The London School of Economics and Political Science

Stochastic Dynamic Programming Methods for the Portfolio Selection Problem

Dimitrios Karamanis

A thesis submitted to the Department of Management of the London School of Economics for the degree of Doctor of Philosophy in Management Science

London, 2013

Declaration

I certify that the thesis I have presented for examination for the MPhil/PhD degree of the London School of Economics and Political Science is solely my own work other than where I have clearly indicated that it is the work of others (in which case the extent of any work carried out jointly by me and any other person is clearly identified in it).

I declare that my thesis consists of 65000 words.

Abstract

In this thesis, we study the portfolio selection problem with multiple risky assets, linear transaction costs and a risk measure in a multi-period setting. In particular, we formulate the multi-period portfolio selection problem as a dynamic program and to solve it we construct approximate dynamic programming (ADP) algorithms, where we include Conditional-Value-at-Risk (CVaR) as a measure of risk, for different separable functional approximations of the value functions. We begin with the simple linear approximation which does not capture the nature of the portfolio selection problem since it ignores risk and leads to portfolios of only one asset. To improve it, we impose upper bound constraints on the holdings of the assets and we notice that we have more diversified portfolios. Then, we implement a piecewise linear approximation, for which we construct an update rule for the slopes of the approximate value functions that preserves concavity as well as the number of slopes. Unlike the simple linear approximation, in the piecewise linear approximation we notice that risk affects the composition of the selected portfolios. Further, unlike the linear approximation with upper bounds, here wealth flows naturally from one asset to another leading to diversified portfolios without us needing to impose any additional constraints on how much we can hold in each asset. For comparison, we consider existing portfolio selection methods, both myopic ones such as the equally-weighted and a single-period portfolio models, and multi-period ones such as multi-stage stochastic programming. We perform extensive simulations using real-world equity data to evaluate the performance of all methods and compare all methods to a market Index. Computational results show that the piecewise linear ADP algorithm significantly outperforms the other methods as well as the market and runs in reasonable computational times. Comparative results of all methods are provided and some interesting conclusions are drawn especially when it comes to comparing the piecewise linear ADP algorithms with multistage stochastic programming.

Acknowledgements

First and foremost, I would like to thank my supervisor, Dr Katerina Papadaki, for giving me the opportunity to undertake this PhD. Her enthusiasm about research and her expertise in the methods that were required by my topic have been invaluable in progressing with my doctoral work. Not to mention that, except for my thesis advisor, she has been a valuable friend of mine.

Furthermore, I am very grateful to my examiners, Professor Victor DeMiguel and Professor M. Grazia Speranza, for their constructive comments and feedback which gave me the opportunity to look at different aspects of the problem. With their help, the contributions of my thesis reflect much better and I am better able to communicate them.

Further, I owe a big thank you to my family, my father Nikiforos Karamanis, my mother Stavroula Karamani and my siblings Alexandra Karamani and Angelos Gkartzonikas to whom I am deeply indebted for their love and support all these years.

I would also like to thank my friends back in Greece, Christos, Evelyn, Sevi, Froso, Charis and Giorgos for standing by my side in good and bad times. I will never forget how impatient I have been towards the end of each term to go back to Greece, meet my friends and spend time with them.

Finally, I would like to thank the friends I made in London, Selvin, Ozlem, Camelia, Sumitra, Shweta, Nayat, Anastasia, Clodaugh, Nikos, and Sinan, for spending a good time with them in London and not only in London. I will always remember our 3-4 day trips to Paris, Ljubljana, Berlin, Lisbon, Amsterdam, Finland, Norway and many more other destinations. Together we had lots of fun!

Contents

List of Tables

List of Figures

List of Algorithms

Nomenclature

$\mathcal{N}$	Set of risky assets
$\mathcal{T}$	Set of discrete points in time horizon
T	End of time horizon
x_{it}	How much we buy from risky asset i at time t
$\boldsymbol{x}_t$	Vector of buying variables at time t
y_{it}	How much we sell from risky asset i at time t
$\boldsymbol{y}_t$	Vector of selling variables at time t
h_{it}	Pre-decision holdings of asset i at time t
$\boldsymbol{h}_t$	Vector of pre-decision holdings at time t
h_{it}^+	Post-decision holdings of asset i at time t
$\boldsymbol{h}_t^+$	Vector of post-decision holdings at time t
R_{it}	Rate of return of risky asset i at time t
$\boldsymbol{R}_t$	Vector of rates of returns of risky assets at time t
V_{it}	Value function of asset i at time t
u_{it}	Slope of asset i in value function V_{it} at time t
v^T	Total wealth at time T
γ	Risk importance parameter
θ	Proportional transaction costs per (monetary) unit of asset traded
s	Scenario
n	Node
p_n	Probability of node n
$\mathcal{K}$	Set of nodes on a scenario tree
$\mathcal{K}_t$	Set of nodes at stage t of a scenario tree
$\mathcal{C}(n)$	Set of children nodes of node n on a scenario tree
k_n	Predecessor node of node n on a scenario tree
VaR	Value-at-Risk
CVaR	Conditional-Value-at-Risk
SP	Stochastic Programming

MSP	Multistage Stochastic Programming
DP	Dynamic Programming
ADP	Approximate Dynamic Programming
O-GARCH	Orthogonal Generalized Autoregressive Conditional Heteroskedasticity
GAMS	General Algebraic Modeling System

Chapter 1

Introduction

The fundamental contribution of this thesis is the development of *approximate dynamic programming (ADP)* algorithms that solve the *portfolio selection problem* over a long-term *planning horizon*. Conditional-Value-at-Risk (CVaR) is used as a *measure of risk* and *transaction costs* are taken proportional to the trading amounts.

Specifically, in this study we formulate the portfolio selection problem as a *dynamic program*, which due to the high-dimensional *state*, *outcome* and *action spaces* becomes quickly computationally intractable. To solve it, we use *approximate dynamic programming* methods, which provide a time decomposition and approximation framework that breaks long-term horizon problems into a group of smaller problems that can be solved using mathematical programming methods.

The *contributions* of this thesis can be summarized as follows:

1. To our knowledge, so far ADP methods have been applied mainly to one-dimensional financial problems without risk. Here, we expand this by applying ADP methods to a multi-dimensional portfolio selection problem with a widely used measure of risk and transaction costs.

2. We introduce a novel piecewise linear ADP scheme that can handle high-dimensional problems. By controlling the number of slopes in the piecewise linear value function approximations and allowing the value of the slopes and the slope intervals to be adaptively estimated, we end up with a scheme that runs in reasonable times and performs extremely well against the competing methods.

3. We provide a novel comparative analysis between a large range of portfolio selection methods, where we compare various ADP algorithms of increasing

complexity against:

(a) *Myopic portfolio selection methods:* A single-period and the equally-weighted portfolio methods, where decisions on portfolio composition at any point in time ignore their impact in the future.

(b) *Multi-period portfolio selection methods:* Multistage stochastic programming (MSP), where the investor accounts for both the short-term and the long-term effects of the investment strategies. In a discrete time setting, this is achieved by considering an investment *planning horizon*, where the investor has to take temporal decisions in order to achieve a goal at some date in the future.

A comparison against a *market Index* is included.

4. In the ADP methods, we need to solve a large number of linear programs the complexity of which varies depending on the assumed value function approximations and increases significantly for the piecewise linear approximations. Solving these linear programs with mathematical programming methods takes a long time. Here, we construct greedy algorithms for all approximation schemes that solve our linear programs much faster and we prove that the solutions we get from these algorithms are optimal.

To evaluate the proposed approximate dynamic programming algorithms, we use real-life equity data from the London Stock Exchange and we divide into the following two parts: the *in-sample data* which we use to generate scenario paths and the *out-of-sample data* which we use to test the methods. For robustness, we have considered all combinations of the market going up or down in the in-sample and out-of-sample data.

Figure 1.1 shows the *system structure*, which is a snapshot of the sequence of events in evaluating and assessing the performance of the approximate dynamic programming methods. The performance evaluation process evolves as follows: Historical returns from a financial time series serve as an input for the *scenario path generator*, which outputs scenario paths. The generated scenario paths have a twofold use. On the one hand, they serve as an input for the approximate dynamic programming and the single-period portfolio methods. On the other hand, they serve as an input for the *scenario tree constructor*, which outputs a scenario

tree using scenario reduction and scenario tree construction methods. The generated scenario tree serves in turn as an input for the multistage stochastic programming method. For each method we compute the out-of-sample terminal wealth (see output part in figure 1.1) which is our *performance measure* and all methods are compared in terms of performance and complexity.

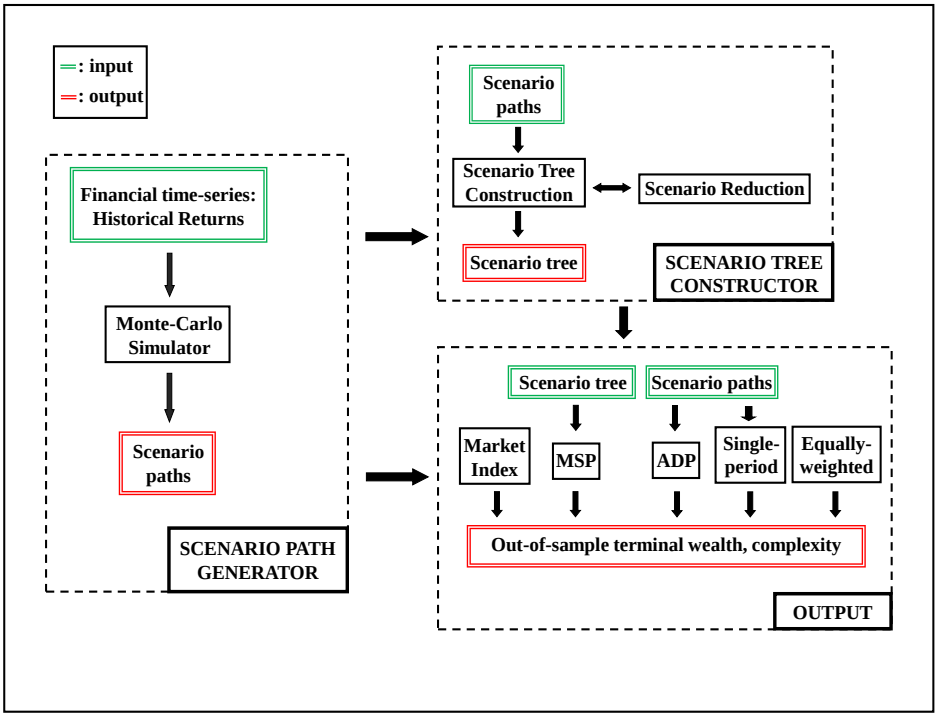


Figure 1.1: System Structure

1.1 Literature Review

In this section, we briefly describe the current state of the art in the era of portfolio optimization. More literature regarding the methods used in this study is provided in the respective chapters.

In the portfolio optimization literature, most portfolio optimization models have been one-period models. The modern portfolio theory dates back to the 1950's (see [54] and [55]), when Markowitz first formulated the portfolio selection problem in terms of the mean (expected) return and the variance of return of a portfolio of assets. His analysis led to introducing the concept of the efficient frontier, which is a hyperbola representing the boundary of the set of portfolios with the maxi-

mum expected return for any given level of risk (variance). Following the work of Markowitz, Tobin showed in 1958 (see [78]) that including a risk-free asset in the portfolio of assets the efficient frontier changes from a hyperbola to a straight line, which led to the so called CAPM model (see [75]).

Despite its theoretical reputation, researchers very early identified that with the available mathematical tools back then the computational burden in solving the portfolio model of Markowitz increased substantially with the number of assets since constructing large-scale portfolios requires solving large-scale quadratic optimization problems (due to variance) with dense covariance matrices. Researchers attempted to alleviate this difficulty from the early years of the history of the modern portfolio theory by introducing factors that drive the stock prices (see for example the index models in [63] and [73]) or by introducing approximation schemes, where linear programming plays a central role (see for example [74] and [76]). The importance of linear programming in financial applications has grown further by the need to include binary variables that help us model other realistic features, such as transaction costs (see for example [39]). Interest in how transaction costs can affect investors' decisions on portfolio composition goes back to Samuelson [71] and Constantinides [14].

Ever since Markowitz, there have been several attempts to formulate the single-period portfolio optimization problem as a linear program. All these attempts have focused on using risk measures, which for discrete approximations of the distributions of returns lead to Linear Programming computable models. Yitzhaki [84] used the Gini's Mean Difference (GMD) as a measure of risk and proposed the so called GMD model. Konno and Yamazaki [47] used the Mean Absolute Deviation (MAD) as a measure of risk and proposed and analyzed the so called MAD model. Another group of risk measures includes the so called downside risk measures. The importance of downside risk measures lies in the behavior of rational investors, who are more interested in the underperformance than the overperformance of portfolios. Markowitz [55] recognized the importance of downside risk measures and proposed the use of semivariance as a measure of risk. For a more comprehensive review of the various risk measures and the single-period portfolio models that exist in the literature, we refer the reader to [53] and references therein.

A significant shift in the era of risk management was made when JP Morgan introduced in 1994 Value-at-Risk (VaR) in order to measure risk across the whole institution. While previous risk measures focused on some theoretical models that related risk to portfolio return, VaR has allowed quantification of risk in term of

possible losses. However, VaR has received a lot of criticism due to lacking sub-additivity, which means that the risk of a portfolio of assets may be greater than the sum of the individual risks of the assets. This makes VaR a non-coherent risk measure in the Artzner et al. sense [3]. A modified version of VaR, called Conditional-Value-at-Risk (CVaR), was introduced in 2000 by Rockafellar and Uryasev (see [70]) and has recently become very popular in the era of financial optimization due to its properties (for applications see for example [2], [38] and [48]). CVaR is a coherent risk measure that accounts for losses which occur in the tails of the distributions of returns and for discrete approximations of the distributions of returns it can be approximated with a linear program, and thus can be easily incorporated in optimization procedures.

Despite the large body of literature in single-period portfolio models, the latter have received a lot of criticism mainly due to their inability to take advantage of (expected) future information when rebalancing the portfolios. Also, they have been found to be inadequate in modeling correctly situations where long-term investors face liabilities and goals at specific dates in the future. To circumvent these inefficiencies, several authors have attempted to model the portfolio selection problem in a multi-period setting. In general, for long term investors multi-period models will perform better than the single-period ones. For a discussion about the advantages of using multi-period models versus single-period ones see [58]. Due to multi-period models being more complex than the single-period ones, in the early years of the modern portfolio theory some authors proposed solving the large-scale multi-period portfolio selection problem as a sequence of single-period portfolio models (see for example [30], [42], [56] and [57]).

Ever since, the rapid advances in computational methods (such as decomposition methods) and technology (enormous improvement in computers' speed and memory) have turned researchers' attention to stochastic programming methods and have allowed solving large-scale problems in various fields, including financial optimization. Stochastic programming has been well studied since the 1950s, when Dantzig [15] and others [13] proposed replacing a stochastic linear problem with a deterministic one assuming a known probability distribution for the random parameters. For applications of stochastic programming methods in multi-period portfolio optimization see, for example, [16], [59] and [86]. For a comprehensive survey of the stochastic programming models used in the era of financial optimization, we refer the reader to [85].

1.2 Thesis Outline

The thesis is structured as follows:

- **Part I: Introduction to the Portfolio Selection Problem.** In this part, we introduce the reader to the portfolio selection problem. Specifically, in chapter 2 we provide the notation used throughout this study, derive the relationships between the variables in the portfolio selection problem, introduce Conditional-Value-at-Risk as a measure of risk and describe how the multivariate process of random returns can be modeled using scenario generation methods.

- **Part II: Dynamic Programming Methods for the Portfolio Selection Problem.** In this part, we formulate the portfolio selection problem as a dynamic program without a measure of risk in chapter 3 and, due to the curses of dimensionality, to solve it in chapter 4 we construct approximate dynamic programming algorithms where we include Conditional-Value-at-Risk as a measure of risk.

- **Part III: Portfolio Selection Methods used as Benchmarks.** In this part, we describe the portfolio selection methods that we use as benchmarks in order to compare with the approximate dynamic programming methods. Specifically, in chapter 5 we use stochastic programming methods to formulate and solve the single-period as well as the multi-period portfolio selection problems and we provide the linear programming formulation of the naive equally-weighted portfolio model.

- **Part IV: Approximate Dynamic Programming Methods.** In this part, we discuss approximate dynamic programming methods. Specifically, we first implement separable linear approximations for the unknown value functions in the dynamic programming formulation of the portfolio selection problem in chapter 6 and then we improve them by implementing a separable piecewise linear approximation in chapter 7.

- **Part V: Experimental Results, Conclusions and Future Research.** In this part, we present our experimental results, draw our conclusions and provide future research directions. Specifically, in chapter 8 we discuss how we evaluate the performance of each method, set up the testing environment for our

experiments, report the numerical results and comment on them. Then, based on our experimental results, in chapter 9 we draw our conclusions and identify possible directions for further research.

Part I

Introduction to the Portfolio Selection Problem

Chapter 2

Basic Concepts and Notation

In this chapter, we summarize the basic concepts in the portfolio selection problem and the notation adopted throughout this study. Specifically, in section 2.1 we define the variables in the portfolio selection problem and derive the relationships between them. Then, in section 2.2 we introduce Conditional-Value-at-Risk as a measure of risk and write it as an optimization problem. Next, in section 2.3 we discuss the utility function of a risk-averse investor and present our objective. Finally, we conclude this chapter with section 2.4, where we discuss how the uncertainty that is introduced by the random returns can be modeled using scenario generation methods.

2.1 Timing of Events and Dynamics

Figure 2.1 shows how transactions evolve in time. Time is divided into equal length sub-periods called slots, such that *period* t corresponds to the time slot between time $t-1$ and time t. We let the *horizon* be the set of all discrete points in time and we denote as $\mathcal{T} = \{0, 1, \ldots, T\}$, where T is the *end of the time horizon*. Suppose we have a portfolio that comprises of N risky assets, such that $\mathcal{N}$ is the set of risky assets, i.e. $\mathcal{N} = \{1, 2, \ldots, N\}$, and a risk-free one. Without loss of generality, we assume that asset 0 is the risk-free asset and is simply a bank account that pays a known constant interest rate at the end of every time period. We denote the rate of return of the risk-free asset in period t with R_{0t}. For simplicity, from this point on we will use term *cash* to refer to the risk-free asset.

Looking at Figure 2.1, in period $t+1$ events evolve in the following sequence: At time t the *decision maker*, i.e. the investor, owns holdings $\boldsymbol{h}_t = (h_{0t}, h_{1t}, \ldots, h_{Nt})$,

where h_{it} is the amount of wealth (in monetary units) in asset category i at time t, and takes actions $(\boldsymbol{x}_t, \boldsymbol{y}_t)$, where $\boldsymbol{x}_t = (x_{1t}, x_{2t}, \ldots, x_{Nt})$ and $\boldsymbol{y}_t = (y_{1t}, y_{2t}, \ldots, y_{Nt})$ are respectively vectors of how much the investor buys and sells (in monetary units) from every risky asset at the beginning of period $t+1$. Every buying/selling decision x_{it}/y_{it} increases/decreases the amount of holdings in asset i at time t by the same amount. That is, every pair of decisions (x_{it}, y_{it}) change the amount of holdings in asset i to $h_{it}^{+} = h_{it} + x_{it} - y_{it}$.

Buying and selling cause transaction costs, which in this study are assumed to be proportional to the trading amount, independent of the asset category and time, and are denoted with θ. Specifically, buying one unit of asset i requires $(1 + \theta)$ units of cash, while selling one unit of asset i increases cash by $(1 - \theta)$ units. Therefore, decisions $(\boldsymbol{x}_t, \boldsymbol{y}_t)$ change the amount of cash from h_{0t} at time t to $h_{0t}^{+} = h_{0t} - (1 + \theta)\sum_{i=1}^{N} x_{it} + (1 - \theta)\sum_{i=1}^{N} y_{it}$. Note that due to transaction costs it is suboptimal to simultaneously buy and sell the same asset. We will see later on that we need not make this assumption as it is directly inferred by the optimization problems that we solve in every time period.

After taking decisions $(\boldsymbol{x}_t, \boldsymbol{y}_t)$, which as explained above change the amount of holdings to $\boldsymbol{h}_t^{+} = \left(h_{0t}^{+}, h_{1t}^{+}, \ldots, h_{Nt}^{+}\right)$, the random return vector $\boldsymbol{R}_{t+1}$ gets realized, where $\boldsymbol{R}_{t+1} = \left(R_{1(t+1)}, R_{2(t+1)}, \ldots, R_{N(t+1)}\right)$ and $R_{i(t+1)}$ is the rate of return of risky asset i at the end of period $t + 1$. Due to the returns, holdings at time $t + 1$ become $\boldsymbol{h}_{t+1} = \left(h_{0(t+1)}, h_{1(t+1)}, \ldots, h_{N(t+1)}\right)$, where $h_{i(t+1)} = R_{it}h_{it}^{+}$. In the above modeling context, we assume that holdings $\boldsymbol{h}_0$ are known to the investor with certainty.

