the above cases. The naive strategy on the other hand is unable to utilise the swings in the price to minimise the incurred overall cost. The approximate dynamic programming method, as expected, provides superior results compared to the original method of Bertsimas and Lo (1998) (as well as the naive strategy) under a volatile market condition, when the assumption is that a buy operation cannot include sales.

Table 2 depicts the values of the expected execution cost for a single random run. Table 2
Table 2: Expected execution cost in example execution of 100000 shares

<table>
<thead>
<tr>
<th>$\sigma^2_n$</th>
<th>ADP</th>
<th>B&amp;L</th>
<th>% Diff.</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.001</td>
<td>5211140</td>
<td>5218652</td>
<td>0.14</td>
</tr>
<tr>
<td>0.002</td>
<td>5183163</td>
<td>5196262</td>
<td>0.25</td>
</tr>
<tr>
<td>0.005</td>
<td>5127863</td>
<td>5151103</td>
<td>0.45</td>
</tr>
<tr>
<td>0.01</td>
<td>5072956</td>
<td>5105838</td>
<td>0.65</td>
</tr>
<tr>
<td>0.02</td>
<td>5003845</td>
<td>5048489</td>
<td>0.89</td>
</tr>
<tr>
<td>0.05</td>
<td>4885584</td>
<td>4947445</td>
<td>1.27</td>
</tr>
<tr>
<td>0.1</td>
<td>4785283</td>
<td>4863176</td>
<td>1.63</td>
</tr>
<tr>
<td>0.2</td>
<td>4727752</td>
<td>4820594</td>
<td>1.96</td>
</tr>
<tr>
<td>0.5</td>
<td>4951978</td>
<td>5047339</td>
<td>1.93</td>
</tr>
<tr>
<td>1</td>
<td>5030476</td>
<td>5141406</td>
<td>2.21</td>
</tr>
<tr>
<td>2</td>
<td>4848128</td>
<td>5004119</td>
<td>3.22</td>
</tr>
<tr>
<td>5</td>
<td>4494581</td>
<td>4739153</td>
<td>5.44</td>
</tr>
<tr>
<td>10</td>
<td>4090485</td>
<td>4435450</td>
<td>8.43</td>
</tr>
<tr>
<td>20</td>
<td>3511292</td>
<td>3999008</td>
<td>13.89</td>
</tr>
<tr>
<td>50</td>
<td>2362097</td>
<td>3133048</td>
<td>32.64</td>
</tr>
<tr>
<td>100</td>
<td>1066983</td>
<td>2157133</td>
<td>102.17</td>
</tr>
</tbody>
</table>

shows that the expected value function improves substantially for the ADP method compared to B&L when the variance of the market volatility increases. The naive strategy is equivalent in almost all instances of $\sigma^2_n$ to that of B&L and hence not shown here. As we are dealing with the volatility of the market, the more important aspect of the improvement of the ADP method over the other methods is apparent in the actual execution cost over the whole trade period as can be seen in Figure 2, which depicts the realised execution cost for various $\sigma^2_n$ rates. To achieve this, we run each algorithm with random realisations of $x$ and calculate the execution cost at each stage. This graph shows the average of total execution cost for 50 simulation runs for each method.

![Figure 2: Actual execution cost for the three methods with variable $\sigma^2_n$ rates](image)
The performances of these three methods are very similar when the $\sigma^2$ rates are very small and when the probability of negative trade being optimal is relatively low. Since the B&L method provides optimality with regard to the uncertainty in the market and includes short-selling in its policies, when the short-selling is prohibited, it loses that advantage. This is reflected in the average execution cost values in $\sigma^2$ rates of 1 and above. The ADP, on the other hand, takes into account the non-negativity constraints and provides optimal policies in which the majority of the trade is performed when the price has swung down and very little trade is engaged in when the prices are high. In other words, ADP is better equipped to take advantage of price variations without the need for short-selling. Note the sharp drop in ADP execution costs when the volatility increases beyond 1, where it is able to execute with higher probability a larger amount of the security in lower prices leading to lower overall execution cost.

4.2 An Empirical Example

The example shown in section 4.1 proves the suitability of the ADP approach in situations where short-selling is not allowed. However, to gauge the improvement that this method will yield in a real-world situation, we test the algorithm on real securities traded in the market and compare it to that of B&L as a benchmark on what would be the optimal policy when we ignore the non-negativity constraints.