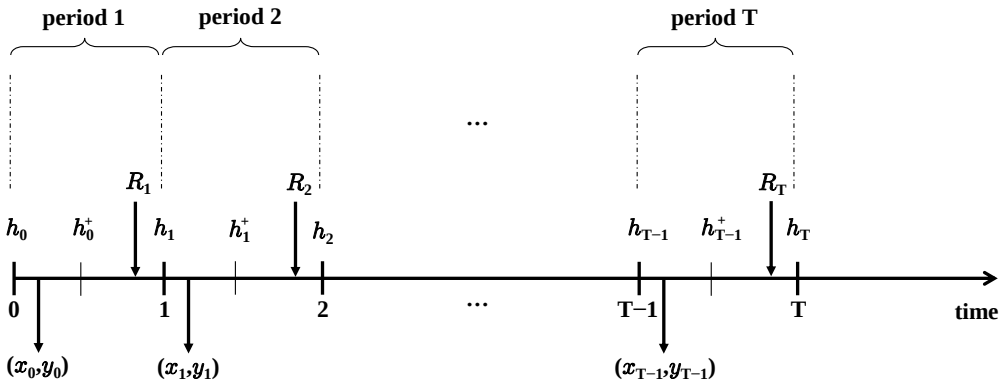


Figure 2.1: Timing of events for the portfolio selection problem

Looking at the timing of events in Figure 2.1, variables $\boldsymbol{h}_t$ are sequenced right

before decisions $(\boldsymbol{x}_t, \boldsymbol{y}_t)$, are called the *pre-decision* holdings and, as we will see in chapter 3, these become our state variables in the optimal dynamic programming formulation of the portfolio selection problem. Variables $\boldsymbol{h}_t^+$ are sequenced right after decisions $(\boldsymbol{x}_t, \boldsymbol{y}_t)$, are called the *post-decision* holdings and, as we will see in chapter 4, these become our new state variables in approximate dynamic programming. Further, variables $\boldsymbol{h}_t^+$ are used in the two-stage stochastic programming formulation of the single-period portfolio selection problem, the multistage stochastic programming formulation of the multi-period portfolio selection problem and the equally-weighted portfolio model in chapter 5.

We are now ready to write the relationships between the variables in the portfolio selection problem.

Considering the above and looking at Figure 2.1, we obtain the relationships between the pre-decision variables:

$$\left.\begin{aligned} h_{i(t+1)} &= R_{i(t+1)}\left(h_{it} + x_{it} - y_{it}\right), \;\; i \in \mathcal{N} \\ h_{0(t+1)} &= R_{0(t+1)}\left[h_{0t} - (1+\theta)\sum_{i=1}^{N} x_{it} + (1-\theta)\sum_{i=1}^{N} y_{it}\right] \end{aligned}\right\} \tag{2.1}$$

where note that first decisions are taken and then returns are realized, and the post-decision variables:

$$\left.\begin{aligned} h_{i0}^+ &= h_{i0} + x_{i0} - y_{i0}, \;\; i \in \mathcal{N} \\ h_{00}^+ &= h_{00} - (1+\theta)\sum_{i=1}^{N} x_{i0} + (1-\theta)\sum_{i=1}^{N} y_{i0} \\ h_{it}^+ &= R_{it} h_{i(t-1)}^+ + x_{it} - y_{it}, \;\; i \in \mathcal{N},\; t = 1, \ldots, T-1 \\ h_{0t}^+ &= R_{0t} h_{0(t-1)}^+ - (1+\theta)\sum_{i=1}^{N} x_{it} + (1-\theta)\sum_{i=1}^{N} y_{it},\; t = 1, \ldots, T-1 \end{aligned}\right\} \tag{2.2}$$

where note that first returns are realized and then decisions are taken.

Further, the pre- and the post-decision variables have the following relationship:

$$h_{it} = R_{it} h_{i(t-1)}^+, \;\; i \in \mathcal{N} \cup \{0\},\; t = 1, \ldots, T \tag{2.3}$$

If υ^T denotes the investor's total wealth at the end of the time horizon, then we

have:

$$v^T = \sum_{i=0}^{N} h_{iT} = \sum_{i=0}^{N} R_{iT} h^+_{i(T-1)} \tag{2.4}$$

In this study, we do not consider either shortselling of assets or borrowing of cash. Instead, we assume non-negativity of the pre- and post-decision variables, as well as of the amounts traded (either bought or sold) in every time period. Non-negativity can be expressed by the following set of inequalities:

$$\left.\begin{aligned} &h_{it} \geq 0, \;\; i \in \mathcal{N} \cup \{0\}, \; t = 1, \ldots, T \\ &h^+_{it} \geq 0, \;\; i \in \mathcal{N} \cup \{0\}, \; t = 0, \ldots, T-1 \\ &x_{it} \geq 0, \;\; i \in \mathcal{N}, \; t = 0, \ldots, T-1 \\ &y_{it} \geq 0, \;\; i \in \mathcal{N}, \; t = 0, \ldots, T-1 \end{aligned}\right\} \tag{2.5}$$

where, as we will explain later on, due to transaction costs x_{it} and y_{it} are never both positive.

2.2 Conditional-Value-at-Risk as a Measure of Risk

One essential aspect in portfolio optimization is risk measurement. In this study, we use *Conditional-Value-at-Risk* (CVaR), also known as *mean excess loss* or *mean shortfall* or *tail-VaR*, as a measure of risk. The reason why we selected in this study CVaR as the risk measure is because of the properties it exhibits. Specifically, CVaR is a coherent downside risk measure which accounts for possible extreme losses that occur in the tails of loss distributions. Moreover, for a discrete set of scenarios it can be approximated with a linear program, thus allowing us to incorporate it easily in optimization procedures. In the discussion that follows, we briefly state the definition of CVaR in line with [70], its properties and its representation as a linear program for a discrete approximation of the random input.

Definition

Let $f(X, Y)$ be the loss associated with decision vector $X \in \mathcal{X}$ and random vector $Y \in \mathcal{Y}$. Also, let $p(Y)$ be the density function of the probability distribution of Y. Then, the probability of loss $f(X, Y)$ being less than or equal to a threshold g_0 is given by:

$$\Psi(X, g_0) = \int_{f(X,Y)\leq g_0} p\,(Y)\,dY \tag{2.6}$$

For a specified probability level $\beta \in (0, 1)$ and the losses associated with decision vector X, the values of Value-at-Risk (VaR$_\beta$) and Conditional-Value-at-Risk (CVaR$_\beta$) are denoted respectively with $\phi_\beta(X)$ and $\chi_\beta(X)$. These are given by:

$$\phi_\beta(X) = \min\left\{g_0 \in \mathbb{R} :\ \Psi(X, g_0) \geq \beta\right\}, \tag{2.7}$$

$$\chi_\beta(X) = \frac{1}{1-\beta}\int_{f(X,Y)\geq \phi_\beta(X)} f(X,Y)p\,(Y)\,dY \tag{2.8}$$

The key to expressing CVaR$_\beta$ as a linear minimization problem is the following function:

$$F_\beta(X, g_0) = g_0 + \frac{1}{1-\beta}\int_{Y\in\mathcal{Y}} [f(X,Y) - g_0]^+\,p\,(Y)\,dY \tag{2.9}$$

where $[k]^+ = k$ if $k > 0$ and $[k]^+ = 0$ if $k \leq 0$.

In optimization problems, we are interested in minimizing CVaR$_\beta$ over all $X \in \mathcal{X}$. According to theorem 2 in [70], minimizing $\chi_\beta(X)$ over all $X \in \mathcal{X}$ is equivalent to minimizing $F_\beta(X, g_0)$ over all $(X, g_0) \in \mathcal{X} \times \mathbb{R}$, i.e. we have:

$$\min_{X\in\mathcal{X}} \chi_\beta(X) = \min_{(X,g_0)\in\mathcal{X}\times\mathbb{R}} F_\beta(X, g_0), \tag{2.10}$$

where the optimal value of problem (2.10) gives us CVaR$_\beta$ and $g_0^\star$ gives us VaR$_\beta$.

Figure 2.2 provides a graphical illustration of VaR$_\beta$ and CVaR$_\beta$ on a discrete loss distribution, where note that CVaR$_\beta$ accounts for losses beyond VaR$_\beta$ or alternatively VaR$_\beta$ never exceeds CVaR$_\beta$. Specifically, VaR$_\beta$ is the loss threshold beyond which there is a probability $1-\beta$ that higher losses occur, and CVaR$_\beta$ is the average of the losses that lie in the $(1-\beta)$-area of the loss distribution. In this thesis we are not examining the impact that different values of quantile β have on estimating CVaR$_\beta$. Instead, in our experiments in chapter 8 we assume a fixed value for parameter β so from this point on we will drop β from our notation.

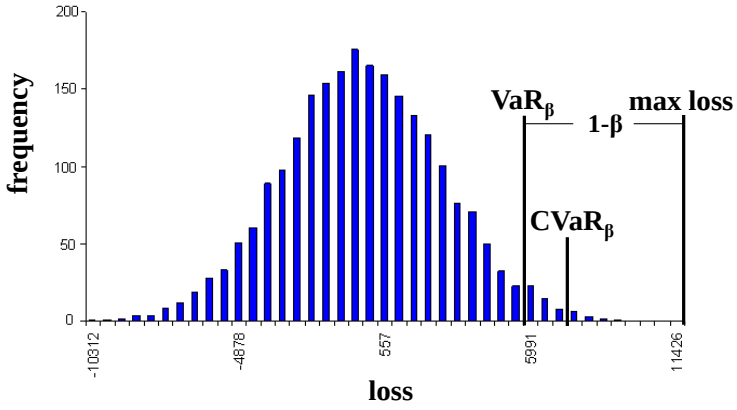


Figure 2.2: An illustration of VaR$_\beta$ and CVaR$_\beta$ on a discrete loss distribution

Properties of CVaR

With respect to loss function $f(\cdot)$, which is a random variable, CVaR satisfies the following properties:

1. CVaR is *translation-equivariant*. That is,

$$\text{CVaR}\,(f+c) = \text{CVaR}\,(f) + c \tag{2.11}$$

2. CVaR is *positively-homogeneous*. That is,

$$\text{CVaR}\,(cf) = c\,\text{CVaR}\,(f)\,, \tag{2.12}$$

   if $c > 0$.

3. CVaR is *convex*. That is, for two arbitrary random losses f_1 and f_2 and $0 < \lambda < 1$ we have:

$$\text{CVaR}\,(\lambda f_1 + (1-\lambda) f_2) \leq \lambda \text{CVaR}\,(f_1) + (1-\lambda)\text{CVaR}\,(f_2) \tag{2.13}$$

4. CVaR is *monotone*. That is, for any arbitrary random losses f_1 and f_2 if $f_1 \leq f_2$, then we have:

$$\text{CVaR}\,(f_1) \leq \text{CVaR}\,(f_2) \tag{2.14}$$

For the proofs of the above properties, we refer the reader to [82]. In the Artzner, Delbaen, Eber and Heath sense [3], a risk measure that exhibits properties (2.11)-(2.14) is called *coherent*.

Polyhedral Representation of CVaR

We are now ready to derive the *polyhedral* representation of CVaR.

Suppose we sample S scenarios from the distribution of random variable Y. If $\mathcal{S}$ is the set containing all scenarios, i.e. $\mathcal{S} = \{1, 2, \ldots, S\}$, and p_s is the probability associated with scenario s, then the integral in (2.9) can be approximated by:

$$\tilde{F}_\beta(X, g_0) = g_0 + \frac{1}{1-\beta} \sum_{s=1}^{S} p_s \left[f(X, Y_s) - g_0 \right]^+ \tag{2.15}$$

If we replace in (2.15) every $[f(X, Y_s) - g_0]^+$ with auxiliary variable g_2^s, we can approximate CVaR with the optimal value of the following linear program:

$$\left.\begin{array}{ll} \min\limits_{\substack{(X, g_0, g_2^s) \in \\ \mathcal{X} \times \mathbb{R} \times \mathbb{R}_+}} & g_0 + \dfrac{1}{(1-\beta)} \displaystyle\sum_{s=1}^{S} p_s g_2^s \\ \text{s.t.} & g_2^s \geq -g_0 + f(X, Y_s), \;\; s \in \mathcal{S} \\ & g_0 \;\; \text{free}, g_2^s \geq 0, \;\; s \in \mathcal{S} \end{array}\right\} \tag{2.16}$$

Using problem (2.16), later on in chapter 4 we define CVaR for the multi-period portfolio selection problem and we adapt it for each other method in the respective chapters.

2.3 Investor's Objective

The goal of the investor is usually expressed as the expected utility of some function of terminal wealth:

$$\max \mathbb{E}[U(v^T)] \tag{2.17}$$

where function $U(v^T)$ describes the risk attitude of the investor. Specifically, utility function $U(v^T)$ is modeled as a *concave function* as in Figure 2.3. A concave utility function means that we value every additional unit of wealth less and less thus protecting ourselves from high losses on a sudden downward move of the returns.

An investor with a concave utility function is said to be *risk-averse*. For a more detailed discussion about utility theory and risk-aversion, we refer the reader to chapter 9 of [52].

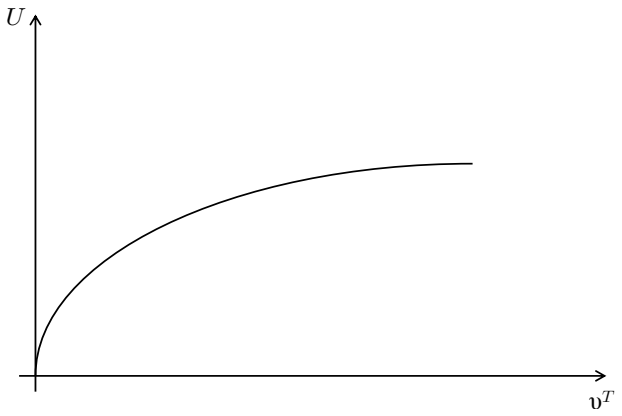


Figure 2.3: A concave utility function

In this study, we consider a mean-risk objective function, according to which the investor aims at co-optimizing a weighted average of the expected terminal wealth and CVaR:

$$\max \gamma \mathbb{E}\left\{v^T\right\} - (1-\gamma)\text{CVaR}(v^T) \tag{2.18}$$

where parameter γ measures how important is risk in the objective, takes values in the range $[0, 1]$, and will be called the *risk importance parameter*. In problems with long horizons, in order to account for the time value of money, we usually see a discount factor outside the expectation of objective (2.18). However, in the portfolio selection problem the time value of money is accounted by the asset returns and thus we disregard it.

Note that objective (2.18) is a concave function of v^T due to CVaR being convex with respect to v^T (this follows from property 2.13, where the loss function is now given by $-v^T$ plus a constant). To understand how the risk attitude of the investor is affected by the value of parameter γ, we can form ratio $\frac{1-\gamma}{\gamma}$, which simply tells us how important CVaR is as compared to expected terminal wealth, takes values in the range $[0, \infty)$, and is usually called the *risk aversion coefficient*. If $\frac{1-\gamma}{\gamma} = 0$, then the investor is infinitely risk-taking, while if $\frac{1-\gamma}{\gamma} \to \infty$ then the investor is infinitely risk-averse.

In expression (2.18), varying parameter γ in the range $[0, 1]$ we obtain the *effi-*

cient frontier of terminal wealth, every point of which represents the best possible expected terminal wealth for a given level of CVaR. The efficient frontier is a hyperbola, reflecting the investor's risk-averse attitude (he expects to earn less and less the more he is exposed to higher losses).

2.4 Modeling Uncertainty

The random returns of the risky assets in equations (2.1)-(2.4) introduce uncertainty that can be described by the multivariate stochastic process of random returns and is usually represented in stochastic optimization with probability spaces.

For the stochastic process of random returns we define probability space $(\Omega, \mathcal{F}, \mathcal{P})$, where:

1. Ω is the continuous sample space consisting of all realizations ω, where $\omega = (\boldsymbol{R}_1, \boldsymbol{R}_2, \ldots, \boldsymbol{R}_T)$ is a particular instance of the complete set of returns and is usually referred to as a *scenario*.

2. $\mathcal{F}$ is the σ-algebra on Ω, i.e. a non-empty collection of subsets of Ω that includes Ω. In chapter 5, we will need $\mathcal{F}_t$, which is the σ-algebra on $(\boldsymbol{R}_1, \boldsymbol{R}_2, \ldots, \boldsymbol{R}_t)$, i.e. $\mathcal{F}_t$ is a collection of all events determined by information available up to time t. For set $\mathcal{F}_t$, we have $\mathcal{F}_t \subset \mathcal{F}_{t+1}$ and is called a *filtration*. When a policy depends on information available up to time t, then this policy is called *non-anticipative* and for the respective decisions we say that they are $\mathcal{F}_t$-measurable.

3. $\mathcal{P}$ is a function that maps any subset of Ω into the unit interval $[0, 1]$, such that $\mathcal{P}(\Omega) = 1$.

For a further discussion on probability spaces and measures see [9] and [72].

When formulating stochastic problems, probability space $(\Omega, \mathcal{F}, \mathcal{P})$ is assumed to be known. In order to obtain a finite and discrete probability space that results in a tractable stochastic model, we need to generate a discrete approximation of the probability space using *scenario generation methods*. In the literature, most scenario generation methods have been developed for multistage stochastic programming and are based upon:

1. the selection of a *model* that explains the behavior of the random variables (such as econometric models for rates of returns, etc.) and is used in order to forecast data trajectory paths.

2. the construction of a *scenario tree*.

For a comprehensive survey of the different models that have been used in the literature to generate data trajectory paths and the different scenario tree generation methods see [24] (other examples can be found in [25] and [41]).

Among the different models that have been used to generate data trajectories, econometric models have gained particular attention and have been used extensively to model and forecast the conditional variance, else known as *volatility*. A typical feature of financial time series is *volatility clustering*, according to which large volatilities tend to be followed by large volatilities, while small volatilities tend to be followed by small volatilities. Any changes from large volatilities to small ones and vice versa occur randomly without exhibiting any systematic pattern. Further, plotting the sample autocorrelation function (ACF) of the squared returns, one notices that squared returns exhibit strong positive autocorrelation, which provides more evidence of volatility clustering. Observations of this type led to the introduction of the so-called Generalized Autoregressive Conditional Heteroskedasticity (GARCH) models (see [11] and [28]).

Univariate GARCH models have been extended to a multivariate setting in order to study the co-movements of the assets' volatilities by modeling and forecasting a positive definite conditional covariance matrix. The proposed models are known as multivariate GARCH models (MGARCH) (see [4] for a comprehensive survey). However, direct generalizations of the univariate GARCH models lead to intractable models as the dimensionality of the stochastic process becomes larger and larger and this is due to the large number of parameters that need to be estimated (this is often referred to as the *curse of dimensionality*). This is the reason why these models have not been used in multivariate time series of more than three or four dimensions.

To circumvent the curse of dimensionality and obtain tractable models, additional structures are imposed in order to reduce the dimensionality of the stochastic processes. One, very popular for its simplicity, method suggests the generation of the covariance matrix through an orthogonal factorization of the assets using principal component analysis. This class of models are known as Orthogonal GARCH (OGARCH) models (see [1]) and in comparative studies against other MGARCH models they have exhibited exceptional forecasting ability (see for example [12] and [29]).

In this study, we generate scenario paths assuming a constant conditional mean and forecasting the covariance matrix using OGARCH models. For a further discus-

sion about how we can generate scenario paths with OGARCH models, we refer the reader to Appendix A. However, as we will explain in chapter 5, in the multistage stochastic programming method we will need to approximate the input distribution of scenario paths with another one that has less scenarios and is in the form of a scenario tree using *scenario reduction* and *scenario tree construction* methods. Since scenario trees are only used in the multistage stochastic programming method, a discussion about them and how they can be constructed is included in the respective chapter.

Part II

Dynamic Programming Methods for the Portfolio Selection Problem

Chapter 3

Dynamic Programming Formulation for the Portfolio Selection Problem

In this chapter, we formulate the multi-period portfolio selection problem as a *dynamic program* (DP) without using a risk measure. As we will explain later in the chapter, we cannot include a risk measure such as CVaR in the dynamic programming formulation because then the problem does not decompose in time. Later on, in chapter 4, we incorporate CVaR as a risk measure in our approximation algorithms.

This chapter is structured as follows: In section 3.1, we define the basic elements of the portfolio selection problem as a *markov decision process* (MDP), state the objective function and derive the optimality equations. Then, in section 3.2, we discuss the *curses of dimensionality*, where we explain why we cannot solve the dynamic program defined in section 3.1 with the available dynamic programming algorithms and as a result we resort to *approximate dynamic programming*. Finally, we conclude with section 3.3 where we explain why we cannot include CVaR in the optimal dynamic programming formulation.

More details regarding dynamic programming methods can be found in [7] and [69].

3.1 Formulation

In line with [69], we begin with defining the five elements of a markov decision process, i.e. decision epochs, states, actions, transition probabilities and rewards, which are related as follows: At each decision epoch the system occupies a state.

The decision maker observes the current state of the system and selects an action from a set of allowable actions. As a result of choosing an action in the current state of the system, the decision maker receives some reward and the system transits to some other state determined by some probability distribution. Note that in some cases the reward might also depend on the probability distribution. Given that we currently occupy a certain state, any future state/decision/outcome is independent of how we reached the current state, which is known as the *markov property*.

Decision Epochs and Periods

A *decision epoch* refers to a discrete point in time. We let the *horizon* be the set of all the decision epochs and we denote it with $\mathcal{T} = \{0, 1, \ldots, T\}$. We divide time into *periods*, such that period t corresponds to the time slot between decision epoch $t-1$ and decision epoch t. The *end of the time horizon* refers to the last decision epoch, which we denote with T. Figure 3.1 shows decision epochs and periods for a MDP.

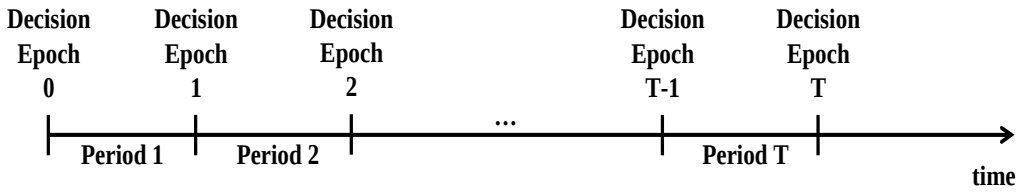


Figure 3.1: Decision Epochs and Periods for a MDP

The State of the System and the Stochastic Process

Suppose sets $\mathcal{H}_i$ and $\mathcal{R}_i$ are respectively subsets of $\mathbb{R}_+$ and $\mathbb{R}_+ \setminus \{0\}$.

We let the *state of the system* be described by vector $\boldsymbol{h}_t = (h_{0t}, h_{1t}, \ldots, h_{Nt})$, where $\boldsymbol{h}_t \in \mathcal{H} = \mathcal{H}_0 \times \mathcal{H}_1 \times \cdots \mathcal{H}_N$, and h_{it} is the amount of holdings (in monetary units) of asset category i at time t and takes values in $\mathcal{H}_i$.

An important aspect of our problem is the arrival of exogenous information, which, as explained in section 2.4, is described by the stochastic process of random returns. A realization of the stochastic process at time t is described by vector $\boldsymbol{R}_t = (R_{1t}, R_{2t}, \ldots, R_{Nt})$, where $\boldsymbol{R}_t \in \mathcal{R} = \mathcal{R}_1 \times \mathcal{R}_2 \times \cdots \mathcal{R}_N$, and R_{it} is the rate

of return of asset i at time t and takes values in $\mathcal{R}_i$. Recall that the returns of cash, R_{0t}, are assumed to be known with certainty for every t.

Policies and Decision Rules

The decisions associated with the portfolio selection problem are to determine how much the *decision maker*, i.e. the investor, buys and sells from every asset in every time period. Let $\boldsymbol{x}_t = (x_{1t}, x_{2t}, \ldots, x_{Nt})$ and $\boldsymbol{y}_t = (y_{1t}, y_{2t}, \ldots, y_{Nt})$ be respectively the buying and selling decision vectors, with x_{it} and y_{it} representing respectively how much the investor buys and sells from risky asset i at time t. We assume that $(\boldsymbol{x}_t, \boldsymbol{y}_t) \in \mathcal{A}_t$, where $\mathcal{A}_t$ is the set of constraints that determine all the feasible decisions for all assets at time t.

We define a *decision rule* to be a function that takes as input the current state variables and returns a vector of feasible decisions:

$$D_t(\boldsymbol{h}_t) = (\boldsymbol{x}_t, \boldsymbol{y}_t) \tag{3.1}$$

We define a *policy* π to be a set of decision rules over all periods:

$$\pi = \big(D_0^\pi(\boldsymbol{h}_0), D_1^\pi(\boldsymbol{h}_1), \ldots, D_{T-1}^\pi(\boldsymbol{h}_{T-1})\big), \;\; \pi \in \Pi, \tag{3.2}$$

where Π is the set of all feasible policies.

If we combine the equations in (2.1) with the non-negativity inequalities in (2.5), we obtain action space $\mathcal{A}_t$ for every $t = 0, 1, \ldots, T-1$, which is the set of values $(\boldsymbol{x}_t, \boldsymbol{y}_t)$ that satisfy the following constraints:

$$-x_{it} + y_{it} \leq h_{it}, \;\; i \in \mathcal{N}, \tag{3.3}$$

$$(1+\theta)\sum_{i=1}^{N} x_{it} - (1-\theta)\sum_{i=1}^{N} y_{it} \leq h_{0t}, \tag{3.4}$$

$$x_{it}, y_{it} \geq 0, \;\; i \in \mathcal{N}, \tag{3.5}$$

where constraint (3.3) ensures non-negativity of the holdings of the risky assets, constraint (3.4) ensures non-negativity of cash and will be called *the budget constraint*, and finally constraint (3.5) ensures non-negativity of the buying and selling decisions of the risky assets.

From the above, the action space can be expressed as the following set:

$$\mathcal{A}_t = \{(\boldsymbol{x}_t, \boldsymbol{y}_t) : \ (3.3) - (3.5) \text{ hold}\}, \ t = 0, 1, \ldots, T-1 \tag{3.6}$$

Information Process

Looking at Figure 3.2, we notice that in every time period the events of the multi-period portfolio selection problem are disclosed in the following order: state variables (i.e. holdings) at the beginning of the period, decisions, random returns and state variables (i.e. holdings) at the end of the period.

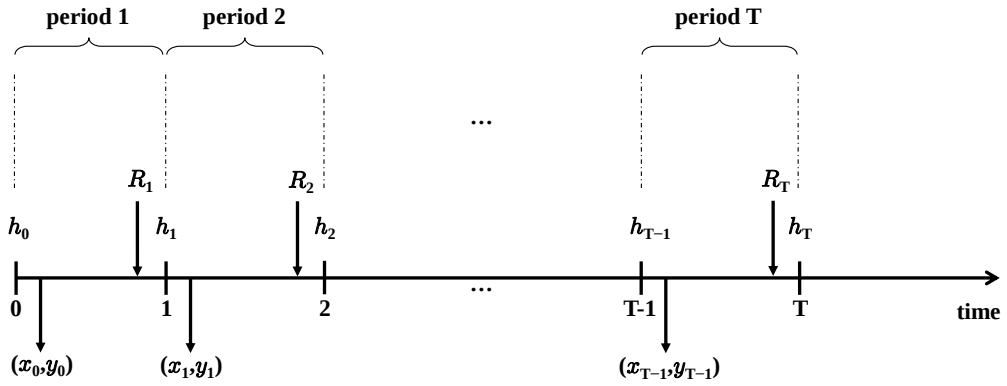


Figure 3.2: Timing of events in the portfolio selection problem

Given the above modeling of the time horizon, for every period t we let the *history* of the process, else known as the *information* process, consist of all the information known to the system up to and including period t. The decisions and the states of the system represent the *endogenous* information, while the random returns represent the *exogenous* information. The *history* of the process for some policy $\pi \in \Pi$ up to time t can then be described as follows:

$$H_t = \left(\boldsymbol{h}_0, D_0^{\pi}(\boldsymbol{h}_0), \boldsymbol{R}_1, \boldsymbol{h}_1, \ldots, \boldsymbol{h}_{t-1}, D_{t-1}^{\pi}(\boldsymbol{h}_{t-1}), \boldsymbol{R}_t, \boldsymbol{h}_t\right) \tag{3.7}$$

Transition Functions

Assuming that at time $t-1$ the state of the system is $\boldsymbol{h}_{t-1}$ and we take decisions $\left(\boldsymbol{x}_{t-1}, \boldsymbol{y}_{t-1}\right)$, we can compute the state of the system at time t using (2.1), according to which:

$$\left.\begin{aligned} h_{it} &= R_{it}\left[h_{i(t-1)} + x_{i(t-1)} - y_{i(t-1)}\right], \ i \in \mathcal{N} \\ h_{0t} &= R_{0t}\left[h_{0(t-1)} - (1+\theta)\sum_{i=1}^{N} x_{i(t-1)} + (1-\theta)\sum_{i=1}^{N} y_{i(t-1)}\right] \end{aligned}\right\} \tag{3.8}$$

The above equations show how we transit from a current state to the next feasible state and are called the *transition functions*.

Rewards and Objective

The objective of the investor is to find the best policy that maximizes expected terminal wealth which from (2.4) is a function of state variables h_{iT} and is given by $\upsilon^T = \sum_{i=0}^{N} h_{iT}$. We define the following function:

$$C_t^{\pi} = \begin{cases} 0, & \text{if } t \in \mathcal{T} \setminus \{0, T\} \\ \sum_{i=0}^{N} h_{iT}^{\pi}, & \text{if } t = T \end{cases}$$

to be the *reward* achieved at time t given that we follow policy π, where h_{iT}^{π} is the amount of terminal holdings in asset i when policy π is followed.

Given the above definition, the objective of the investor can now be expressed as follows:

$$\max_{\pi \in \Pi} \ \mathbb{E}\left\{\sum_{t=1}^{T} C_t^{\pi}\right\} \tag{3.9}$$

Note that in optimization problem (3.9) we do not use a discount factor, since the opportunity cost of cash held into assets is directly calculated by the returns of the assets.

Optimality Equations

In most problems, optimization problem (3.9) is computationally intractable. To reduce complexity, we formulate it as a dynamic program using recursive equations. We introduce the *value function* $V_t(\boldsymbol{h}_t)$, which represents the value of being in state $\boldsymbol{h}_t$ at time t, i.e. the value of owning holdings $\boldsymbol{h}_t$ at time t. Therefore, with each state $\boldsymbol{h}_t$ we associate a value function $V_t(\boldsymbol{h}_t)$ as shown in Figure 3.3.

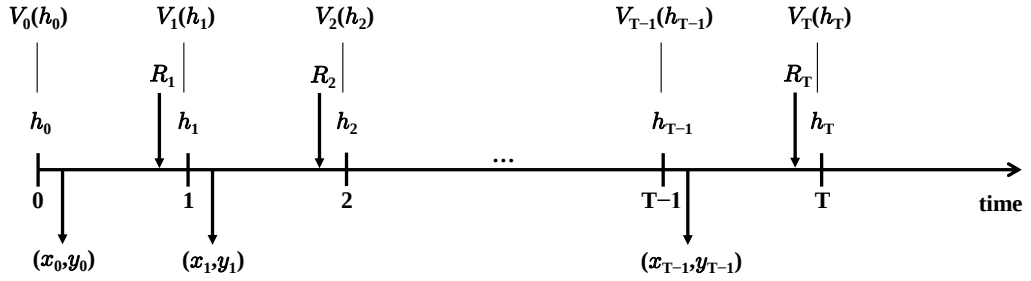


Figure 3.3: Modeling time horizon: states and value functions

The quantity $V_t(\boldsymbol{h}_t)$ gives us the total optimal reward from time t until the end of the time horizon, given that at time t we are in state $\boldsymbol{h}_t$, and is given by:

$$\left.\begin{aligned} V_t(\boldsymbol{h}_t) &= \max_{(\boldsymbol{x}_t,\boldsymbol{y}_t)\in\mathcal{A}_t} \mathbb{E}_{\mathbf{R}_{t+1}}\left\{V_{t+1}(\boldsymbol{h}_{t+1}) \mid \boldsymbol{h}_t\right\},\ t=0,1,\ldots,T-1 \\ V_T(\boldsymbol{h}_T) &= \sum_{i=0}^{N} h_{iT} \end{aligned}\right\} \tag{3.10}$$

The recursive equations in (3.10) give us the relationship between $V_t(\boldsymbol{h}_t)$ and $V_{t+1}(\boldsymbol{h}_{t+1})$ for $t = 0,1,\ldots,T-1$ and are known as the *Bellman's optimality equations*. Note that inside the expectation of the optimality equations $\boldsymbol{h}_t$ is constant but $\boldsymbol{h}_{t+1}$ is a function of $\boldsymbol{R}_{t+1}$ which is a random variable.

Normally we would expect to see a reward function and a discount factor in the optimality equations of (3.10). However, as explained earlier, here all the one-period rewards are zero and the investor receives the total reward only once at the end. Further, the time value of money is accounted by the returns and we do not need to use a discount factor.

For a discrete approximation of the joint distribution of the random returns and a discrete approximation of the states, we can replace the expectation in (3.10) with transition probabilities and write the optimality equations as follows:

$$\left.\begin{aligned} V_t(\boldsymbol{h}_t) &= \max_{(\boldsymbol{x}_t,\boldsymbol{y}_t)\in\mathcal{A}_t} \sum_{\boldsymbol{h}_{t+1}\in\mathcal{H}} p\left(\boldsymbol{h}_{t+1} \mid \boldsymbol{h}_t, \boldsymbol{x}_t, \boldsymbol{y}_t\right) V_{t+1}\left(\boldsymbol{h}_{t+1}\right),\ t=0,1,\ldots,T-1 \\ V_T(\boldsymbol{h}_T) &= \sum_{i=0}^{N} h_{iT} \end{aligned}\right\} \tag{3.11}$$

where $p\left(\boldsymbol{h}_t \mid \boldsymbol{h}_{t-1}, \boldsymbol{x}_{t-1}, \boldsymbol{y}_{t-1}\right)$ denotes the probability that the system transits

to state $\boldsymbol{h}_t$ given that we choose action $\left(\boldsymbol{x}_{t-1}, \boldsymbol{y}_{t-1}\right)$ when we are in state $\boldsymbol{h}_{t-1}$ at time $t-1$.

3.2 Dynamic Programming and the Curses of Dimensionality

In classical dynamic programming, authors solve *finite horizon* problems, such as the one under study, using *backward dynamic programming* (see [65] and [69]) which assumes that we have discrete action, outcome and state spaces. Solving a dynamic program with backward dynamic programming is straightforward: At the end of the time horizon, we compute for each terminal state the terminal value function. Then, we recursively step back one time period computing the value function for all states using the optimality equations. In this manner, in the end we will have computed the value function at time 0 which is the optimal value of problem (3.9).

In our problem, however, we have continuous action/outcome/state spaces. Even if we discretize actions and states to the nearest pound and also discretize returns the problem is still hard to solve due to the three *curses of dimensionality* (see [65]). The first one comes from the state space which in our problem grows intractably large because the state of the system is described by vector $\boldsymbol{h}_t$ which has dimension $N+1$. The second one comes from the outcome space which does not allow us to compute the expectation in the optimality equations of (3.10). This expectation is due to the random return vector $\boldsymbol{R}_t$ which has dimension N. Finally, enumerating all decisions from the action space to solve the optimization problem in each time period would result in the third curse of dimensionality that comes from the action space which has dimension $2N$.

In chapter 4, we attempt to circumvent the three curses of dimensionality by using approximate dynamic programming. In particular, by fitting value function approximations for the unknown value functions we treat the complexity that comes from the state space. By using Monte-Carlo sampling methods we treat the complexity that comes from the outcome space. Finally, we introduce fast algorithms to compute our decisions and this treats the complexity that comes from the action space.

3.3 CVaR and Dynamic Programming

From section 2.2, CVaR is the optimal value of the following minimization problem:

$$\min_{X \in \mathcal{X}} \frac{1}{1-\beta} \int_{f(X,Y) \geq \text{VaR}_\beta(X)} f(X,Y) p\,(Y)\, dY,$$

which can also be written using conditional expectation as follows:

$$\min_{X \in \mathcal{X}} \frac{1}{1-\beta} \mathbb{E}_Y \left\{ f(X,Y) | f(X,Y) \geq \text{VaR}_\beta(X) \right\}, \tag{3.12}$$

Thus, given decision vector X and random variable Y which result in losses $f(X,Y)$, CVaR is a minimization problem that contains a conditional expectation on losses $f(X,Y)$. The above occurs in a single-period setting.

In the multi-period portfolio selection problem, the losses are given by $-\sum_{i=0}^{N} h_{iT}$ $+\sum_{i=0}^{N} h_{i0}$, where $\sum_{i=0}^{N} h_{i0}$ is a constant, and from transition equations (3.8) we notice that in order to compute terminal wealth $\sum_{i=0}^{N} h_{iT}$ we need a mixture of decisions and realizations of returns for the entire horizon. This implies that in order to include CVaR in the optimal dynamic programming formulation we would need to take an expectation over $\sum_{i=0}^{N} h_{iT}$ that depends on states/actions/realizations of the entire horizon. However, given that in markov decision systems the state contains all we need to know in order to take our decisions, decomposing such an expectation in time would violate the markovian property. Thus, we cannot include CVaR in the optimal dynamic programming formulation. In chapter 4, where we introduce approximate dynamic programming algorithms, CVaR can be easily incorporated and adaptively estimated.

Chapter 4

Approximate Dynamic Programming

In chapter 3, we formulated the portfolio selection problem as a dynamic program which becomes computationally intractable due to the curses of dimensionality. This is where *approximate dynamic programming* (ADP) arises, providing us with a powerful set of modeling and algorithmic strategies to solve large-scale dynamic programs. Specifically, to deal with the complexity that comes from the large state space, where we need to evaluate value function $V_t(\boldsymbol{h}_t)$ for every state $\boldsymbol{h}_t \in \mathcal{H}$, we fit value function approximations for the unknown value functions. To deal with the complexity that comes from the large outcome space, where we need to evaluate the expectation in (3.10), we use Monte-Carlo sampling. Finally, to deal with the complexity that comes from the large action space, we introduce fast algorithms that compute our decisions.

This chapter is structured as follows: In section 4.1, we introduce in the dynamic programming formulation the post-decision state variables and we write the new optimality equations with respect to the new state variables. Then, in section 4.2, we derive a formula that computes CVaR given a sample of losses. Next, in section 4.3, we construct an iterative algorithm which uses the new state variables and solves the portfolio selection problem by stepping forward through time in every iteration. Finally, in section 4.4, we concentrate on update issues of the value functions in every iteration of the proposed algorithm.

For a more detailed discussion about theory and applications of approximate dynamic programming methods we refer the reader to [65] and references therein.

4.1 Pre-decision and Post-decision State Variables

One of the biggest challenges in dynamic programming is computing the expectation within the max operator of the optimality equations of (3.10). In this section, we consider an approximation of the value function using Monte-Carlo sampling. As we will explain later on, to develop and implement our approximation methods we need the concept of post-decision state variables, which allow us to compute the suboptimal decisions using past information.

If $\omega = (\boldsymbol{R}_1, \boldsymbol{R}_2, \ldots, \boldsymbol{R}_T)$ is a sample realization of the stochastic process, then we propose the following approximation:

$$\hat{V}_t(\boldsymbol{h}_t(\omega)) = \max_{(\boldsymbol{x}_t, \boldsymbol{y}_t) \in \mathcal{A}_t} \hat{V}_{t+1}\left(\boldsymbol{h}_{t+1}(\omega)\right), \tag{4.1}$$

Recall from chapter 3 that equations (3.8) describe the transitions from one state to another . Looking at these equations, we notice that state vector $\boldsymbol{h}_{t+1}$ is a function of return vector $\boldsymbol{R}_{t+1}$. That is, in order to compute the suboptimal decisions $(\boldsymbol{x}_t, \boldsymbol{y}_t)$ in the optimization problem of (4.1), we need to use future information from time $t+1$. In order to correct this problem, we change the time at which state is measured so that state at time t is measured just before the arrival of information $\boldsymbol{R}_{t+1}$ and right after decisions $(\boldsymbol{x}_t, \boldsymbol{y}_t)$ are made.

Recall from section 3.1 that the information process up to time t is given by:

$$H_t = \left(\boldsymbol{h}_0, D_0^\pi(\boldsymbol{h}_0), \boldsymbol{R}_1, \boldsymbol{h}_1, D_1^\pi(\boldsymbol{h}_1), \boldsymbol{R}_2, \boldsymbol{h}_2, \ldots, \boldsymbol{h}_{t-1}, D_{t-1}^\pi(\boldsymbol{h}_{t-1}), \boldsymbol{R}_t, \boldsymbol{h}_t\right)$$

Measuring the state after decisions are made suggests the definition of new state variables $\boldsymbol{h}_t^+$ (recall that these were defined in section 2.1 and were called the post-decision variables):

$$\left.\begin{aligned} h_{it}^+ &= h_{it} + x_{it} - y_{it}, \;\; i \in \mathcal{N}, \; t = 0, 1, \ldots, T-1 \\ h_{0t}^+ &= h_{0t} - (1+\theta)\sum_{i=1}^{N} x_{it} + (1-\theta)\sum_{i=1}^{N} y_{it}, \; t = 0, 1, \ldots, T-1 \end{aligned}\right\} \tag{4.2}$$

In approximate dynamic programming, variables $\boldsymbol{h}_t$ and $\boldsymbol{h}_t^+$ are called respectively *pre-decision* or *complete state variables* and *post-decision* or *incomplete state variables*. Considering that the new state variables $\boldsymbol{h}_t^+$ are measured before the re-

alization of vector $\boldsymbol{R}_{t+1}$, the information process becomes:

$$H_t = \left(\boldsymbol{h}_0, D_0^\pi(\boldsymbol{h}_0), \boldsymbol{h}_0^+, \boldsymbol{R}_1, \boldsymbol{h}_1, \ldots, \boldsymbol{h}_{t-1}, D_{t-1}^\pi(\boldsymbol{h}_{t-1}), \boldsymbol{h}_{t-1}^+, \boldsymbol{R}_t, \boldsymbol{h}_t\right), \tag{4.3}$$

where we can remove the old state variables $\boldsymbol{h}_t$ and get:

$$H_t = \left(\boldsymbol{h}_0, D_0^\pi(\boldsymbol{h}_0), \boldsymbol{h}_0^+, \boldsymbol{R}_1, D_1^\pi(\boldsymbol{h}_0^+), \boldsymbol{h}_1^+, \ldots, \boldsymbol{R}_{t-1}, D_{t-1}^\pi(\boldsymbol{h}_{t-2}^+), \boldsymbol{h}_{t-1}^+, \boldsymbol{R}_t\right) \tag{4.4}$$

Therefore, our new state variables are: $\boldsymbol{h}_0, \boldsymbol{h}_0^+, \boldsymbol{h}_1^+, \ldots, \boldsymbol{h}_{T-1}^+, \boldsymbol{h}_T$, where states $\boldsymbol{h}_0$ and $\boldsymbol{h}_T$ are still used since they are essential to the portfolio selection problem. Similarly to variables $\boldsymbol{h}_t$, with every state vector $\boldsymbol{h}_t^+$ we associate a value function, which we denote with V_t^+. Figure 4.1 shows the timing of events with both types of variables.