We consider three securities from the London Stock Exchange that represent various levels of liquidity and record the tick price data for each of the selected stock for the duration of two weeks. The FTSE100 index is considered as the information vector $x$ during the same period. We also record the publicly available trade and price information of each share for a particular day during the two weeks. Out of the FTSE100 companies, we choose Lloyds Banking Group shares as shares with high liquidity levels, Rolls-Royce as a medium-liquidity share and Next Plc as representative of low liquidity securities. The choice of the three securities is not based on strict criteria but a loose interpretation of liquidity in order to evaluate the effects when applying the methods in this paper on stocks of varying liquidity. We also acquire intra-day price data for these securities where the information is available for the opening and closing prices and the volume of trade during each minute. We consider the price between the opening and closing prices to be the price of the share in that time period.

The first step is to estimate the parameters of $x_{t+1} = \rho x_t + \eta_t$ in order to be able to forecast the value of $x$ from the previous period. We perform a linear regression fit in R statistical software based on the historical time-series data above. The parameters of information evolution $(x_{t+1} = \rho x_t + \eta_t)$ which represent the FTSE100 index are calculated as

$$x_{t+1} = 70.91 + 0.98x_t + \eta_t,$$

where $\mu_\eta = 0$ and $\sigma_\eta = 14.15$.

Table 3 details the coefficients of the price evolution model $p_{t+1} = p_t + \beta x_t + \alpha u_t + \epsilon_t$ in relation to each chosen security.

Based on the parameters found in Table 3 pertaining to the behaviour of price based on the independent variables, as well as the parameters of the information evolution, we run our algorithm against that of Bertsimas and Lo (1998) to gauge the performance of these algorithms in a real-world application. We compare the results based on different numbers of trade periods as well as different rates of market volatility.

Figure 3 depicts the expected value function for the two methods based on 10 trade periods and varying degrees of market volatility. As can be seen, the B&L method only outperforms our method in the case of very small market volatility. As the volatility increases, the expected cost of trading the predefined number of shares increases for the B&L method. The ADP method increases in performance as the volatility increases while the performance of B&L is declining. This is indicative of the ability of the ADP method in taking advantage of price swings when...
Table 3: Regression analysis on the three chosen stocks

<table>
<thead>
<tr>
<th></th>
<th>Lloyds</th>
<th>Rolls-Royce</th>
<th>Next Plc</th>
</tr>
</thead>
<tbody>
<tr>
<td>Price (1 lag)</td>
<td>9.805e−01***</td>
<td>8.776e−01***</td>
<td>9.159e−01***</td>
</tr>
<tr>
<td>Std. Error</td>
<td>(1.615e−02)</td>
<td>(1.886e−02)</td>
<td>(2.370e−02)</td>
</tr>
<tr>
<td>Trade Volume</td>
<td>−1.322e−10</td>
<td>1.227e−06</td>
<td>6.378e−05</td>
</tr>
<tr>
<td>Std. Error</td>
<td>(2.861e−09)</td>
<td>(1.359e−06)</td>
<td>(3.140e−05)</td>
</tr>
<tr>
<td>UK100 Index</td>
<td>7.790e−05</td>
<td>2.802e−02**</td>
<td>4.523e−02***</td>
</tr>
<tr>
<td>Std. Error</td>
<td>(1.573e−04)</td>
<td>(4.223e−03)</td>
<td>(1.335e−02)</td>
</tr>
<tr>
<td>Residual Std. Err.</td>
<td>0.227</td>
<td>3.735</td>
<td>11.06</td>
</tr>
<tr>
<td>R²</td>
<td>0.9474</td>
<td>0.9713</td>
<td>0.9612</td>
</tr>
<tr>
<td>Adj. R²</td>
<td>0.9467</td>
<td>0.971</td>
<td>0.9607</td>
</tr>
</tbody>
</table>

*** p < 0.001, ** p < 0.01, * p < 0.05, + p < 0.1

Figure 3: Expected execution cost of Lloyds shares with variable rates of $\sigma^2_{\eta}$
Table 4: Actual execution cost realisation from each algorithm in execution of 100000 Lloyds shares and the percentage difference between the two algorithms based on the number of trading periods.