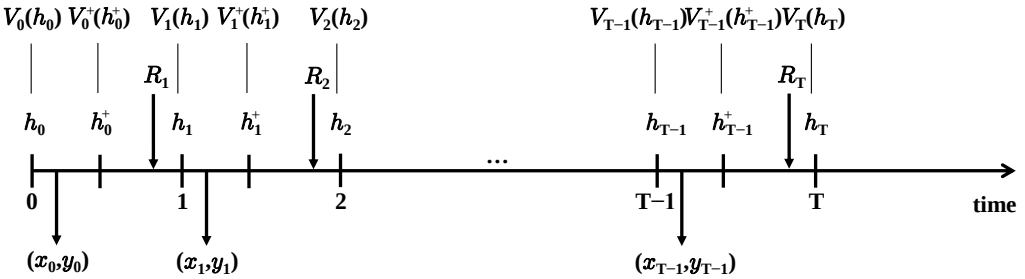


Figure 4.1: Timing of events with pre- and post-decision state variables

From (2.2) and using $h_{it} = R_{it} h_{i(t-1)}^+$ from (2.3), the relationships between the new state variables are given by:

$$\left.\begin{aligned}
&h_{i0}^+ = h_{i0} + x_{i0} - y_{i0}, \;\; i \in \mathcal{N} \\
&h_{00}^+ = h_{00} - (1+\theta)\sum_{i=1}^{N} x_{i0} + (1-\theta)\sum_{i=1}^{N} y_{i0} \\
&h_{it}^+ = R_{it} h_{i(t-1)}^+ + x_{it} - y_{it}, \;\; i \in \mathcal{N}, \; t = 1, \ldots, T-1 \\
&h_{0t}^+ = R_{0t} h_{0(t-1)}^+ - (1+\theta)\sum_{i=1}^{N} x_{it} + (1-\theta)\sum_{i=1}^{N} y_{it}, \; t = 1, \ldots, T-1 \\
&h_{iT} = R_{iT} h_{i(T-1)}^+, \;\; i \in \mathcal{N}
\end{aligned}\right\} \tag{4.5}$$

and are the new transition equations. For simplicity of notation, to describe the transitions in (4.5) we use functions $f_0 : \mathcal{H} \times \mathcal{A}_0 \mapsto \mathcal{H}$, $f : \mathcal{H} \times \mathcal{R} \times \mathcal{A}_t \mapsto \mathcal{H}$ and $f_T : \mathcal{H} \times \mathcal{R} \mapsto \mathcal{H}$ such that:

$$\left.\begin{aligned} \boldsymbol{h}_0^+ &= f_0\left(\boldsymbol{h}_0, \boldsymbol{x}_0, \boldsymbol{y}_0\right) \\ \boldsymbol{h}_t^+ &= f\left(\boldsymbol{h}_{t-1}^+, \boldsymbol{R}_t, \boldsymbol{x}_t, \boldsymbol{y}_t\right),\ t = 1, \ldots, T-1 \\ \boldsymbol{h}_T &= f_T\left(\boldsymbol{h}_{T-1}^+, \boldsymbol{R}_T\right) \end{aligned}\right\} \tag{4.6}$$

We are now ready to write the action space for the new state variables.

From (3.3)-(3.5), decisions $(\boldsymbol{x}_0, \boldsymbol{y}_0)$ must satisfy the following constraints:

$$-x_{i0} + y_{i0} \leq h_{i0},\ \ i \in \mathcal{N}, \tag{4.7}$$

$$(1+\theta)\sum_{i=1}^{N} x_{i0} - (1-\theta)\sum_{i=1}^{N} y_{i0} \leq h_{00}, \tag{4.8}$$

$$x_{i0}, y_{i0} \geq 0,\ \ i \in \mathcal{N}, \tag{4.9}$$

where (4.8) is the budget constraint, and action space $\mathcal{A}_0$ is given as before by:

$$\mathcal{A}_0 = \{(\boldsymbol{x}_0, \boldsymbol{y}_0) :\ (4.7) - (4.9) \text{ hold}\}, \tag{4.10}$$

Substituting $h_{it} = R_{it} h_{i(t-1)}^+$ in (3.3)-(3.5), decisions $(\boldsymbol{x}_t, \boldsymbol{y}_t)$ for $t = 1, 2, \ldots, T-1$ must satisfy the following constraints:

$$-x_{it} + y_{it} \leq R_{it} h_{i(t-1)}^+,\ \ i \in \mathcal{N}, \tag{4.11}$$

$$(1+\theta)\sum_{i=1}^{N} x_{it} - (1-\theta)\sum_{i=1}^{N} y_{it} \leq R_{0t} h_{0(t-1)}^+, \tag{4.12}$$

$$x_{it}, y_{it} \geq 0,\ \ i \in \mathcal{N}, \tag{4.13}$$

where (4.12) is the new budget constraint for $t = 1, 2, \ldots, T-1$ and the action space $\mathcal{A}_t$ is given by:

$$\mathcal{A}_t = \{(\boldsymbol{x}_t, \boldsymbol{y}_t) :\ (4.11) - (4.13) \text{ hold}\},\ t = 1, 2, \ldots, T-1 \tag{4.14}$$

What we are missing in order to complete the description of the dynamic programming formulation of the portfolio selection problem around the new state variables are the new optimality equations. As explained above, every new state vector is associated with a value function. To derive the new optimality equations, we begin with defining function $V_t^+(\boldsymbol{h}_t^+)$ as follows:

$$V_{t-1}^+(\boldsymbol{h}_{t-1}^+) = \mathbb{E}_{\boldsymbol{R}_t}\left\{V_t(\boldsymbol{R}_t \circ \boldsymbol{h}_{t-1}^+)|\boldsymbol{h}_{t-1}^+\right\},\ t = 1, 2, \ldots, T \tag{4.15}$$

where $\boldsymbol{R}_t \circ \boldsymbol{h}_{t-1}^+$ is the Hadamard product of vectors $\boldsymbol{R}_t$ and $\boldsymbol{h}_{t-1}^+$ (i.e. the i$^{\text{th}}$ element of vector $\boldsymbol{R}_t \circ \boldsymbol{h}_{t-1}^+$ is $R_{it}h_{i(t-1)}^+$).

Substituting $\boldsymbol{h}_t = \boldsymbol{R}_t \circ \boldsymbol{h}_{t-1}^+$ for $t = 1, 2, \ldots, T$ in the old optimality equations of (3.10), we have:

$$V_t\left(\boldsymbol{R}_t \circ \boldsymbol{h}_{t-1}^+\right) = \max_{(\boldsymbol{x}_t, \boldsymbol{y}_t) \in \mathcal{A}_t}\ \mathbb{E}_{\boldsymbol{R}_{t+1}}\left\{V_{t+1}\left(\boldsymbol{R}_{t+1} \circ \boldsymbol{h}_t^+\right) \mid \boldsymbol{R}_t \circ \boldsymbol{h}_{t-1}^+\right\} \tag{4.16}$$

If we substitute $R_{it}h_{i(t-1)}^+$ and $R_{0t}h_{0(t-1)}^+$ respectively with $R_{it}h_{i(t-1)}^+ + x_{it} - y_{it}$ and $R_{0t}h_{0(t-1)}^+ - (1+\theta)\sum_{i=1}^N x_{it} + (1-\theta)\sum_{i=1}^N y_{it}$ inside the max operator of (4.16), then $\boldsymbol{R}_t \circ \boldsymbol{h}_{t-1}^+$ can be replaced by $\boldsymbol{h}_t^+$ in the right-hand side of (4.16) which becomes:

$$V_t\left(\boldsymbol{R}_t \circ \boldsymbol{h}_{t-1}^+\right) = \max_{(\boldsymbol{x}_t, \boldsymbol{y}_t) \in \mathcal{A}_t}\ \mathbb{E}_{\boldsymbol{R}_{t+1}}\left\{V_{t+1}\left(\boldsymbol{R}_{t+1} \circ \boldsymbol{h}_t^+\right) \mid \boldsymbol{h}_t^+\right\}, \tag{4.17}$$

Substituting now the expectation in (4.17) with $V_t^+(\boldsymbol{h}_t^+)$ from (4.15), we have:

$$V_t\left(\boldsymbol{R}_t \circ \boldsymbol{h}_{t-1}^+\right) = \max_{(\boldsymbol{x}_t, \boldsymbol{y}_t) \in \mathcal{A}_t}\ V_t^+\left(\boldsymbol{h}_t^+\right) \tag{4.18}$$

If we take now expectation on both sides of (4.18) with respect to $\boldsymbol{R}_t$ and conditional on $\boldsymbol{h}_{t-1}^+$, (4.18) becomes:

$$\mathbb{E}_{\boldsymbol{R}_t}\left\{V_t\left(\boldsymbol{R}_t \circ \boldsymbol{h}_{t-1}^+\right) \mid \boldsymbol{h}_{t-1}^+\right\} = \mathbb{E}_{\boldsymbol{R}_t}\left\{\max_{(\boldsymbol{x}_t, \boldsymbol{y}_t) \in \mathcal{A}_t}\ V_t^+\left(\boldsymbol{h}_t^+\right) \mid \boldsymbol{h}_{t-1}^+\right\} \tag{4.19}$$

From (4.15), the left-hand side of (4.19) is $V_{t-1}^+\left(\boldsymbol{h}_{t-1}^+\right)$. Therefore, we have:

$$V_{t-1}^{+}\left(\boldsymbol{h}_{t-1}^{+}\right)=\mathbb{E}_{\boldsymbol{R}_t}\left\{\max_{(\boldsymbol{x}_t,\boldsymbol{y}_t)\in\mathcal{A}_t}\ V_t^{+}\left(\boldsymbol{h}_t^{+}\right)\mid\boldsymbol{h}_{t-1}^{+}\right\} \tag{4.20}$$

Recursive equation (4.20) describes the relationship between the value functions around the new state variables.

Note that from (4.15) we can compute the value function at state $\boldsymbol{h}_{T-1}^{+}$ using the value function at state $\boldsymbol{h}_T$, while from (4.18) we can compute the value function at state $\boldsymbol{h}_0$ using the value function at state $\boldsymbol{h}_0^{+}$.

Considering the above, the old optimality equations, which in chapter 3 were given by:

$$\left.\begin{aligned}V_t\left(\boldsymbol{h}_t\right)&=\max_{(\boldsymbol{x}_t,\boldsymbol{y}_t)\in\mathcal{A}_t}\ \mathbb{E}_{\mathbf{R}_{t+1}}\left\{V_{t+1}\left(\boldsymbol{h}_{t+1}\right)\mid\boldsymbol{h}_t\right\},\ t=0,1,\ldots,T-1\\V_T(\boldsymbol{h}_T)&=\sum_{i=0}^{N}h_{iT}\end{aligned}\right\}$$

are now replaced by:

$$\left.\begin{aligned}V_0\left(\boldsymbol{h}_0\right)&=\max_{(\boldsymbol{x}_0,\boldsymbol{y}_0)\in\mathcal{A}_0}\ V_0^{+}\left(\boldsymbol{h}_0^{+}\right)\\V_{t-1}^{+}\left(\boldsymbol{h}_{t-1}^{+}\right)&=\mathbb{E}_{\boldsymbol{R}_t}\left\{\max_{(\boldsymbol{x}_t,\boldsymbol{y}_t)\in\mathcal{A}_t}\ V_t^{+}\left(\boldsymbol{h}_t^{+}\right)\mid\boldsymbol{h}_{t-1}^{+}\right\},\ t=1,2,\ldots,T-1\\V_{T-1}^{+}(\boldsymbol{h}_{T-1}^{+})&=\mathbb{E}_{\boldsymbol{R}_T}\left\{V_T(\boldsymbol{h}_T)|\boldsymbol{h}_{T-1}^{+}\right\}\\V_T(\boldsymbol{h}_T)&=\sum_{i=0}^{N}h_{iT}\end{aligned}\right\} \tag{4.21}$$

The recursive equations of (4.21) are the *new optimality equations*. To simplify notation, from this point on we will use in the thesis $V_0^{-}\left(\boldsymbol{h}_0\right)$ instead of $V_0\left(\boldsymbol{h}_0\right)$ and $V_t\left(\boldsymbol{h}_t^{+}\right)$ instead of $V_t^{+}\left(\boldsymbol{h}_t^{+}\right)$. Thus, instead of (4.21), from this point on we will be referencing the following system of equations:

$$\left.\begin{aligned} V_0^- (\boldsymbol{h}_0) &= \max_{(\boldsymbol{x}_0,\boldsymbol{y}_0)\in\mathcal{A}_0} \; V_0\left(\boldsymbol{h}_0^+\right) \\ V_{t-1}\left(\boldsymbol{h}_{t-1}^+\right) &= \mathbb{E}_{\boldsymbol{R}_t}\left\{ \max_{(\boldsymbol{x}_t,\boldsymbol{y}_t)\in\mathcal{A}_t} \; V_t\left(\boldsymbol{h}_t^+\right) \mid \boldsymbol{h}_{t-1}^+ \right\},\; t = 1, 2, \ldots, T-1 \\ V_{T-1}(\boldsymbol{h}_{T-1}^+) &= \mathbb{E}_{\boldsymbol{R}_T}\left\{ V_T(\boldsymbol{h}_T) | \boldsymbol{h}_{T-1}^+ \right\} \\ V_T(\boldsymbol{h}_T) &= \sum_{i=0}^{N} h_{iT} \end{aligned}\right\} \tag{4.22}$$

Note that when we perform maximization in (4.22) we compute decisions $(\boldsymbol{x}_t, \boldsymbol{y}_t)$ using past information $\boldsymbol{R}_t$.

4.2 CVaR Estimation Given a Sample of Losses

We begin with reminding the reader of the polyhedral representation of CVaR from section 2.2. Recall that for decision vector X and for a sample of S realizations of random vector Y, where each realization Y_s is associated with a probability p_s and incurs a loss $f(X, Y_s)$ for $s \in \mathcal{S}$, where $\mathcal{S} = \{1, 2, \ldots, S\}$, CVaR is the optimal value of the following minimization problem:

$$\left.\begin{aligned} \min_{\substack{(X,g_0,g_2^s)\in \\ \mathcal{X}\times\mathbb{R}\times\mathbb{R}_+}} \quad & g_0 + \frac{1}{(1-\beta)} \sum_{s=1}^{S} p_s g_2^s \\ \text{s.t.} \quad & g_2^s \geq -g_0 + f(X, Y_s), \;\; s \in \mathcal{S} \\ & g_0 \;\text{ free}, g_2^s \geq 0, \;\; s \in \mathcal{S} \end{aligned}\right\} \tag{4.23}$$

Further, recall that the losses in the portfolio selection problem are given by $-\sum_{i=0}^{N} h_{iT} + \sum_{i=0}^{N} h_{i0}$. Considering now problem (4.23) in a multi-period setting and assuming that each scenario s is associated with fixed (given) losses $-\sum_{i=0}^{N} h_{iT}^s + \sum_{i=0}^{N} h_{i0}$ and equal probability $p_s = \frac{1}{S}$ for $s \in \mathcal{S}$, CVaR is the optimal value of the following minimization problem:

$$\left.\begin{array}{ll} \underset{\substack{((\boldsymbol{x},\boldsymbol{y}),g_0,g_2^s)\\ \in\mathcal{A}\times\mathbb{R}\times\mathbb{R}_+}}{\min} & g_0+\dfrac{1}{S(1-\beta)}\displaystyle\sum_{s=1}^{S}g_2^s \\ \text{s.t.} & g_2^s+g_0\geq-\displaystyle\sum_{i=0}^{N}h_{iT}^s\left(\boldsymbol{x},\boldsymbol{y},\mathcal{R}\right)+\sum_{i=0}^{N}h_{i0},\ \ s\in\mathcal{S} \\ & g_0\ \text{ free}, g_2^s\geq 0,\ \ s\in\mathcal{S} \end{array}\right\} \tag{4.24}$$

where $h_{iT}^s\left(\boldsymbol{x},\boldsymbol{y},\mathcal{R}\right)$ represents the amount of holdings in asset i at time T under scenario s and is a function of the buying decisions $\boldsymbol{x}$, the selling decisions $\boldsymbol{y}$ and the outcome space $\mathcal{R}$ over the entire horizon.

In the discussion that follows, we solve problem (4.24) by looking at its dual and derive an analytic formula for computing CVaR given that terminal wealth $\sum_{i=0}^{N}h_{iT}^s$ is fixed and known for all $s\in\mathcal{S}$.

If we assign dual variables u_s with $s\in\mathcal{S}$ to the constraints of optimization problem (4.24), then its dual is given by:

$$\left.\begin{array}{ll} \underset{u_s}{\max} & \displaystyle\sum_{s=1}^{S}\big[-\sum_{i=0}^{N}h_{iT}^s+\sum_{i=0}^{N}h_{i0}\big]u_s \\ \text{s.t.} & u_s\leq\dfrac{1}{S(1-\beta)},\ \ s\in\mathcal{S} \\ & \displaystyle\sum_{s=1}^{S}u_s=1 \\ & u_s\geq 0,\ \ s\in\mathcal{S} \end{array}\right\} \tag{4.25}$$

Before we proceed with the solution to problem (4.25), we first define function *ceil*. If A is a matrix with elements in $\mathbb{R}$, then function ceil(A) rounds up every element of A to the nearest integer greater than or equal to it.

We are now ready to derive the solution to problem (4.25).

We begin by letting $f_s=-\sum_{i=0}^{N}h_{iT}^s+\sum_{i=0}^{N}h_{i0}$ in problem (4.25) which becomes:

$$\left.\begin{aligned} \max_{u_s} \quad & \sum_{s=1}^{S} f_s u_s \\ \text{s.t.} \quad & u_s \leq \frac{1}{S(1-\beta)}, \;\; s \in \mathcal{S} \\ & \sum_{s=1}^{S} u_s = 1 \\ & u_s \geq 0, \;\; s \in \mathcal{S} \end{aligned}\right\} \tag{4.26}$$

Problem (4.26) can be viewed as a continuous knapsack problem with upper bounds on the variables (see [46] and references therein) and can be solved greedily as follows: First, we sort losses f_s in a decreasing order. Then, starting from the scenario that incurs the highest losses, we increase the level of variable u_s until upper bound $\frac{1}{S(1-\beta)}$ is reached. After that, we move on to the second scenario in the order and work in a similar manner. The allocation process ends at some scenario l, where for the first time the following condition holds:

$$\sum_{s=1}^{l} \frac{p_s}{1-\beta} \geq 1 \Leftrightarrow \sum_{s=1}^{l} p_s \geq 1-\beta$$

Note that increasing variables u_s in a different order from the one used above would be suboptimal.

Considering the above, it is straightforward to verify that the optimal solution of problem (4.26) is as follows:

$$\left.\begin{aligned} u_1^* &= \frac{1}{S(1-\beta)}, \\ u_2^* &= \frac{1}{S(1-\beta)}, \\ &\vdots \\ u_{l-1}^* &= \frac{1}{S(1-\beta)}, \\ u_l^* &= 1 - \frac{l-1}{S(1-\beta)}, \\ u_s^* &= 0, \text{ for } s > l \end{aligned}\right\}$$

The optimal value of problem (4.26) is CVaR and is given by:

$$\text{CVaR} = \sum_{s=1}^{l} f_s u_s = \sum_{s=1}^{l-1} \frac{f_s}{S(1-\beta)} + f_l - \frac{(l-1)f_l}{S(1-\beta)} = f_l + \frac{1}{S(1-\beta)} \sum_{s=1}^{l-1} (f_s - f_l), \tag{4.27}$$

where f_l gives us VaR.

We will now provide a numerical example to illustrate how we can compute VaR and CVaR using (4.27).

Example 4.1. Suppose we have 20 scenarios, each with probability $\frac{1}{20}$, and losses (in £) as shown in Table 4.1.

s	loss	s	loss	s	loss	s	loss
1	69912.77	6	72630.89	11	155215.83	16	71914.18
2	43558.97	7	118916.57	12	159649.55	17	147567.85
3	111911.30	8	195738.76	13	122867.99	18	150631.95
4	91463.27	9	169753.02	14	154131.76	19	177367.21
5	77645.87	10	190190.81	15	129466.14	20	135389.64

Table 4.1: Losses per scenario s

First, we sort scenarios in a decreasing order in terms of losses as in Table 4.2. Then, assuming that $\beta = 0.85$, we use the first $l = \text{ceil}(20 * 0.15) = 3$ scenarios to compute VaR and CVaR. Specifically, VaR is given by the losses incurred by the third scenario, i.e. VaR=$£177367.21$ and CVaR is equal to VaR plus the average deviations of the losses incurred by the first two scenarios from the third one, i.e. CVaR=$177367.21 + \frac{(195738.76-177367.21)+(190190.81-177367.21)}{3} = £187765.59$.

s	loss	s	loss	s	loss	s	loss
8	195738.76	11	155215.83	15	129466.14	5	77645.87
10	190190.81	14	154131.76	13	122867.99	6	72630.89
19	177367.21	18	150631.95	7	118916.57	16	71914.18
9	169753.02	17	147567.85	3	111911.30	1	69912.77
12	159649.55	20	135389.64	4	91463.27	2	43558.97

Table 4.2: Sorted losses

4.3 General ADP Algorithm

In this section, we construct an iterative approximate dynamic programming algorithm to solve the multi-period portfolio selection problem.

As we will see later on, in each iteration of the algorithm, we sample one scenario path of returns and we do a forward pass through time on it, where each time

a return vector is revealed we compute new actions and transit to new states until we reach the end of the horizon. In this manner, we transit to state $\boldsymbol{h}_T^s$ which gives us losses $-\sum_{i=0}^{N} h_{iT}^s + \sum_{i=0}^{N} h_{i0}$. Then, we store the losses of the current scenario and we use them as an input for the next iterations. Using these losses along with the losses of the previous iterations we compute CVaR at the current iteration using formula (4.27).

Further, we avoid evaluating the expectations in the optimality equations of (4.22) by using Monte-Carlo sampling as follows: We assume that we have S scenario realizations of returns $\omega^s = (\boldsymbol{R}_1^s, \boldsymbol{R}_2^s, ..., \boldsymbol{R}_T^s)$, where $s = 1, 2, \ldots, S$ is the iteration counter. For each realization we perform one iteration of the algorithm, which takes us forward through time.

We begin our discussion by introducing functional approximations for the value functions in the sense that value at each state is estimated by a closed form function.

In line with the notation of [8], [18] and [19], for $t = 0, 1, \ldots, T-1$ a generic structure for an approximation of $V_t(\boldsymbol{h}_t^+)$ is the following:

$$\hat{V}_t(\boldsymbol{h}_t^+) = \sum_{j \in \mathcal{P}} \lambda_{jt} \phi_{jt} \left(\boldsymbol{h}_t^+\right), \tag{4.28}$$

where λ_{jt} are parameters and $\phi_{jt}\left(\boldsymbol{h}_t^+\right)$ are fixed *basis functions* that capture important characteristics of state vector $\boldsymbol{h}_t^+$.

In this thesis, we consider separable functional approximations for the unknown value functions of the assets and express them as follows:

$$\hat{V}_t\left(\boldsymbol{h}_t^+\right) = \sum_{i=0}^{N} \hat{V}_{it}\left(h_{it}^+\right) \tag{4.29}$$

In particular, in chapter 6 we replace every $\hat{V}_{it}\left(h_{it}^+\right)$ with a linear increasing function and thus for each asset we estimate one slope, and in chapter 7 we replace every $\hat{V}_{it}\left(h_{it}^+\right)$ with a piecewise linear concave function and thus for each asset we estimate a small number of slopes. Note that (4.29) is simply a different representation of (4.28). To solve the optimization subproblems of the ADP methods in chapters 6 and 7 we will need the following lemma which simply tells us that due to transaction costs we should never consider simultaneously selling and buying the same asset.

Lemma 4.3.1. *Let $\hat{V}_t\left(\boldsymbol{h}_t^+\right)$ be a sum of separable increasing functions of type (4.29). If our objective is to maximize function $\hat{V}_t\left(\boldsymbol{h}_t^+\right)$, then it is suboptimal to simultaneously sell and buy the same asset.*

Proof Suppose we sell δ units of risky asset j, where $\delta > 0$, and using the resulting cash we buy back the same asset. Due to the budget constraint, which is given by:

$$\sum_{i=1}^{N} x_{it} \leq \frac{1}{1+\theta} R_{0t} h^{+}_{0(t-1)} + \frac{1-\theta}{1+\theta} \sum_{i=1}^{N} y_{it},$$

every incremental increase of a y_{it} increases one of the x_{it}'s by $\frac{1-\theta}{1+\theta}$ units. Therefore, if we sell δ units of risky asset j, then due to transaction costs, with the resulting wealth we can buy back $\frac{1-\theta}{1+\theta}\delta$ units of the same asset.

Considering the above, if we substitute y_{jt} and x_{jt} with $y_{jt} + \delta$ and $x_{jt} + \frac{1-\theta}{1+\theta}\delta$ respectively, then the value function of cash in (4.29) becomes:

$$\hat{V}_{0t}\left(R_{0t} h^{+}_{0(t-1)} - (1+\theta)\left(\sum_{i=1}^{N} x_{it} + \frac{1-\theta}{1+\theta}\delta\right) + (1-\theta)\left(\sum_{i=1}^{N} y_{it} + \delta\right)\right) =$$
$$\hat{V}_{0t}\left(R_{0t} h^{+}_{0(t-1)} - (1+\theta)\sum_{i=1}^{N} x_{it} + (1-\theta)\sum_{i=1}^{N} y_{it}\right),$$

i.e. it remains the same, while the value function of asset j changes to:

$$\hat{V}_{jt}\left(R_{jt} h^{+}_{j(t-1)} + x_{jt} + \frac{1-\theta}{1+\theta}\delta - y_{jt} - \delta\right) =$$
$$\hat{V}_{jt}\left(R_{jt} h^{+}_{j(t-1)} + x_{jt} - y_{jt} - \frac{2\theta\delta}{1+\theta}\right)$$

Given that function $\hat{V}_{jt}\left(h^{+}_{jt}\right)$ is increasing, the objective function becomes smaller. ■

We are now ready to describe one iteration of our approximate dynamic programming algorithm which is summarized by Algorithm 4.1.

Suppose that at iteration $s-1$ we estimated value function approximations $\hat{V}_t^{s-1}$ for $t = 0, 1, \ldots, T$ of optimal value functions $V_0, V_1, \ldots, V_T$ (note that we do not estimate V_0^- and the reason for this becomes apparent later). Using these estimates along with the current realization $\omega^s = (\boldsymbol{R}_1^s, \boldsymbol{R}_2^s, ..., \boldsymbol{R}_T^s)$, we move forward in time at iteration s computing decisions and states in the order in which they appear in history (4.4), which at iteration s and from time 0 up to time T is given by the following sequence:

$$\left(\boldsymbol{h}_0, (\boldsymbol{x}_0^s, \boldsymbol{y}_0^s), \boldsymbol{h}_0^{s,+}, \boldsymbol{R}_1^s, (\boldsymbol{x}_1^s, \boldsymbol{y}_1^s), \boldsymbol{h}_1^{s,+}, \ldots, \boldsymbol{h}_{T-1}^{s,+}, \boldsymbol{R}_T^s, \boldsymbol{h}_T^s\right)$$

Starting with known holdings $\boldsymbol{h}_0$, we compute decisions $(\boldsymbol{x}_0^s, \boldsymbol{y}_0^s)$ by solving:

$$\tilde{V}_0^{s,-}(\boldsymbol{h}_0) = \max_{(\boldsymbol{x}_0^s, \boldsymbol{y}_0^s)} \hat{V}_0^{s-1}(f_0(\boldsymbol{h}_0, \boldsymbol{x}_0^s, \boldsymbol{y}_0^s)) \tag{4.30}$$

Then, after taking decisions $(\boldsymbol{x}_0^s, \boldsymbol{y}_0^s)$, we transit to state $\boldsymbol{h}_0^{s,+}$ using:

$$\boldsymbol{h}_0^{s,+} = f_0(\boldsymbol{h}_0, \boldsymbol{x}_0^s, \boldsymbol{y}_0^s) \tag{4.31}$$

For $t = 1, 2, \ldots, T-1$ let $\boldsymbol{h}_{t-1}^{s,+}$ be the current state at time $t-1$. Then, after we observe returns $\boldsymbol{R}_t^s$, we compute decisions $(\boldsymbol{x}_t^s, \boldsymbol{y}_t^s)$ by solving:

$$\tilde{V}_{t-1}^s\left(\boldsymbol{h}_{t-1}^{s,+}\right) = \max_{(\boldsymbol{x}_t^s, \boldsymbol{y}_t^s)} \hat{V}_t^{s-1}\left(f\left(\boldsymbol{h}_{t-1}^{s,+}, \boldsymbol{R}_t^s, \boldsymbol{x}_t^s, \boldsymbol{y}_t^s\right)\right), \tag{4.32}$$

and after taking decisions $(\boldsymbol{x}_t^s, \boldsymbol{y}_t^s)$ we transit to state $\boldsymbol{h}_t^{s,+}$ using:

$$\boldsymbol{h}_t^{s,+} = f\left(\boldsymbol{h}_{t-1}^{s,+}, \boldsymbol{R}_t^s, \boldsymbol{x}_t^s, \boldsymbol{y}_t^s\right) \tag{4.33}$$

Continuing in the above manner, at the end of the time horizon we will have computed state $\boldsymbol{h}_{T-1}^{s,+}$ and, after we observe returns $\boldsymbol{R}_T^s$, we transit to state $\boldsymbol{h}_T^s$ using:

$$\boldsymbol{h}_T^s = f_T\left(\boldsymbol{h}_{T-1}^{s,+}, \boldsymbol{R}_T^s\right), \tag{4.34}$$

Using a risk importance parameter $\gamma \in [0, 1]$, we define the terminal value function $\hat{V}_T^s(\cdot)$ as follows:

$$\hat{V}_T^s(\boldsymbol{h}_T^s) = \gamma \sum_{i=0}^{N} h_{iT}^s - (1-\gamma)\hat{\text{CVaR}}^s\left(\boldsymbol{h}_T^1, \ldots, \boldsymbol{h}_T^s\right), \tag{4.35}$$

which resembles the mean-risk objective function (2.18) in the sense that they are both weighted averages of a terminal wealth term and a CVaR term with weights respectively γ and $(1-\gamma)$.

To estimate $\hat{\text{CVaR}}^s$ in (4.35), we use the terminal holdings of all previous iterations $\boldsymbol{h}_T^1, \ldots, \boldsymbol{h}_T^s$. We use the losses incurred by the current scenario, which are computed by $-\sum_{i=0}^N h_{iT}^s + \sum_{i=0}^N h_{i0}$, together with the losses incurred by the previous iterations (note that these are taken as fixed and do not depend on the policy of the current iteration) and compute $\hat{\text{CVaR}}^s$ as the average of the $100(1-\beta)\%$ worst losses using formula (4.27). We store the losses of the current iteration in order to use them in a similar manner in the iterations to follow.

Further, we let:

$$\tilde{V}_{T-1}^{s}\left(\boldsymbol{h}_{T-1}^{s,+}\right)=\hat{V}_{T}^{s}\left(f_{T}\left(\boldsymbol{h}_{T}^{s,+},\boldsymbol{R}_{T}^{s}\right)\right), \tag{4.36}$$

Note that (4.30), (4.32), (4.35) and (4.36) resemble the optimality equations of (4.22) without a risk measure.

Finally, at each time t we use *gradient information* and update function $\hat{V}_t^s$ using the following update rule:

$$\hat{V}_{t}^{s}=U\left(\hat{V}_{t}^{s-1},\Delta\tilde{V}_{it}^{s}\right),\ t=0,1,\ldots,T-1, \tag{4.37}$$

where $U(\cdot)$ is some update function that depends on the value function approximations of the current iteration (for smoothing) as well as on gradient information. Note that function $U(\cdot)$ is specific to the type of approximation we use and in this thesis we define two update functions, one for linear approximations (see chapter 6) and one for a piecewise linear approximation (see chapter 7).

In Algorithm 4.1, we input holdings $\boldsymbol{h}_0$ and realizations $\omega^s=(\boldsymbol{R}_1^s,\boldsymbol{R}_2^s,...,\boldsymbol{R}_T^s)$ for $s=1,2,\ldots,S$ and in the last iteration we obtain the terminal estimates of the approximate value functions $\hat{V}_t^S$ for $t=0,1,\ldots,T-1$.

Algorithm 4.1 General ADP Algorithm

Input: $\boldsymbol{h}_0$, $\omega^s=(\boldsymbol{R}_1^s,\boldsymbol{R}_2^s,...,\boldsymbol{R}_T^s)$ for $s=1,2,\ldots,S$

Step 0. Set $s:=1$ and initialize $\hat{V}_t^0(\boldsymbol{h}_t^+)$ for $t=0,\ldots,T-1$.

Step 1. Choose a sample realization $\omega^s\in\Omega$ and set $t:=0$.

Step 2. Compute decisions and next state:

Step 2a. If $t=0$ compute $(\boldsymbol{x}_0^s,\boldsymbol{y}_0^s)$ by solving:

$$\tilde{V}_0^{s,-}\left(\boldsymbol{h}_0\right)=\max_{\left(\boldsymbol{x}_0^s,\boldsymbol{y}_0^s\right)}\ \hat{V}_0^{s-1}\left(f_0\left(\boldsymbol{h}_0,\boldsymbol{x}_0^s,\boldsymbol{y}_0^s\right)\right),$$

and transit to $\boldsymbol{h}_0^+$ by setting:

$$\boldsymbol{h}_0^{s,+}:=f_0\left(\boldsymbol{h}_0,\boldsymbol{x}_0^s,\boldsymbol{y}_0^s\right)$$

Step 2b. If $0<t\leq T-1$ compute $(\boldsymbol{x}_t^s,\boldsymbol{y}_t^s)$ by solving:

$$\tilde{V}_{t-1}^{s}\left(\boldsymbol{h}_{t-1}^{s,+}\right)=\max_{(\boldsymbol{x}_t^s,\boldsymbol{y}_t^s)}\ \hat{V}_t^{s-1}\left(f\left(\boldsymbol{h}_{t-1}^{s,+},\boldsymbol{R}_t^s,\boldsymbol{x}_t^s,\boldsymbol{y}_t^s\right)\right)\ \ (\star 1)\ ,$$

In Algorithm 4.1, for each realization $\omega^s=(\boldsymbol{R}_1^s,\boldsymbol{R}_2^s,...,\boldsymbol{R}_T^s)$, where $s=$

and transit to $\boldsymbol{h}_t^+$ by setting:

$$\boldsymbol{h}_t^{s,+} := f\left(\boldsymbol{h}_{t-1}^{s,+}, \boldsymbol{R}_t^s, \boldsymbol{x}_t^s, \boldsymbol{y}_t^s\right)$$

Step 2c. If $t = T$ transit to state $\boldsymbol{h}_T^s$ by setting:

$$\boldsymbol{h}_T^s := f_T\left(\boldsymbol{h}_{T-1}^{s,+}, \boldsymbol{R}_T^s\right),$$

compute $\hat{\text{CVaR}}^s$ using (4.27) and set:

$$\hat{V}_T^s\left(\boldsymbol{h}_T^s\right) := \gamma \sum_{i=0}^N h_{iT}^s - (1-\gamma)\hat{\text{CVaR}}^s\left(\boldsymbol{h}_T^1, \ldots, \boldsymbol{h}_T^s\right)$$

$$\tilde{V}_{T-1}^s\left(\boldsymbol{h}_{T-1}^{s,+}\right) := \hat{V}_T^s\left(f_T\left(\boldsymbol{h}_{T-1}^{s,+}, \boldsymbol{R}_T^s\right)\right) \quad (\star 2)$$

Step 3. If $0 \leq t \leq T-1$, then compute gradient information $\Delta\tilde{V}_{it}^s$ using $(\star 1)$, $(\star 2)$ and update $\hat{V}_t$ by setting:

$$\hat{V}_t^s := U\left(\hat{V}_t^{s-1}, \Delta\tilde{V}_{it}^s\right)$$

Step 4. If $t < T$ set $t := t+1$ and go to Step 2. If $t = T$ and $s < S$ set $s := s+1$ and go to Step 1. Else stop.

Output: $\hat{V}_t^S \; \forall t$

$1, 2, \ldots, S$, we do a forward pass in time solving T maximization problems. In total, we need to solve $S \times T$ maximization problems and in each maximization problem we have a total of $2N$ decision variables, N for buying and another N for selling. As for the number of constraints in each maximization problem, this depends on the functional approximation and any other additional assumptions made (see how feasible region changes in chapters 6 and 7), but in general this is a polynomial function of N. Finally, regarding the complexity that comes from computing $\hat{\text{CVaR}}$, at each iteration s we have a time complexity of $O(s \log s)$ which comes from sorting the losses of all previous iterations $1, 2, \ldots, s$.

Note that from this point on in the thesis we will call the maximization subproblems in (4.30) and (4.32) that we solve in order to compute decisions $(\boldsymbol{x}_t^s, \boldsymbol{y}_t^s)$ for all $t = 0, 1, \ldots, T-1$ the *subproblems of ADP*.

4.4 Gradient Information

In Step 3 of Algorithm 4.1, we update $\hat{V}_t^s$ using function $U(\cdot)$, which takes as arguments value function estimates $\hat{V}_t^{s-1}$ and gradients $\Delta\tilde{V}_{it}^s$. In fact, for $t = 0, 1, \ldots, T-2$ gradients $\Delta\tilde{V}_{it}^s$ are estimates at a particular state $\boldsymbol{h}_t^{s,+}$ and we use:

$$\Delta\tilde{V}_{it}^s\left(\boldsymbol{h}_t^{s,+}\right) = \tilde{V}_t^s\left(\boldsymbol{h}_t^{s,+} + \boldsymbol{e}_i\right) - \tilde{V}_t^s\left(\boldsymbol{h}_t^{s,+}\right),\ i = 0, 1, \ldots, N, \tag{4.38}$$

where $\boldsymbol{e}_i$ is the $1 \times (N+1)$ unit vector with one at position i.

At time $T-1$ we use a different gradient for $\tilde{V}_{T-1}^s$. The reason for this is because $\tilde{V}_{T-1}^s$ is a function of $\hat{\text{CVaR}}^s$ which uses not only the terminal holdings of the current iteration but also the terminal holdings of the previous iterations.

If we substitute (4.35) in (4.36) we get:

$$\tilde{V}_{T-1}^s\left(\boldsymbol{h}_{T-1}^{s,+}\right) = \gamma\sum_{i=0}^{N} R_{iT}^s h_{i(T-1)}^{s,+} - (1-\gamma)\hat{\text{CVaR}}^s\left(\boldsymbol{R}_T^1 \circ \boldsymbol{h}_{T-1}^{1,+}, \ldots, \boldsymbol{R}_T^s \circ \boldsymbol{h}_{T-1}^{s,+}\right), \tag{4.39}$$

where we use $\boldsymbol{h}_T = \boldsymbol{R}_T \circ \boldsymbol{h}_{T-1}^+$ (this is the Hadamard product of vectors $\boldsymbol{R}_T$ and $\boldsymbol{h}_{T-1}^+$).

$\tilde{V}_{T-1}^s$ has two components and we define a gradient for each component. For the first component, which is the terminal wealth, increasing $h_{i(T-1)}^{s,+}$ by one unit will give us an increase of γR_{iT}^s. For the second component, we define the following gradient:

$$\begin{aligned}
&\Delta\hat{\text{CVaR}}_i^s\left(\boldsymbol{R}_T^1 \circ \boldsymbol{h}_{T-1}^{1,+}, \ldots, \boldsymbol{R}_T^s \circ \boldsymbol{h}_{T-1}^{s,+}\right) = \\
&\hat{\text{CVaR}}_i^s\left(\boldsymbol{R}_T^1 \circ \left(\boldsymbol{h}_{T-1}^{1,+} + \boldsymbol{e}_i\right), \ldots, \boldsymbol{R}_T^s \circ \left(\boldsymbol{h}_{T-1}^{s,+} + \boldsymbol{e}_i\right)\right) \\
&- \hat{\text{CVaR}}_i^s\left(\boldsymbol{R}_T^1 \circ \boldsymbol{h}_{T-1}^{1,+}, \ldots, \boldsymbol{R}_T^s \circ \boldsymbol{h}_{T-1}^{s,+}\right),
\end{aligned} \tag{4.40}$$

where $\Delta\hat{\text{CVaR}}_i^s$ is computed by subtracting the current estimate $\hat{\text{CVaR}}^s$ from what this estimate would have been if we increased holdings $h_{i(T-1)}^+$ of the previous iterations and the current one by the respective scenario return R_{iT}.

Therefore, the gradient of $\tilde{V}_{T-1}^s$ is given by:

$$\Delta\tilde{V}_{i(T-1)}^s\left(\boldsymbol{h}_{T-1}^{s,+}\right) = \gamma R_{iT}^s - (1-\gamma)\Delta\hat{\text{CVaR}}_i^s\left(\boldsymbol{R}_T^1 \circ \boldsymbol{h}_{T-1}^{1,+}, \ldots, \boldsymbol{R}_T^s \circ \boldsymbol{h}_{T-1}^{s,+}\right), \tag{4.41}$$

Note that computing $\Delta\hat{\text{CVaR}}_i^s$ in the above manner implies that $\Delta\hat{\text{CVaR}}_i^s < 0$, which makes $\Delta\tilde{V}^s_{i(T-1)} > 0$, since smaller scenario losses mean smaller $100(1-\beta)\%$ worst losses and as a result a smaller estimate $\hat{\text{CVaR}}^s$. This will be used later on in chapters 6 and 7 to justify why the slopes in our approximate value functions are positive everywhere, in all time periods of all iterations.

Part III

Portfolio Selection Methods used as Benchmarks

Chapter 5

Stochastic Programming and the Equally-weighted Portfolio

In this chapter, we briefly describe the portfolio selection methods which are used as *benchmarks* to compare with the approximate dynamic programming methods.

This chapter is structured as follows: First, in section 5.1 we model the portfolio selection problem with CVaR as a measure of risk and proportional transaction costs as a *Stochastic Program*. In particular, in subsection 5.1.1 we formulate the *myopic* single-period portfolio selection problem as a two-stage stochastic program and in subsection 5.1.2 we formulate the *multi-period* portfolio selection problem as a multistage stochastic program. Finally, in section 5.2 we consider the naive *equally-weighted* portfolio model which due to transaction costs is expressed as a linear program.

5.1 Stochastic Programming

Stochastic programming provides a modeling framework for optimization problems that involve uncertainty. The *key idea* is that decisions are based on information available at the time decisions are taken and not on future observations and the *main assumption* is that the probability distributions of the random parameters are known.

Figure (5.1) shows the sequence of events in a stochastic program. Here, we need to construct decision rules in the beginning of the planning horizon for all the stages at which decisions need to be taken. Assuming a planning horizon with T stages, the first decision rule refers to a first stage where we take decisions with

respect to some *deterministic constraints*. The first-stage decisions are taken before the realization of any random event and are known as *here-and-now* or *anticipative* decisions. Later on, after the realization of a random event, second-stage decisions are taken. Then, after the realization of another random event, third-stage decisions are made, and so on, until final stage T is reached. The decisions taken in the second-stage and thereafter depend on some previous realization of a random event and are known as *wait-and-see*, *adaptive*, *non-anticipative* or *recourse* decisions. A decision process such as the one described above, where we cannot anticipate the future, i.e. decisions made at each stage are not dependent either on future arrival of stochastic information or on future decisions, is called *non-anticipative*.

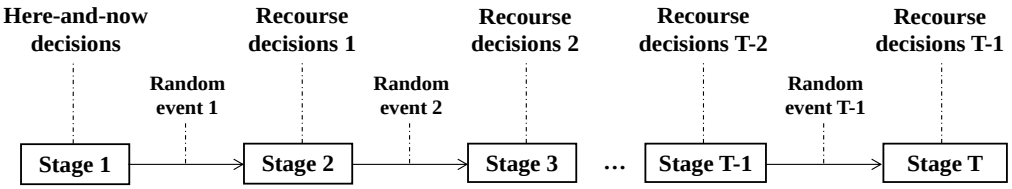


Figure 5.1: Sequence of events in a multistage stochastic program

In the above modeling context, if $T = 2$ then we have a *two-stage stochastic program* and if $T > 2$ then we have a *multistage stochastic program*. Further, if the objective and the constraints of the stochastic program are linear in the decisions, then the stochastic program belongs to the general class of *Stochastic Linear Programs (SLP)*. For a general discussion about stochastic linear programs, we refer the reader to [44] and [72].

In this section, we use CVaR as a measure of risk and assuming proportional transaction costs we first formulate the single-period portfolio selection problem as a two-stage stochastic program. For similar single-period portfolio optimization models, we refer the reader to [38] and [48]. Then, assuming that we have a planning horizon of T periods, we write the corresponding stochastic programming formulation of the multi-period portfolio selection problem. For similar multistage stochastic programming formulations of the portfolio selection problem, we refer the reader to [16], [79] and [86].

5.1.1 The Single-period Portfolio as a Two-stage Stochastic Program

In this section, we formulate the single-period portfolio problem with CVaR as a measure of risk and proportional transaction costs as a two-stage stochastic linear program. Specifically, we write the two-stage stochastic programming formulation of the single-period portfolio selection problem and to solve it we generate a discrete approximation of the stochastic process of random returns.