<table>
<thead>
<tr>
<th># of periods</th>
<th>ADP</th>
<th>B&amp;L</th>
<th>% Diff.</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>8868042</td>
<td>8867250</td>
<td>-0.008</td>
</tr>
<tr>
<td>20</td>
<td>10595009</td>
<td>10586747</td>
<td>-0.077</td>
</tr>
<tr>
<td>30</td>
<td>11552346</td>
<td>11563438</td>
<td>0.096</td>
</tr>
<tr>
<td>40</td>
<td>11522654</td>
<td>11543933</td>
<td>0.151</td>
</tr>
<tr>
<td>50</td>
<td>11484477</td>
<td>11509757</td>
<td>0.220</td>
</tr>
<tr>
<td>60</td>
<td>11449253</td>
<td>11475887</td>
<td>0.232</td>
</tr>
<tr>
<td>70</td>
<td>11408889</td>
<td>11427038</td>
<td>0.159</td>
</tr>
<tr>
<td>80</td>
<td>11362621</td>
<td>11364245</td>
<td>0.014</td>
</tr>
<tr>
<td>90</td>
<td>11290233</td>
<td>11304324</td>
<td>0.124</td>
</tr>
</tbody>
</table>

the B&L falls short because of neglecting the sign constraints. If short-selling is prohibited, our method adapts the policies online by incorporating the non-negativity into its framework from the start.

Table 4 outlines the actual execution costs of the two algorithms for the Lloyds shares. The ADP algorithm outperforms the original B&L method in most cases. When the trading periods are relatively few, there will naturally be less probability of favourable prices occurring and thus the ADP algorithm is slightly under-performing.

Figure 4 on the other hand depicts the results of the actual execution cost in the case of Lloyds shares with both methods over varying market volatility. The benefits of the approximate dynamic programming method over the B&L method is apparent from this figure when the volatility increases.

Both other stocks show similar behaviour with varying trade periods and market volatility. Due to space limitation, the interested reader can find graphs, similar to those above, for the
Our reasons for extending the work of Bertsimas and Lo (1998) is to bring the problem closer to the real-world applications. This includes adding non-negativity constraints to the definition of the problem; however, our extensions add significantly to the complexity of the problem. It is a well-known shortcoming of the dynamic programming method that it cannot accommodate constraints very easily. We provide an approximate dynamic programming method that circumvents this. Our simulated and empirical results support the inclusion of non-negativity constraints, when the volatility of the market, and consequently of prices, are such that the probability of negative trades is significant. It improves the expected (and actual) execution cost in real circumstances and in comparison to the model by Bertsimas and Lo (1998). The difference between the performance of the models is not substantial when applied to scenarios incorporating low volatility; however, as we raise the market and price fluctuations in the model, the significance of our method over the other two methods becomes apparent. The benefits of the new method are also apparent in the empirical study, where we test the methods on a set of data that warrants large swings in price. In both cases we observe improvement in the results which encourages the use of the new method over the previous methods under the conditions discussed.

References


A Appendix

A.1 Proof of Lemma 1

Lemma $\alpha_t = (b^T e_1 + b^T K_{t+1} b) > 0$ is true for all $t = 1, ..., T - 1$.

Proof. Since $K_T = (A^T e_1) e_T^2 + b^T e_1 (e_T e_1^2)$, we have

\[
K_T = \begin{bmatrix} 1 \\ \beta \\ 0 \end{bmatrix} \begin{bmatrix} 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} + a \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}
\]

\[
= \begin{bmatrix} 0 & 1 & 0 \\ 0 & a & 0 \\ 0 & \beta & 0 \end{bmatrix}
\]

and

\[
(b^T e_1 + b^T K_T b) = a + \begin{bmatrix} a \\ -1 \\ 0 \end{bmatrix} \begin{bmatrix} 0 & 1 & 0 \\ 0 & a & 0 \\ 0 & \beta & 0 \end{bmatrix} \begin{bmatrix} a \\ 0 \\ 0 \end{bmatrix}
\]

\[
= a + \begin{bmatrix} a \\ -1 \\ 0 \end{bmatrix} \begin{bmatrix} -1 \\ -a \\ -\beta \end{bmatrix} = a.
\]

Since $a$ is assumed to be positive, $(b^T e_1 + b^T K_T b) > 0$ is true.