Two-stage Stochastic Programming Formulation

Assuming a time horizon with two discrete points in time (i.e. one time period), we start with known holdings $\boldsymbol{h}_0$ and in a first-stage we decide how much to buy and sell from every risky asset. The first-stage decisions are denoted by vector $X_0 = \left(\boldsymbol{h}_0^+, \boldsymbol{x}_0, \boldsymbol{y}_0\right)$ and from (2.2) they must satisfy the following constraints:

$$h_{i0}^+ = h_{i0} + x_{i0} - y_{i0}, \;\; i \in \mathcal{N}, \tag{5.1}$$

$$h_{00}^+ = h_{00} - (1+\theta)\sum_{i=1}^{N} x_{i0} + (1-\theta)\sum_{i=1}^{N} y_{i0}, \tag{5.2}$$

$$h_{i0}^+ \geq 0, \;\; i \in \mathcal{N} \cup \{0\}, \tag{5.3}$$

$$x_{i0}, y_{i0} \geq 0, \;\; i \in \mathcal{N} \tag{5.4}$$

Constraints (5.1)-(5.4) define action space $\mathcal{X}_0$, i.e. we have:

$$\mathcal{X}_0 = \left\{ \left(\boldsymbol{h}_0^+, \boldsymbol{x}_0, \boldsymbol{y}_0\right) : \; (5.1) - (5.4) \text{ hold} \right\} \tag{5.5}$$

Note that the constraints in set (5.5) are deterministic since they do not require that we have a priori knowledge of the stochastic process of random returns. Then, after the realization of the random return vector $\boldsymbol{R}_1 = (R_{11}, R_{21}, \ldots, R_{N1})$, we measure terminal wealth, which from (2.4) is given by $v^1 = \sum_{i=0}^{N} R_{i1} h_{i0}^+$, and CVaR, which by definition (see section 2.2) is the optimal value of the following minimization problem:

$$\left.\begin{aligned} \min_{\substack{((\boldsymbol{x}_0,\boldsymbol{y}_0),g_0,g_2)\\ \in\mathcal{X}_0\times\mathbb{R}\times\mathbb{R}_+}} \quad & g_0+\frac{1}{1-\beta}\mathbb{E}\left\{g_2\right\} \\ \text{s.t.} \quad & g_2\geq -g_0-\sum_{i=0}^{N}R_{i1}\left(h_{i0}+x_{i0}-y_{i0}\right)+\sum_{i=0}^{N}h_{i0}, \quad \textbf{a.s.} \\ & g_0 \text{ free}, g_2\geq 0 \end{aligned}\right\} \tag{5.6}$$

In problem (5.6), *a.s.* is an abbreviation for *almost surely*, is used for constraints that have random variables and means that the constraints with which it is associated are satisfied with probability one. Note that in the above problem variables g_0, g_2, as well as terminal wealth $\sum_{i=0}^{N}R_{i1}h_{i0}^{+}$ need the realization of the random variable $\boldsymbol{R}_1$ and are determined in a second stage.

Figure 5.2 shows the timing of events and the two stages in the single-period portfolio problem.

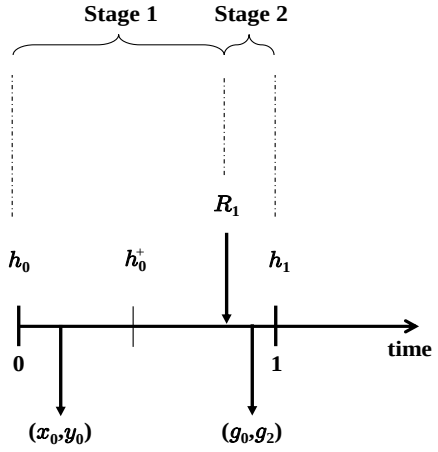


Figure 5.2: Timing of events and stages in the single-period portfolio problem

As for the objective, we consider a mean-risk objective of type (2.18), where we use CVaR as a measure of risk and according to which we aim at co-optimizing the following weighted average of expected terminal wealth and CVaR:

$$\max \gamma\mathbb{E}_{\boldsymbol{R}_1}\left\{\sum_{i=0}^{N}R_{i1}h_{i0}^{+}\right\}-(1-\gamma)\text{CVaR} \tag{5.7}$$

where γ is the risk importance parameter and CVaR is given by (5.6) and contains an expectation on losses $-\sum_{i=0}^{N}R_{i1}h_{i0}^{+}\left(x_{i0},y_{i0}\right)+\sum_{i=0}^{N}h_{i0}$.

Combining action space (5.5) with CVaR problem (5.6) and objective (5.7), and assuming a probability space $(\Omega, \mathcal{F}, \mathcal{P})$ for stochastic process $\mathbf{R}_1$, we can write the two-stage stochastic programming formulation of the single-period portfolio problem as follows:

$$\left.\begin{aligned} \max_{(\boldsymbol{h}_0^+, \boldsymbol{x}_0, \boldsymbol{y}_0, g_0, g_2)} \quad & \gamma \mathbb{E}_{\boldsymbol{R}_1}\left[\sum_{i=0}^{N} R_{i1} h_{i0}^+\right] - (1-\gamma) g_0 - \frac{1-\gamma}{1-\beta}\mathbb{E}_{\boldsymbol{R}_1}\left[g_2\right] \\ \text{s.t.} \quad & h_{i0}^+ = h_{i0} + x_{i0} - y_{i0}, \;\; i \in \mathcal{N} \\ & h_{00}^+ = h_{00} - (1+\theta)\sum_{i=1}^{N} x_{i0} + (1-\theta)\sum_{i=1}^{N} y_{i0} \\ & g_2 \geq -g_0 - \sum_{i=0}^{N} R_{i1} h_{i0}^+ + \sum_{i=0}^{N} h_{i0}, \quad \textbf{a.s.} \\ & h_{i0}^+ \geq 0, \;\; i \in \mathcal{N} \cup \{0\} \\ & x_{i0}, y_{i0} \geq 0, \;\; i \in \mathcal{N} \\ & g_0 \;\text{ free}, g_2 \geq 0 \end{aligned}\right\} \tag{5.8}$$

In the above formulation, we say that decisions (g_0, g_2) are $\mathcal{F}_1$-measurable. That is, to compute decisions $[g_0, g_2]$ we need first-stage decisions $\left(\boldsymbol{h}_0^+, \boldsymbol{x}_0, \boldsymbol{y}_0\right)$ and realization $\boldsymbol{R}_1$.

Deterministic Equivalent Linear Program

In order to render model (5.8) tractable, we need to generate a finite number of realizations of stochastic process $\boldsymbol{R}_1$. If $\mathcal{S}$ is a set with S realizations of vector $\boldsymbol{R}_1$ and p_s is the probability associated with realization s such that $\sum_{s=1}^{S} p_n = 1$, then problem (5.8) can be approximated by the following linear program:

$$\left.\begin{aligned}
\max_{(\boldsymbol{h}_0^+,\boldsymbol{x}_1,\boldsymbol{y}_1,g_0,g_2^s)} \quad & \gamma \sum_{s=1}^{S} p_s \left[\sum_{i=0}^{N} R_{i1}^s h_{i0}^+ \right] - (1-\gamma) g_0 - \frac{1-\gamma}{1-\beta} \sum_{s=1}^{S} p_s g_2^s \\
\text{s.t.} \quad & h_{i0}^+ = h_{i0} + x_{i0} - y_{i0}, \;\; i \in \mathcal{N} \\
& h_{00}^+ = h_{00} - (1+\theta) \sum_{i=1}^{N} x_{i0} + (1-\theta) \sum_{i=1}^{N} y_{i0} \\
& g_2^s \geq -g_0 - \sum_{i=0}^{N} R_{i1}^s h_{i0}^+ + \sum_{i=0}^{N} h_{i0}, \;\; s \in \mathcal{S} \\
& h_{i0}^+ \geq 0, \;\; i \in \mathcal{N} \cup \{0\} \\
& x_{i0}, y_{i0} \geq 0, \;\; i \in \mathcal{N} \\
& g_0 \;\text{ free}, g_2^s \geq 0, s \in \mathcal{S}
\end{aligned}\right\} \tag{5.9}$$

Problem (5.9) is often referred to as the *deterministic equivalent linear program* or simply the *deterministic equivalent*. Note that in the linear program (5.9) we have a total of $3N + S + 2$ decision variables and $N + S + 1$ constraints. That is, the size of problem (5.9) grows linearly with the number of assets and realizations.

In this study, we solve problem (5.9) using mathematical programming methods. However, note that when the number of realizations becomes too large, problem (5.9) becomes intractable. In such a case, to solve problem (5.9), we would need to use *decomposition methods*. We refer the reader to [10] for a general discussion on how *Benders decomposition* (this is due to Benders, see [5]) can be used to solve two-stage stochastic linear programs. Specifically for those with CVaR in the objective, Benders decomposition has been implemented in [51] as the CVaRMin solver where due to the CVaR constraints sharing the common variable g_0, the optimal value of which gives us VaR, this is determined in the first stage with decisions X_0 through a constraint generation process.

5.1.2 The Multi-period Portfolio as a Multistage Stochastic Program

In this section, we expand the one-period portfolio model of the previous section by including multiple time periods. Assuming a planning horizon of T periods, we write the multistage stochastic programming formulation of the multi-period portfolio selection problem and to solve it we generate a finite approximation of the

stochastic process of random returns which is in the form of a *scenario tree*.

Multistage Stochastic Programming Formulation

Assuming a time horizon with $T+1$ discrete points in time (i.e. T time periods), we start with known holdings $\boldsymbol{h}_0$ and in a first-stage we decide how much to buy and sell from every risky asset. The first-stage decisions are denoted by vector $X_0 = \left(\boldsymbol{h}_0^+, \boldsymbol{x}_0, \boldsymbol{y}_0\right)$ and as in the single-period portfolio problem of section 5.1.1 they must satisfy the deterministic constraints of action space $\mathcal{X}_0$ which includes constraints (5.1)-(5.4).

The main difference from the single-period portfolio problem of section 5.1.1 is that here we measure terminal wealth and CVaR at the end of the time horizon which is at time T, and every time a random return vector $\boldsymbol{R}_t$ is revealed we take recourse decisions $X_t = \left(\boldsymbol{h}_t^+, \boldsymbol{x}_t, \boldsymbol{y}_t\right)$, which from (2.2) and (2.5) must satisfy the following constraints:

$$h_{it}^+ = R_{it} h_{i(t-1)}^+ + x_{it} - y_{it}, \ \textbf{a.s.}, \ i \in \mathcal{N}, \tag{5.10}$$

$$h_{0t}^+ = R_{0t} h_{0(t-1)}^+ - (1+\theta)\sum_{i=1}^{N} x_{it} + (1-\theta)\sum_{i=1}^{N} y_{it}, \tag{5.11}$$

$$h_{it}^+ \geq 0, \ i \in \mathcal{N} \cup \{0\}, \tag{5.12}$$

$$x_{it}, y_{it} \geq 0, \ i \in \mathcal{N}, \tag{5.13}$$

where as explained earlier in this chapter *a.s.* is an abbreviation for *almost surely* and means that the constraints with which it is associated are satisfied with probability one.

Constraints (5.10)-(5.13) define action space $\mathcal{X}_t$, i.e. for $t = 1, 2, \ldots, T-1$ we have:

$$\mathcal{X}_t\left(X_{t-1}, \boldsymbol{R}_t\right) = \left\{\left(\boldsymbol{h}_t^+, \boldsymbol{x}_t, \boldsymbol{y}_t\right) : \ (5.10) - (5.13) \text{ hold}\right\} \tag{5.14}$$

Note that constraints of type (5.10) depend on random variables $\boldsymbol{R}_t$ as well as on decisions X_{t-1} and they must be satisfied with probability one. Further, decisions g_0 and g_2 as well as terminal wealth $\sum_{i=0}^{N} R_{iT} h_{i(T-1)}^+$ need decisions X_{T-1} and the realizations of random variables $\boldsymbol{R}_T$.

Figure 5.3 shows the timing of events and the stages in the multi-period portfolio problem. Note that for a T-period planning horizon, the multi-period portfolio selection problem with CVaR as a measure of risk has $T + 1$ stages.

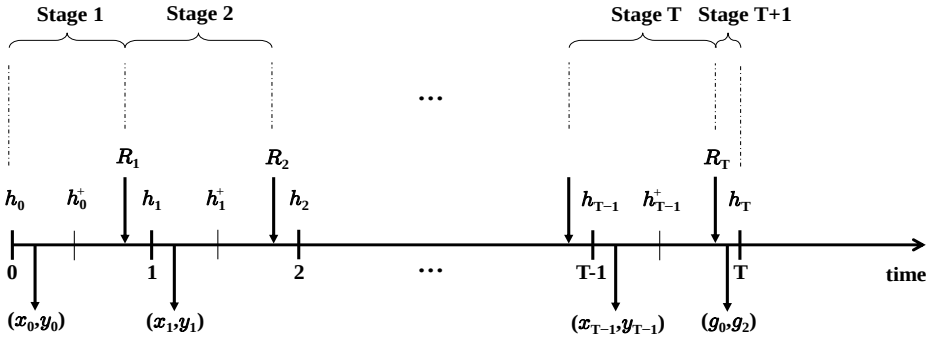


Figure 5.3: Timing of events and stages in the multi-period portfolio selection problem

We are now ready to write the multistage stochastic programming formulation of the portfolio selection problem.

If we expand the two-stage stochastic programming formulation (5.8) to include the decisions as well as the constraints of the additional time periods, then this becomes:

$$
\left.
\begin{aligned}
\max_{(\boldsymbol{h}_t^+,\boldsymbol{x}_t,\boldsymbol{y}_t,g_0,g_2)} \quad & \gamma\mathbb{E}\left\{\sum_{i=0}^{N} R_{iT}h_{i(T-1)}^+\right\} - (1-\gamma)g_0 - \frac{1-\gamma}{1-\beta}\mathbb{E}\left\{g_2\right\} \\
\text{s.t.} \quad & h_{i0}^+ = h_{i0} + x_{i0} - y_{i0}, \;\; i \in \mathcal{N} \\
& h_{00}^+ = h_{00} - (1+\theta)\sum_{i=1}^{N} x_{i0} + (1-\theta)\sum_{i=1}^{N} y_{i0} \\
& h_{it}^+ = R_{it}h_{i(t-1)}^+ + x_{it} - y_{it}, \;\; i \in \mathcal{N}, t \in \mathcal{T}\setminus\{0,T\}, \;\; \textbf{a.s.} \\
& h_{0t}^+ = R_{0t}h_{0(t-1)}^+ - (1+\theta)\sum_{i=1}^{N} x_{it} + (1-\theta)\sum_{i=1}^{N} y_{it}, t \in \mathcal{T}\setminus\{0,T\} \\
& g_2 \geq -g_0 - \sum_{i=0}^{N} R_{iT}h_{i(T-1)}^+ + \sum_{i=0}^{N} h_{i0}, \quad \textbf{a.s.} \\
& h_{it}^+ \geq 0, \;\; i \in \mathcal{N}\cup\{0\}, t \in \mathcal{T}\setminus\{T\} \\
& x_{it}, y_{it} \geq 0, \;\; i \in \mathcal{N}, t \in \mathcal{T}\setminus\{T\} \\
& g_0 \;\; \text{free}, g_2 \geq 0
\end{aligned}
\right\} \tag{5.15}
$$

which is the multistage stochastic programming formulation of the portfolio selection problem. In the above formulation, we say that decisions X_t are $\mathcal{F}_t$-measurable.

Scenario Trees

In order to render problem (5.15) tractable, we need to discretize the stochastic process of random returns and generate a finite number of sample paths called *scenarios*. However, even after a discretization of the underlying stochastic process, we are often faced with a very large number of scenarios, which still makes it difficult to solve problem (5.15).

To circumvent this inefficiency, it is very useful to approximate the initial distribution of scenario paths with another distribution that is close to the original one, carries less information and exhibits a tree structure. The resulting approximate distribution is called a *scenario tree* and apart from less information it makes sense as events unfold in time. If the scenario tree is sufficiently small, then we can proceed with numerical calculations.

Figure 5.4 shows a scenario tree with $T+1$ stages, K_{T+1} nodes and $K_{T+1}-K_T$ scenarios.

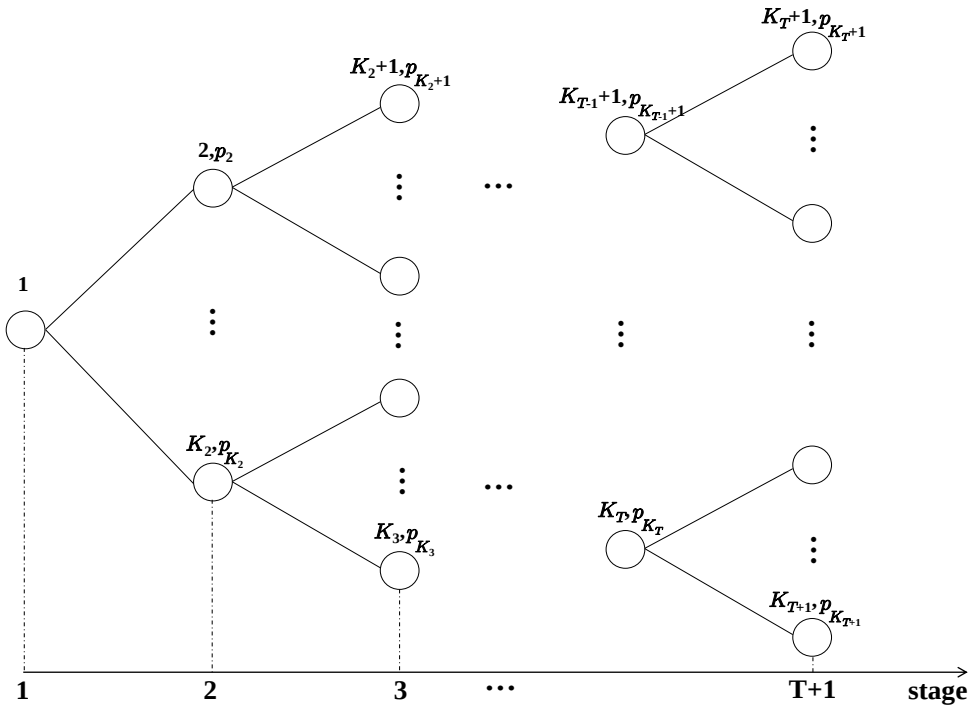


Figure 5.4: A scenario tree with $T+1$ stages, K_{T+1} nodes and $K_{T+1}-K_T$ scenarios

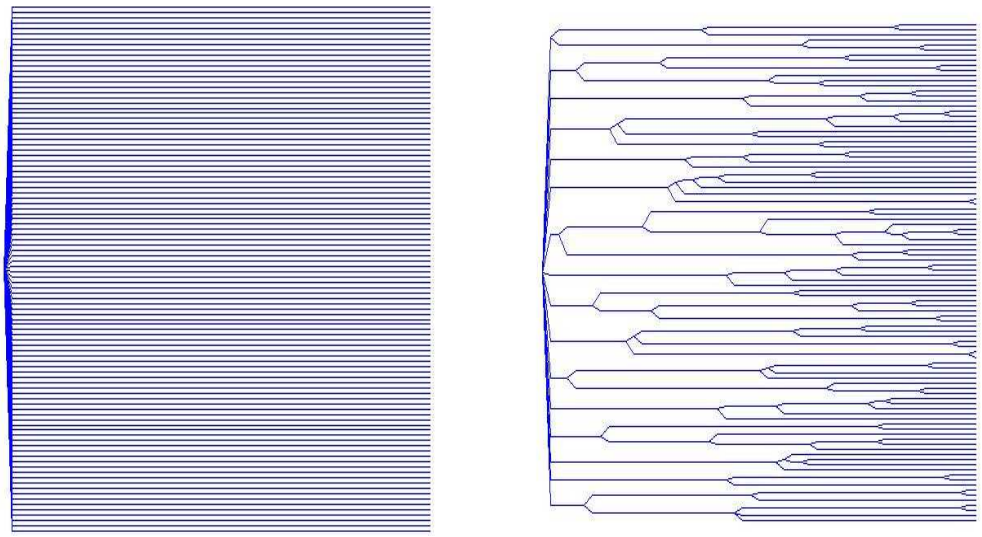
A scenario tree, such as the one shown in Figure 5.4, has levels that relate to stages as follows: At level 1, we have a unique *root* node that defines a first stage and is associated with decisions taken before the arrival of any exogenous information. Then, at level 2 (this is after the arrival of information $\boldsymbol{R}_1$), stage 2 is defined, where the number of nodes is determined by the number of realizations of vector $\boldsymbol{R}_1$. Every node of stage 2 is connected with the root node by an *arc*. Generally, at level t, stage t is defined. Every node at level t corresponds to a realization of vector $\boldsymbol{R}_{t-1}$, has a unique predecessor node at stage $t-1$, called the *ancestor node*, and is also connected with nodes at stage $t+1$, called the *children nodes*.

We are now ready to construct a scenario tree that will allow us to render problem (5.15) tractable. As explained above, every node n is associated with a stage t and a realization of the random return vector. We let $\mathcal{K}$ be the set of all nodes on the scenario tree, and $\mathcal{K}_t$ the set of nodes in stage t, such that $\mathcal{K}_1 = \{1\}$ and $\mathcal{K}_t = \{K_{t-1}+1, ..., K_t\}$ for $t_n > 1$. Every node n is connected with a unique parent node, which we denote with k_n, and a number of children nodes for which we define set $\mathcal{C}(n)$. The last-stage nodes are called the *leaf nodes*.

Note that the tree structure defined above allows us to specify the conditional distribution of $\boldsymbol{R}_t$ given history $(\boldsymbol{R}_1, \boldsymbol{R}_2, \ldots, \boldsymbol{R}_{t-1})$. That is, if we are currently at node n, then we denote the *conditional probability* of transiting from node n to one of its children nodes, say node $m \in \mathcal{C}(n)$, with ρ_{nm} and we have: $\rho_{nm} \geq 0$ and $\sum_{m\in\mathcal{C}(n)} \rho_{nm} = 1$. Using the conditional probabilities, we define the probability of a node n at stage t to be the product of all the conditional probabilities that are associated with the transitions required to move from the root node to node n of stage t, and we denote this probability with p_n.

In this study, we assume that the original distribution is a group of scenario paths and we use *scenario tree construction* algorithms to approximate the original distribution with a new one that has less nodes and is in the form of a scenario tree. These algorithms, which have been implemented both in a forward and a backward fashion, use in every stage *scenario reduction* to delete a subset of the initial scenarios, updating at the same time the probabilities of the preserved scenarios with the probabilities of the deleted ones. The subset of the preserved scenarios has a prescribed cardinality and is closest to the original set of scenarios in terms of a probability metric out of the family of Wasserstein metrics. For other scenario generation methods see [25], [41] and [45].

Figure 5.5a shows a distribution with 100 scenario paths and figure 5.5b shows the approximate distribution after applying backward scenario tree construction.



(a) Initial distribution with 100 scenario paths (b) Approximate distribution after scenario tree construction

Figure 5.5: A distribution of 100 scenarios on the left and its approximation on the right.

In line with the notation in [37], in Appendix B we provide a brief description of how scenario reduction can be used in constructing scenario trees. Different variants of scenario tree construction algorithms can be found in [43].

Deterministic Equivalent Linear Program

To proceed with numerical calculations in (5.15), we replace the expectation in (5.15) with the sum of the probabilities of the last-stage nodes of the scenario tree of Figure 5.4 and approximate stochastic problem (5.15) with the following deterministic problem:

$$
\left.
\begin{aligned}
\max_{\substack{(\boldsymbol{h}_n^+,\boldsymbol{x}_n,\boldsymbol{y}_n \\ ,g_0,g_2^n)}} \quad & \gamma \sum_{n=K_T+1}^{K_{T+1}} p_n \sum_{i=0}^{N} R_{in} h_{ik_n}^+ - (1-\gamma) g_0 - \frac{1-\gamma}{1-\beta} \sum_{n=K_T+1}^{K_{T+1}} p_n g_2^n \\
\text{s.t.} \quad & h_{i1}^+ = h_{i0} + x_{i1} - y_{i1}, \;\; i \in \mathcal{N} \\
& h_{01}^+ = h_{00} - (1+\theta) \sum_{i=1}^{N} x_{i1} + (1-\theta) \sum_{i=1}^{N} y_{i1} \\
& h_{in}^+ = R_{in} h_{ik_n}^+ + x_{in} - y_{in}, \;\; i \in \mathcal{N}, \;\; n = 2, \ldots, K_T \\
& h_{0n}^+ = R_{0n} h_{0k_n}^+ - (1+\theta) \sum_{i=1}^{N} x_{in} + (1-\theta) \sum_{i=1}^{N} y_{in}, \;\; n = 2, \ldots, K_T \\
& g_2^n \geq -g_0 - \sum_{i=0}^{N} R_{in} h_{ik_n}^+ + \sum_{i=0}^{N} h_{i0}, \;\; n = K_T+1, \ldots, K_{T+1} \\
& h_{in}^+ \geq 0, \;\; i \in \mathcal{N} \cup \{0\}, n = 1, \ldots, K_T \\
& x_{in}, y_{in} \geq 0, \;\; i \in \mathcal{N}, n = 1, \ldots, K_T \\
& g_0 \;\; \text{free}, g_2^n \geq 0, \;\; n = K_T+1, \ldots, K_{T+1}
\end{aligned}
\right\} \tag{5.16}
$$

Note that in problem (5.16) non-anticipativity is imposed by the scenario tree in an implicit form.

Problem (5.16) is often referred to as the *deterministic equivalent linear program* or simply the *deterministic equivalent*. Note that the decisions taken at any node $n \in \mathcal{K} \setminus \{1\}$ depend on the decisions taken at parent node k_n, thus creating a *staircase structure*, through which information is carried forward in time. In stochastic programming, this structure allows us to decompose the deterministic equivalent in a sequence of smaller problems, one for each node of the tree, and solve it using decomposition algorithmic methods (see for example [32] for an implementation of Benders decomposition and [44] for other decomposition algorithms). However, in our problem, due to the CVaR constraints sharing one common variable g_0 (this is the one that gives us VaR) we cannot decompose problem (5.16) and use decomposition algorithms to solve it. Instead, we solve the big deterministic problem (5.16) using mathematical programming methods.

We conclude this section by discussing complexity issues. In the linear program (5.16), we have a total of $(K_{T+1} + 3NK_T + 1)$ decision variables, $(3N+1)K_T$ from the first K_T nodes and $(K_{T+1} - K_T + 1)$ from the leaf nodes, and $(K_{T+1} + NK_T)$ constraints, $(N+1)K_T$ from the first K_T nodes and $K_{T+1} - K_T$ from the

leaf nodes. That is, the size of problem (5.16) depends on the number of nodes as well as on the number of assets.

5.2 The Naive Equally-weighted Portfolio

In this section, we formulate the equally-weighted portfolio model, where we take buying and selling decisions so that the post-decision total wealth is equally split among the assets (including cash). The reason why we study this specific type of portfolio models is that in a recently published paper (see [22]) the equally-weighted portfolio has worked considerably well as compared to a wide range of models across several datasets.

To have the same amount of holdings in all assets after taking decisions $(\boldsymbol{x}_t, \boldsymbol{y}_t)$, we require that the following condition holds:

$$h_{it}^{+} = \frac{1}{N+1} \sum_{j=0}^{N} h_{jt}^{+}, \;\; i \in \mathcal{N} \cup \{0\} \tag{5.17}$$

Combining (5.17) with equations (2.2), in period $t+1$ we want to find the non-negative feasible decisions $\left(\boldsymbol{h}_t^{+}, \boldsymbol{x}_t, \boldsymbol{y}_t\right)$ that satisfy the following system of equations and inequalities:

$$\left.\begin{aligned} h_{it}^{+} &= R_{it} h_{i(t-1)}^{+} + x_{it} - y_{it}, \;\; i \in \mathcal{N} \\ h_{0t}^{+} &= R_{0t} h_{0(t-1)}^{+} - (1+\theta) \sum_{i=1}^{N} x_{it} + (1-\theta) \sum_{i=1}^{N} y_{it} \\ h_{it}^{+} &= \frac{1}{N+1} \sum_{j=0}^{N} h_{jt}^{+}, \;\; i \in \mathcal{N} \cup \{0\} \end{aligned}\right\} \tag{5.18}$$

where $R_{it} h_{i(t-1)}^{+}$ is known when we take decisions $\boldsymbol{x}_t$ and $\boldsymbol{y}_t$, and for $t = 0$ it is replaced by h_{i0}.

In (5.18), we have a total of $3N + 1$ decision variables and $2N + 2$ equations which gives us infinitely many solutions. Among these solutions, there might be some which involve buying and selling the same assets. To avoid these transactions which incur unnecessary transaction costs, we impose an objective which requires that we maximize total wealth after taking the buying and selling decisions. Thus, the equally-weighted portfolio problem can be expressed as the following linear program:

$$\left.\begin{aligned}
\max_{(\boldsymbol{h}_t^+, \boldsymbol{x}_t, \boldsymbol{y}_t)} \quad & \sum_{i=0}^{N} h_{it}^+ \\
\text{s.t.} \quad & h_{it}^+ = R_{it} h_{i(t-1)}^+ + x_{it} - y_{it}, \;\; i \in \mathcal{N} \\
& h_{0t}^+ = R_{0t} h_{0(t-1)}^+ - (1+\theta) \sum_{i=1}^{N} x_{it} + (1-\theta) \sum_{i=1}^{N} y_{it} \\
& h_{it}^+ = \frac{1}{N+1} \sum_{j=0}^{N} h_{jt}^+, \;\; i \in \mathcal{N} \cup \{0\} \\
& h_{it}^+ \geq 0, \;\; i \in \mathcal{N} \cup \{0\} \\
& x_{it}, y_{it} \geq 0, \;\; i \in \mathcal{N}
\end{aligned}\right\} \tag{5.19}$$

Note that in problem (5.19) if we substitute the first $N+1$ equations in the objective, then this will become equal to $-\theta \left(\sum_{i=1}^{N} x_{it} + \sum_{i=1}^{N} y_{it} \right) + \sum_{i=0}^{N} R_{it} h_{i(t-1)}^+$. That is, in problem (5.19) we want to decide how to allocate our wealth between the assets so that we pay the minimum transaction costs.

In this study, we consider two types of investment strategies to evaluate the equally-weighted portfolio. First, a *buy-and-hold* strategy where we solve problem (5.19) only once at the beginning of the investment horizon and hold the selected portfolio until the end of the investment horizon without doing any rebalancing. Second, a *fixed-mix* strategy where we assume the following decision rule: In each time period, the portfolio is rebalanced so that the total available wealth is equally split among the assets. Assuming a planning horizon of T periods, in the fixed-mix strategy problem (5.19) needs to be solved T times. For a discussion about the different investment strategies considered, we refer the reader to [31] and [64].

Part IV

Approximate Dynamic Programming Methods

Chapter 6

Separable Linear Approximations

In this chapter, we implement separable linear functional approximations for the unknown value functions of the assets in the dynamic programming formulation of the portfolio selection problem. Figure 6.1 shows a plot of a linear approximate value function for an asset i at time t.

Note that a linear function is not a good fit for the true value functions of the assets (these are concave due to CVaR) because the distance between the approximate and the true value function becomes too large after increasing our holdings beyond a certain amount. Further, as we will explain later on, in a separable linear approximation every additional unit of wealth in the assets is valued the same thus ignoring risk and resulting in portfolios that comprise of only one asset per time period.

To achieve a better fit for the true value functions we impose upper bound constraints on the holdings of the risky assets (we assume that risky assets are only available up to a certain amount). Here, from each approximate value function we crop the linear part that is too far from the true value function when the holdings of the assets increase beyond a certain amount and we keep the linear part which corresponds to smaller amounts of holdings since this is closer to the true value function. Unlike the simple linear approximation which ignores risk and leads to portfolios that comprise of only asset per time period, here, although we do not directly account for risk since our value functions are still linear, imposing upper bounds on how much we can hold in each asset leads to diversified portfolios which indirectly accounts for risk as diversification can protect us from high losses. However, despite the improvement achieved in the composition of the selected portfolios, we do not know the actual position of these bounds. Further, by imposing upper bound

constraints we cut off states which might be good but we never get the chance to visit.

Note that the linear approximation methods are quite naive but due to their simplicity they have been extensively used and have many applications in various fields (see for example [61] for an application in wireless mesh networks, [62] for an application in the batch dispatch problem, [66] for an application in resource allocation problems and [80] for an application in an inventory allocation problem). In the portfolio selection problem, however, they do not capture the nature of the problem. Their role is solely expository and help us introduce the reader to the approximate dynamic programming methods. Nevertheless, in chapter 7, where we consider separable piecewise linear approximations for the unknown value functions (these are more appropriate for the portfolio selection problem as they capture the expected risk attitude of an investor), we will see that the optimization problem that we need to solve in every time period to take our rebalancing decisions resembles the respective optimization problem of the bounded linear approximation.

This chapter is structured as follows: In section 6.1, we write the optimality equations for the assumed separable linear approximate value functions. In sections 6.2 and 6.3, we write the linear programming formulations of the subproblems of ADP for the assumed separable linear approximations without and with upper bounds on the holdings of the risky assets and solve them. Then, in section 6.4 we provide a discussion about the linear approximate methods. Finally, in section 6.5 we compute the gradients for Step 3 of the General ADP Algorithm 4.1.

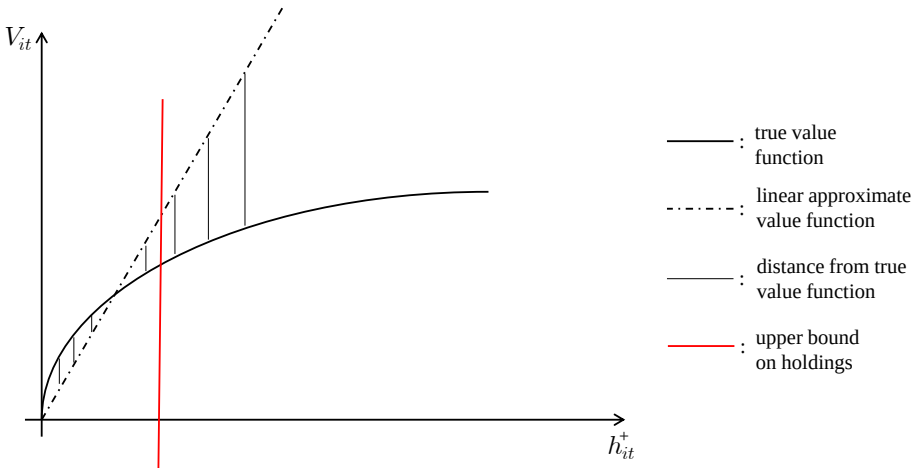


Figure 6.1: Linear approximate value function of asset i in period $t+1$ with an upper bound on its holdings

6.1 Linear Approximations for the Value Functions

Recall from (4.30) and (4.32) that in order to compute decisions $(\boldsymbol{x}_t, \boldsymbol{y}_t)$ for $t = 0, 1, \ldots, T-1$ we use the following optimality equations:

$$\left.\begin{array}{l} \tilde{V}_0^- (\boldsymbol{h}_0) = \max\limits_{(\boldsymbol{x}_0, \boldsymbol{y}_0) \in \mathcal{A}_0} \hat{V}_0 \left(\boldsymbol{h}_0^+\right) \\ \tilde{V}_{t-1} \left(\boldsymbol{h}_{t-1}^+\right) = \max\limits_{(\boldsymbol{x}_t, \boldsymbol{y}_t) \in \mathcal{A}_t} \hat{V}_t \left(\boldsymbol{h}_t^+\right),\ t = 1, 2, \ldots, T-1 \end{array}\right\} \tag{6.1}$$

where for simplicity of notation here we have removed iteration indexing.

In this section, we implement separable linear functional approximations for the unknown value functions of the assets. Specifically, we replace each $\hat{V}_t \left(\boldsymbol{h}_t^+\right)$ in (6.1) with a separable approximation of the following type:

$$\hat{V}_t \left(\mathbf{h}_t^+\right) = \sum_{i=0}^{N} \hat{V}_{it} \left(h_{it}^+\right), \tag{6.2}$$

where we assume that every function $\hat{V}_{it} \left(h_{it}^+\right)$ is given by:

$$\hat{V}_{it} \left(h_{it}^+\right) = u_{it} h_{it}^+,\ \ i \in \mathcal{N} \cup \{0\}, \tag{6.3}$$

where $u_{it} > 0$. Every slope u_{it} can be understood as the value of holding one monetary unit of asset i from period $t+1$ until the end of the time horizon.

Substituting the above approximation in optimality equations (6.1) we get:

$$\left.\begin{array}{l} \tilde{V}_0^- (\boldsymbol{h}_0) = \max\limits_{(\boldsymbol{x}_0, \boldsymbol{y}_0) \in \mathcal{A}_0} \sum\limits_{i=0}^{N} u_{i0} h_{i0}^+ \\ \tilde{V}_{t-1} \left(\boldsymbol{h}_{t-1}^+\right) = \max\limits_{(\boldsymbol{x}_t, \boldsymbol{y}_t) \in \mathcal{A}_t} \sum\limits_{i=0}^{N} u_{it} h_{it}^+,\ t = 1, 2, \ldots, T-1 \end{array}\right\} \tag{6.4}$$

Further, if we substitute every h_{it}^+ in (6.4) using transition equations (4.5), then optimality equations (6.4) take the following form:

$$\left.\begin{aligned}
\tilde{V}_0^{-}(\boldsymbol{h}_0) = & \max_{(\boldsymbol{x}_0,\boldsymbol{y}_0)\in\mathcal{A}_0} \sum_{i=1}^{N}\left[u_{i0}-(1+\theta)\right]x_{i0} + \sum_{i=1}^{N}\left[-u_{i0}+(1-\theta)\right]y_{i0} \\
& + \sum_{i=0}^{N} h_{i0} \\
\tilde{V}_{t-1}\left(\boldsymbol{h}_{t-1}^{+}\right) = & \max_{(\boldsymbol{x}_t,\boldsymbol{y}_t)\in\mathcal{A}_t} \sum_{i=1}^{N}\left[u_{it}-(1+\theta)\right]x_{it} + \sum_{i=1}^{N}\left[-u_{it}+(1-\theta)\right]y_{it} \\
& + \sum_{i=0}^{N} R_{it}h_{i(t-1)}^{+},\ t=1,2,\ldots,T-1
\end{aligned}\right\} \tag{6.5}$$

In this thesis we consider two linear approximate schemes. First, we impose no restrictions on how much we can hold in each asset. In the thesis, we will call the General ADP Algorithm 4.1 after we replace $\hat{V}_t\left(\mathbf{h}_t^{+}\right)$ with a sum of separable linear functional approximations of type (6.3) and without any restrictions on the holdings of the assets the *Linear Approximate Dynamic Programming Algorithm (LADP)*. Second, we impose upper bounds on the holdings of the risky assets (note that we allow the investor to hold as much cash as he wants). In the thesis, we will call the LADP algorithm with the extra upper bounds on the holdings of each risky asset the *Linear Approximate Dynamic Programming Algorithm with Upper Bounds (LADP-UB)*. What changes in the latter approximate scheme is the feasible region of the maximization problems in (6.5) which is expanded to include the upper bound constraints on the holdings of the risky assets.

For a given realization $\boldsymbol{R}_t$ and a given state $\boldsymbol{h}_{t-1}^{+}$ an instance of (6.5) without and with upper bounds is called respectively *the subproblem of LADP* and *the subproblem of LADP-UB*. Note that the main difference between the first subproblem, where we compute decisions $(\boldsymbol{x}_0, \boldsymbol{y}_0)$, and the other subproblems, where we compute decisions $(\boldsymbol{x}_t, \boldsymbol{y}_t)$ for $t = 1, 2, \ldots, T-1$, is constant $R_{it}h_{i(t-1)}^{+}$ which in the objective and the constraints of the first subproblem is replaced by constant h_{i0}. Since the two types of subproblems are similar, here we solve the subproblem of LADP for $t = 1, 2, \ldots, T-1$ and state the analogous solution for $t = 0$.

6.2 The Subproblem of LADP

The optimality equations in (6.5) consist of many maximization problems, each one associated with the respective decision variables $(\boldsymbol{x}_t, \boldsymbol{y}_t)$. These are called the subproblems of LADP. In this section, we write the linear programming formulation of every subproblem of LADP as a linear program and solve it.

Linear Programming Formulation of the Subproblem of LADP

Using action space (4.14), the optimization problem in (6.5) for $t = 1, 2, \ldots, T-1$ is given by:

$$\left.\begin{aligned} \max_{(\boldsymbol{x}_t, \boldsymbol{y}_t)} \quad & \sum_{i=1}^{N} k_{it} x_{it} + \sum_{i=1}^{N} l_{it} y_{it} + \sum_{i=0}^{N} u_{it} R_{it} h^{+}_{i(t-1)} \\ \text{s.t.} \quad & -x_{it} + y_{it} \leq R_{it} h^{+}_{i(t-1)}, \;\; i \in \mathcal{N} \\ & \sum_{i=1}^{N} x_{it} \leq \frac{1}{1+\theta} R_{0t} h^{+}_{0(t-1)} + \frac{1-\theta}{1+\theta} \sum_{i=1}^{N} y_{it} \\ & x_{it}, y_{it} \geq 0, \;\; i \in \mathcal{N} \end{aligned}\right\} \tag{6.6}$$

where $k_{it} = u_{it} - (1+\theta)\, u_{0t}$ and $l_{it} = -u_{it} + (1-\theta)\, u_{0t}$ are the coefficients of the buying and selling variables in the objective. Coefficient k_{it} can be understood as the value of buying one monetary unit of asset i in period $t+1$ with $(1+\theta)$ units of cash and will be referred to as the *buying slope*. Coefficient l_{it} can be understood as the value of selling one monetary unit of asset i in period $t+1$, which increases cash by $(1-\theta)$ units and will be referred to as the *selling slope*.

Buying and Selling Slopes

Before we proceed with solving problem (6.6), we will first derive the properties of the buying and selling slopes and discuss what these mean to our problem. These properties will be of great value in understanding and solving problem (6.6) and are summarized in the following Lemma.

Lemma 6.2.1. *Let k_{it} and l_{it} be respectively the buying and selling slopes of asset i in period t+1. Then, the following statements hold:*

1. *If $k_{it} > 0$, then $l_{it} < -2\theta u_{0t}$ and $|l_{it}| > k_{it}$.*
2. *If $l_{it} > 0$, then $k_{it} < -2\theta u_{0t}$ and $|k_{it}| > l_{it}$.*

3. $-2\theta u_{0t} \leq k_{it} \leq 0$ *if and only if* $-2\theta u_{0t} \leq l_{it} \leq 0$.

Proof

1. Suppose $k_{it} > 0$. Then,

$$k_{it} = u_{it} - (1+\theta)\, u_{0t} > 0 \Rightarrow u_{it} > (1+\theta)\, u_{0t}$$

   Using the above inequality we have:

$$l_{it} = -u_{it} + (1-\theta)\, u_{0t} < -(1+\theta)\, u_{0t} + (1-\theta)\, u_{0t} = -2\theta u_{0t} < 0$$

   It is easy to verify that $l_{it} = -k_{it} - 2\theta u_{0t}$, from which $|l_{it}| > k_{it}$ follows immediately.

2. Suppose $l_{it} > 0$. Then,

$$l_{it} = -u_{it} + (1-\theta)\, u_{0t} > 0 \Rightarrow u_{it} < (1-\theta)\, u_{0t}$$

   Using the above inequality we have:

$$k_{it} = u_{it} - (1+\theta)\, u_{0t} < (1-\theta)\, u_{0t} - (1+\theta)\, u_{0t} = -2\theta u_{0t} < 0$$

   It is easy to verify that $k_{it} = -l_{it} - 2\theta u_{0t}$, from which $|k_{it}| > l_{it}$ follows immediately.