For $K_{T-1}$ we have

\[
K_{T-1} = A^T K_T A - \frac{L_{T-1} L_{T-1}^T}{4\alpha_{T-1}}
\]

\[
= \begin{pmatrix} 0 & 1 & 0 \\ 0 & a - \frac{a^2}{4\alpha_{T-1}} & 0 \\ 0 & \beta + \beta\rho - \frac{\beta\rho a}{4\alpha_{T-1}} & -\frac{\beta^2\rho^2}{4\alpha_{T-1}} \end{pmatrix}
\]
resulting in $\alpha_{T-2} = (b^T e_1 + b^T K_{T-3} b) = a - \frac{\alpha_{T-1}}{4}$.

Continuing in this fashion we get

$$\alpha_t = \alpha_{t+1} - \left(\frac{2^{t+1}+1}{4\alpha_{t+1}}\right)^2.$$  

Since $\alpha_{T-1} = a$ which is assumed to be positive and since at each backward stage a value smaller than itself is deduced from it, it is concluded that $\alpha_t$ is a non-negative value for all $t = 1, ..., T - 1$.

\[\square\]

### A.2 Proof of Lemma 2

**Lemma** $\Phi_{t,j}$ and consequently $Q_t$ have the following format:

$$\Phi_{t,j} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & d_1 & d_2 \\ 0 & d_3 & d_4 \end{bmatrix}, \quad Q_t = \begin{bmatrix} 0 & 1 & 0 \\ 0 & x_1 & x_2 \\ 0 & x_3 & x_4 \end{bmatrix}.$$  

**Proof.** We prove this by way of induction.

First of all, $K_T$ has the format $\begin{bmatrix} 0 & 1 & 0 \\ 0 & a & 0 \\ 0 & \beta & 0 \end{bmatrix}$ from the formula $K_T = (A^T e_1 e_2^T + b^T e_1 (e_2 e_2^T))$ which corresponds to the general format.

We assume $K_{t+1}$ to be of the format $\begin{bmatrix} 0 & 1 & 0 \\ 0 & d_1 & d_2 \\ 0 & d_3 & d_4 \end{bmatrix}$.

Since we have $K_t = A^T K_{t+1} A - \frac{L_t L_t^T}{4\alpha_t}$, in which $L_t$ and $\alpha_t$ are

$$L_t = A^T e_1 + A^T K_{t+1} b + b^T K_{t+1} A$$

$$= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ \beta & 0 & \rho \end{bmatrix} + \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ \beta & 0 & \rho \end{bmatrix} \begin{bmatrix} a \\ 0 \\ 0 \end{bmatrix}$$

$$+ \begin{bmatrix} a & -1 & 0 \\ 0 & d_1 & d_2 \\ 0 & d_3 & d_4 \end{bmatrix} \begin{bmatrix} 1 & 0 & \beta \\ 0 & 1 & 0 \\ 0 & 0 & \rho \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ \beta & 0 & \rho \end{bmatrix} + \begin{bmatrix} -1 \\ -d_1 \\ -d_3 \end{bmatrix} + \begin{bmatrix} a & -1 & 0 \\ 0 & d_1 & d_2 \rho \\ 0 & d_3 & d_4 \rho \end{bmatrix}$$

$$= \begin{bmatrix} 1 & -1 \\ 0 & -d_1 \\ \beta & -\rho d_3 \end{bmatrix} + \begin{bmatrix} a - d_1 \\ 0 \end{bmatrix}$$

$$= \begin{bmatrix} 0 \\ a - 2d_1 \\ -\rho (d_2 + d_3) \end{bmatrix},$$
\[
\alpha_t = b^T e_1 + b^T K_{t+1} b \\
\quad = a + \begin{bmatrix} a & -1 & 0 \end{bmatrix} \begin{bmatrix} 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & d_1 & d_2 \\ 0 & d_3 & d_4 \end{bmatrix} \begin{bmatrix} a \\ -1 \\ -d_1 \\ -d_3 \end{bmatrix} = d_1.
\]

Thus we have
\[
K_t = A^T K_{t+1} A - \frac{L_t L_t^T}{4\alpha_t}
\]
\[
= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ \beta & 0 & \rho \end{bmatrix} \begin{bmatrix} 0 & 1 & 0 \\ 0 & d_1 & d_2 \\ 0 & d_3 & d_4 \end{bmatrix} \begin{bmatrix} 1 & 0 & \beta \\ 0 & 1 & 0 \\ 0 & 0 & \rho \end{bmatrix}
\]
\[
\quad - \begin{bmatrix} 0 \\ a - 2d_1 \\ -\rho(d_2 + d_3) \end{bmatrix} \begin{bmatrix} 0 & a - 2d_1 & -\rho(d_2 + d_3) \end{bmatrix}/4d_1
\]
\[
\quad = \begin{bmatrix} 0 & 1 & 0 \\ 0 & d_1 & \rho d_2 \\ 0 & \beta + \rho d_3 & \rho^2 d_4 \end{bmatrix} - \begin{bmatrix} 0 & 0 & 0 \\ 0 & \varpi & \varphi \\ 0 & \varphi & \zeta \end{bmatrix}
\]
\[
\quad = \begin{bmatrix} 0 & 1 & 0 \\ 0 & d_1 - \varpi & \rho d_2 - \varphi \\ 0 & \beta + \rho d_3 - \varphi & \rho^2 d_4 - \zeta \end{bmatrix},
\]

where \( \varpi = \frac{(a - 2d_1)^2}{4d_1} \), \( \varphi = \frac{(a - 2d_1)(-\rho(d_2 + d_3))}{4d_1} \) and \( \zeta = \frac{\rho^2(d_2 + d_3)^2}{4d_1} \).

The end result clearly has the same form as indicated.

\[\square\]

**A.3 Additional performance indicators for section 4.2**

The following tables and figures demonstrate the results shown in relation to the ADP and B&L algorithms for Rolls-Royce and Next plc. share prices.

Table 5: Actual execution cost realisations from running each algorithm on 100000 Rolls-Royce shares

<table>
<thead>
<tr>
<th># of periods</th>
<th>ADP</th>
<th>B&amp;L</th>
<th>% Diff.</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>113196572</td>
<td>113105567</td>
<td>-0.080</td>
</tr>
<tr>
<td>20</td>
<td>113150598</td>
<td>113072115</td>
<td>-0.069</td>
</tr>
<tr>
<td>30</td>
<td>113128975</td>
<td>113178923</td>
<td>0.044</td>
</tr>
<tr>
<td>40</td>
<td>113092688</td>
<td>113065843</td>
<td>-0.023</td>
</tr>
<tr>
<td>50</td>
<td>113045862</td>
<td>113028378</td>
<td>-0.015</td>
</tr>
<tr>
<td>60</td>
<td>113002658</td>
<td>113158182</td>
<td>0.137</td>
</tr>
<tr>
<td>70</td>
<td>112949417</td>
<td>113053358</td>
<td>0.092</td>
</tr>
<tr>
<td>80</td>
<td>112852015</td>
<td>113009933</td>
<td>0.139</td>
</tr>
<tr>
<td>90</td>
<td>112723957</td>
<td>113060720</td>
<td>0.298</td>
</tr>
<tr>
<td>100</td>
<td>112684469</td>
<td>112971536</td>
<td>0.254</td>
</tr>
</tbody>
</table>
Table 6: Actual execution cost realisations from running each algorithm on 100000 Next plc. shares

<table>
<thead>
<tr>
<th># of periods</th>
<th>ADP</th>
<th>B&amp;L</th>
<th>% Diff.</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>721966680</td>
<td>721800780</td>
<td>-0.022</td>
</tr>
<tr>
<td>20</td>
<td>721755698</td>
<td>721629741</td>
<td>-0.017</td>
</tr>
<tr>
<td>30</td>
<td>721653652</td>
<td>721952501</td>
<td>0.041</td>
</tr>
<tr>
<td>40</td>
<td>721365874</td>
<td>721687618</td>
<td>0.044</td>
</tr>
<tr>
<td>50</td>
<td>721052563</td>
<td>721556396</td>
<td>0.069</td>
</tr>
<tr>
<td>60</td>
<td>720865896</td>
<td>721908650</td>
<td>0.144</td>
</tr>
<tr>
<td>70</td>
<td>720765149</td>
<td>721684057</td>
<td>0.127</td>
</tr>
<tr>
<td>80</td>
<td>719697785</td>
<td>721385540</td>
<td>0.234</td>
</tr>
<tr>
<td>90</td>
<td>719122901</td>
<td>721530130</td>
<td>0.334</td>
</tr>
<tr>
<td>100</td>
<td>718250352</td>
<td>721282194</td>
<td>0.422</td>
</tr>
</tbody>
</table>

Figure 5: Actual execution cost for Rolls-Royce share price with variable market volatility
Figure 6: Actual execution cost for Next plc share price with variable market volatility