3. Suppose $-2\theta u_{0t} \leq k_{it} \leq 0$. Then,

$$\begin{gathered} -2\theta u_{0t} \leq k_{it} \leq 0 \Leftrightarrow \\ -2\theta u_{0t} \leq u_{it} - (1+\theta)\, u_{0t} \leq 0 \Leftrightarrow \\ (1-\theta)\, u_{0t} \leq u_{it} \leq (1+\theta)\, u_{0t} \Leftrightarrow \\ -2\theta u_{0t} \leq -u_{it} + (1-\theta)\, u_{0t} \leq 0 \Leftrightarrow \\ -2\theta u_{0t} \leq l_{it} \leq 0 \quad \blacksquare \end{gathered}$$

Remark 6.1. Due to the relationships between the slopes, it is easy to verify the following:

1. $k_{it} = 0 \Leftrightarrow u_{it} = (1+\theta)\, u_{0t}$
2. $l_{it} = 0 \Leftrightarrow u_{it} = (1-\theta)\, u_{0t}$
3. $k_{it} = l_{it} = 0 \Leftrightarrow u_{it} = u_{0t} = 0$

However, as we will explain later on, the values of the slopes u_{it} depend on the random returns, which are positive and differ both from one asset to another and from one iteration to another. Therefore, it is unlikely that buying and selling slopes will ever take the above values.

Considering now the relationships between k_{it} and l_{it} from Lemma 6.2.1, we define the following categories (sets) of assets:

1. *Buy-assets*: Those assets for which $k_{it} > 0$ and $l_{it} < 0$. These assets are candidates for buying. Later on, we will see that some of these assets may be considered for selling in order to buy another asset of the same category.
2. *Sell-assets*: Those assets for which $l_{it} > 0$ and $k_{it} < 0$. Later on, we will see that the Sell-assets with positive holdings are sold and converted to cash.
3. *Neutral-assets*: Those assets for which $-2\theta u_{0t} \leq l_{it}, k_{it} \leq 0$. Later on, we will see that these assets may be considered for selling in order to buy a Buy-asset.

Let now $j^\star$ be the Buy-asset with the highest buying slope, which is given by $j^\star = \{\arg\max_{i \in \mathcal{N}} k_{it} : \, k_{it} > 0\}$ and is empty if set Buy-assets is empty. We introduce a new category (set) which we call *Sell-to-buy-assets* and includes all Buy- and Neutral-assets for which $\frac{1-\theta}{1+\theta} k_{j^\star t} + l_{it} > 0$ holds. The new set overlaps with Neutral-assets and some of the Buy-assets and, as we will see later on, the Sell-to-buy-assets with positive holdings will be sold in order to buy asset $j^\star$. From this point on, we will call the transactions between Sell-to-buy-assets and asset $j^\star$ the *sell-to-buy* transactions.

Figure (6.2) illustrates the categories of the assets on a pie chart and summarizes the conditions that the assets of each category satisfy.

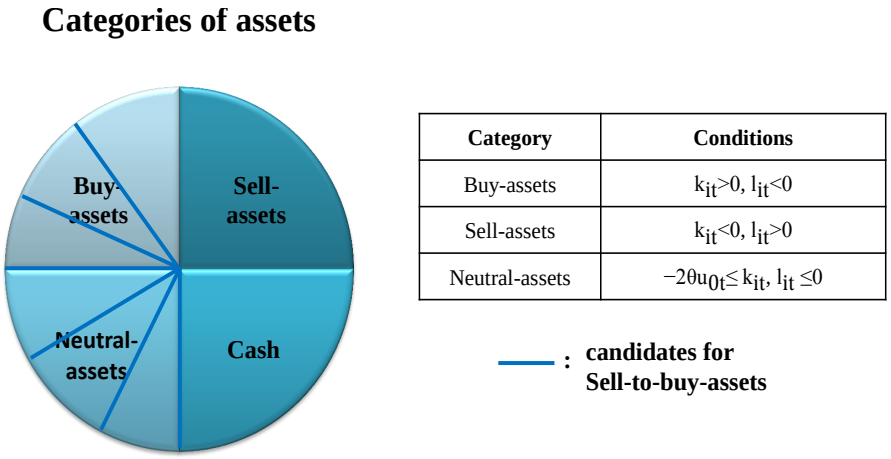


Figure 6.2: Categories of assets and conditions

Solution to the Subproblem of LADP

We are now ready to solve problem (6.6). For this, we will need the following sets:

1. $\mathcal{I}$ = Sell-assets & $\left(R_{it}h^{+}_{i(t-1)} > 0\right)$: All Sell-assets that have positive holdings (these assets will be sold).
2. $\mathcal{J}$ = Sell-to-buy-assets & $\left(R_{it}h^{+}_{i(t-1)} > 0\right)$: All Sell-to-buy-assets that have positive holdings (these assets will be sold in order to buy asset $j^{\star}$).

We propose the following algorithm:

Begin with setting $x_{it}, y_{it} := 0$ for every asset $i \in \mathcal{N}$, assign risky assets to one of the non-overlapping categories of Figure 6.2, pick asset $j^{\star}$ by setting:

$$j^{\star} := \left\{ \arg\max_{i \in \mathcal{N}} k_{it} : \; k_{it} > 0 \right\}, \tag{6.7}$$

and initialize sets $\mathcal{I}$ and $\mathcal{J}$. This is Step 0.

If set $\mathcal{I}$ is non-empty, then take every asset i of this set, sell $R_{it}h^{+}_{i(t-1)}$ units and update cash by setting:

$$\begin{aligned} y_{it} &:= R_{it}h^{+}_{i(t-1)}, \; \forall i \in \mathcal{I} \\ h_{0t} &:= R_{0t}h^{+}_{0(t-1)} + (1-\theta)\sum_{i \in \mathcal{I}} R_{it}h^{+}_{i(t-1)} \end{aligned} \tag{6.8}$$

This is Step 1.

Next, after Step 1, if there exist an asset $j^{\star}$ and cash, then we use all cash to buy $\frac{1}{1+\theta}h_{0t}$ units of asset $j^{\star}$ by setting:

$$\begin{aligned} x_{j^\star t} &:= \frac{1}{1+\theta} h_{0t} \\ h_{0t} &:= 0 \end{aligned} \tag{6.9}$$

This is Step 2.

Finally, if there exists an asset $j^\star$ and set $\mathcal{J}$ is non-empty, then take every asset $i \in \mathcal{J}$, sell $R_{it}h^+_{i(t-1)}$ units and with the resulting cash buy $\frac{1-\theta}{1+\theta}\sum_{i\in\mathcal{J}} R_{it}h^+_{i(t-1)}$ units of asset $j^\star$ by setting:

$$\begin{aligned} y_{it} &:= R_{it}h^+_{i(t-1)},\ \forall i \in \mathcal{J} \\ x_{j^\star t} &:= x_{j^\star t} + \frac{1-\theta}{1+\theta}\sum_{i\in\mathcal{J}} R_{it}h^+_{i(t-1)} \end{aligned} \tag{6.10}$$

This is Step 3.

Algorithm 6.1 summarizes the proposed algorithm to solve problem (6.6).

Algorithm 6.1 Allocation Algorithm for Linear Approximation

Input: k_{it}, l_{it} $\forall i \in \mathcal{N}$, h_{0t}, $R_{it}h^+_{i(t-1)}$ $\forall i \in \mathcal{N} \cup \{0\}$

Step 0. Initialization:
assign risky assets to one of the non-overlapping categories of Figure 6.2
set $x_{it}, y_{it} := 0\ \forall i \in \mathcal{N}$
set $j^\star := \{\arg\max_{i\in\mathcal{N}} k_{it} :\ k_{it} > 0\}$
set $\mathcal{I} :=$ Sell-assets & $\left(R_{it}h^+_{i(t-1)} > 0\right)$
set $\mathcal{J} :=$ Sell-to-buy-assets & $\left(R_{it}h^+_{i(t-1)} > 0\right)$

Step 1. Sell Sell-assets and update cash:
if $\mathcal{I} \neq \emptyset$ **then**
  set $y_{it} := R_{it}h^+_{i(t-1)}\ \forall i \in \mathcal{I}$
  set $h_{0t} := h_{0t} + (1-\theta)\sum_{i\in\mathcal{I}} R_{it}h^+_{i(t-1)}$
end

Step 2. Buy with cash:
if $(j^\star \neq \emptyset)$ & $(h_{0t} > 0)$ **then**
  set $x_{j^\star t} := \frac{1}{1+\theta}h_{0t}$
  set $h_{0t} := 0$
end

Step 3. Perform sell-to-buy-transactions:
if $(j^\star \neq \emptyset)$ & $(\mathcal{J} \neq \emptyset)$ **then**

set $y_{it} := R_{it}h^+_{i(t-1)}\ \forall i \in \mathcal{J}$
set $x_{j^\star t} := x_{j^\star t} + \frac{1-\theta}{1+\theta}\sum_{i\in\mathcal{J}} R_{it}h^+_{i(t-1)}$
end

Output: $x_{it}, y_{it}\ \forall i \in \mathcal{N}$

Theorem 6.2.2. *Algorithm 6.1 solves problem (6.6) optimally.*

Proof

We begin this proof by setting all variables to zero and examining the first N constraints of problem (6.6) which for every asset $i \in \mathcal{N}$ give us:

$$-x_{it} + y_{it} \leq R_{it}h^+_{i(t-1)}$$

In Lemma 4.3.1, we showed that we will never simultaneously sell and buy the same risky asset, i.e. we will have either $x^\star_{it} = 0$ or $y^\star_{it} = 0$, or $x^\star_{it} = y^\star_{it} = 0$. Looking at (6.2), we notice that setting $x_{it} = 0$ implies $y_{it} \leq R_{it}h^+_{i(t-1)}$, while the latter two cases are redundant. Therefore, for every $i \in \mathcal{N}$ inequality $-x_{it} + y_{it} \leq R_{it}h^+_{i(t-1)}$ can be replaced by $y_{it} \leq R_{it}h^+_{i(t-1)}$ and problem (6.6) becomes:

$$\left.\begin{aligned} \max_{(\boldsymbol{x}_t,\boldsymbol{y}_t)} \quad & \sum_{i=1}^{N} k_{it}x_{it} + \sum_{i=1}^{N} l_{it}y_{it} + \sum_{i=0}^{N} u_{it}R_{it}h^+_{i(t-1)} \\ \text{s.t.} \quad & y_{it} \leq R_{it}h^+_{i(t-1)},\ \ i \in \mathcal{N} \\ & \sum_{i=1}^{N} x_{it} \leq \frac{1}{1+\theta}R_{0t}h^+_{0(t-1)} + \frac{1-\theta}{1+\theta}\sum_{i=1}^{N} y_{it} \\ & x_{it}, y_{it} \geq 0,\ \ i \in \mathcal{N} \end{aligned}\right\} \tag{6.11}$$

In problem (6.11), every y_{it} is constrained from above by $R_{it}h^+_{i(t-1)}$. Thus, for every y_{it} with $l_{it} > 0$ as well as $R_{it}h^+_{i(t-1)} > 0$ the optimal decision is to set it to its upper bound. That is, for every $i \in \mathcal{I}$ the optimal decisions are $x^\star_{it} = 0$ and $y^\star_{it} = R_{it}h^+_{i(t-1)}$. After we fix x_{it} and y_{it} for every $i \in \mathcal{I}$, problem (6.11) becomes:

$$\left.\begin{aligned}\max_{(\boldsymbol{x}_t,\boldsymbol{y}_t)} \quad & \sum_{i\in\mathcal{N}\setminus\mathcal{I}} k_{it}x_{it} + \sum_{i\in\mathcal{N}\setminus\mathcal{I}} l_{it}y_{it} + \sum_{i\in\mathcal{I}} l_{it}R_{it}h^{+}_{i(t-1)} + \sum_{i=0}^{N} u_{it}R_{it}h^{+}_{i(t-1)} \\ \text{s.t.} \quad & y_{it} \leq R_{it}h^{+}_{i(t-1)}, \;\; i \in \mathcal{N}\setminus\mathcal{I} \\ & \sum_{i\in\mathcal{N}\setminus\mathcal{I}} x_{it} \leq \frac{1}{1+\theta}R_{0t}h^{+}_{0(t-1)} + \frac{1-\theta}{1+\theta}\sum_{i\in\mathcal{I}} R_{it}h^{+}_{i(t-1)} + \frac{1-\theta}{1+\theta}\sum_{i\in\mathcal{N}\setminus\mathcal{I}} y_{it} \\ & x_{it}, y_{it} \geq 0, \;\; i \in \mathcal{N}\end{aligned}\right\} \tag{6.12}$$

Looking now at the budget constraint of problem (6.12), we notice that x_{it}'s are constrained from above by constant $\frac{1}{1+\theta}R_{0t}h^{+}_{0(t-1)} + \frac{1-\theta}{1+\theta}\sum_{i\in\mathcal{I}} R_{it}h^{+}_{i(t-1)}$ as well as by y_{it}'s, where every incremental increase in a y_{it} increases the upper bound of the sum of x_{it}'s by $\frac{1-\theta}{1+\theta}$ units.

Thus, a positive contribution in the objective can be achieved if we increase $x_{j^\star t}$, where $j^\star$ is the asset with the highest positive buying slope, by the constant amount of the budget constraint. That is, initially for asset $j^\star$ the optimal decisions are $y_{j^\star t} = 0$ and $x_{j^\star t} = \frac{1}{1+\theta}R_{0t}h^{+}_{0(t-1)} + \frac{1-\theta}{1+\theta}\sum_{i\in\mathcal{I}} R_{it}h^{+}_{i(t-1)}$.

Further, a positive contribution in the objective can be achieved if we continue simultaneously increasing $x_{j^\star t}$ and y_{it} of any other asset $i \notin \mathcal{I}$ if $R_{it}h^{+}_{i(t-1)} > 0$ and $\frac{1-\theta}{1+\theta}k_{j^\star t} + l_{it} > 0$. Thus, for every $i \in \mathcal{J}$ the optimal decision is to sell as much as possible, i.e we will have $x^\star_{it} = 0$ and $y^\star_{it} = R_{it}h^{+}_{i(t-1)}$, and for asset $j^\star$ the optimal decision is to continue buying it until it becomes $x^\star_{j^\star t} = \frac{1}{1+\theta}R_{0t}h^{+}_{0(t-1)} + \frac{1-\theta}{1+\theta}\sum_{i\in\mathcal{I}\cup\mathcal{J}} R_{it}h^{+}_{i(t-1)}$. Note that if we do not consider simultaneously increasing $x_{j^\star t}$ and y_{it}'s, then we would end up with a suboptimal solution for problem (6.6).

For every other $i \in \mathcal{N} \setminus \{\mathcal{I}\cup\mathcal{J}\}$ we will have $y^\star_{it} = 0$, while for every other $i \in \mathcal{N} \setminus \{j^\star\}$ we will have $x^\star_{it} = 0$. Note that if all $k'_{it}s$ are negative, then there exists no asset $j^\star$ and we will have $x^\star_{it} = 0$ for every $i \in \mathcal{N}$. ■

Considering the above, for $t = 1, 2, \ldots, T-1$ the optimal decisions of problem (6.6) are given by:

$$y^\star_{it} = \begin{cases} R_{it}h^{+}_{i(t-1)}, & \text{if } i \in \mathcal{I}\cup\mathcal{J} \\ 0, & \text{if } i \in \mathcal{N}\setminus(\mathcal{I}\cup\mathcal{J}) \end{cases} \tag{6.13}$$

$$x^\star_{it} = \begin{cases} \frac{R_{0t}h^{+}_{0(t-1)} + (1-\theta)\sum_{i\in\mathcal{I}\cup\mathcal{J}} R_{it}h^{+}_{i(t-1)}}{1+\theta}, & \text{if } i = j^\star \\ 0, & \text{if } i \neq j^\star \end{cases} \tag{6.14}$$

Note that if we replace $R_{it}h^+_{i(t-1)}$ with h_{i0} in (6.13) and (6.14) we will obtain the optimal decisions for time $t = 0$.

We will now provide an example to illustrate how we perform buying and selling in Algorithm 6.1.

Example 6.1. Suppose we have three stocks with buying slopes, selling slopes, current holdings and cash as shown in Figure 6.3.

Figure 6.4 shows the updated wealth in all assets after applying Step 1 of Algorithm 6.1, where, due to the positive selling slope of stock 3, we sell h_{3t} units of stock 3. This increases cash by $(1-\theta)h_{3t}$ units.

Figure 6.5 shows the updated wealth in all assets after applying Step 2 of Algorithm 6.1, where we use all cash to buy stock 1 (this is the asset with the highest positive buying slope). This increases the holdings of stock 1 by $\frac{h_{0t}+(1-\theta)h_{3t}}{1+\theta}$ units.

Finally, figure 6.6 shows the updated wealth in all assets after applying Step 3 of Algorithm 6.1, where, assuming that $k_{1t} + \frac{1-\theta}{1+\theta}l_{2t} > 0$, we sell h_{2t} units of stock 2 to buy $\frac{1-\theta}{1+\theta}h_{2t}$ units of stock 1.

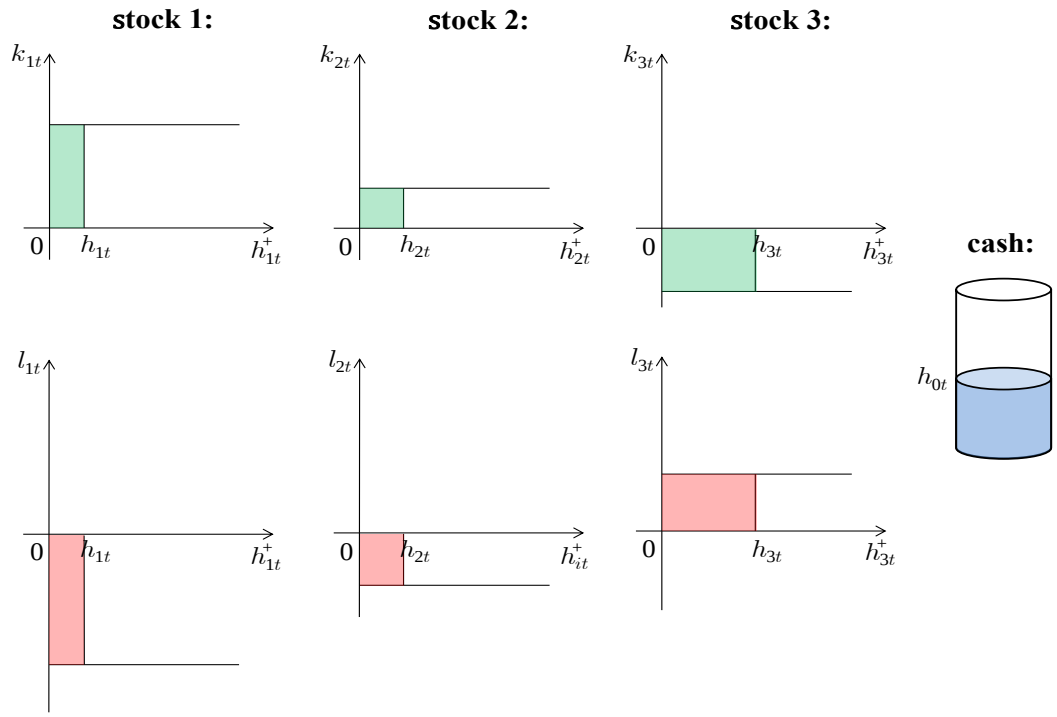


Figure 6.3: Buying and selling slopes vs current holdings before allocation

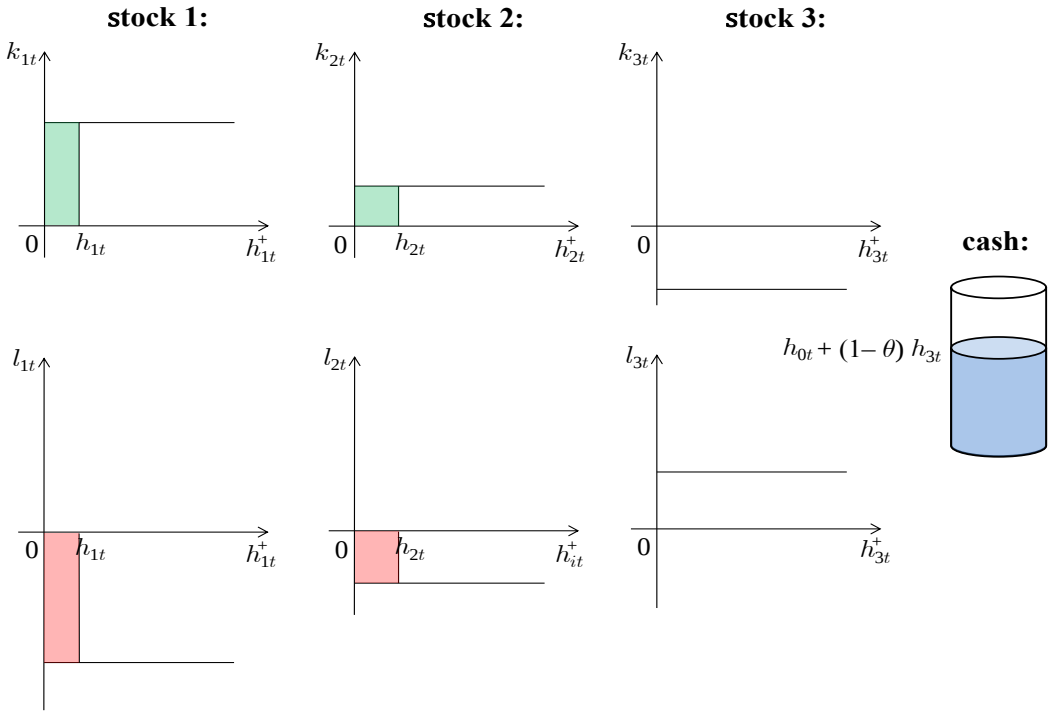


Figure 6.4: Step 1. Sell stock 3 and update cash

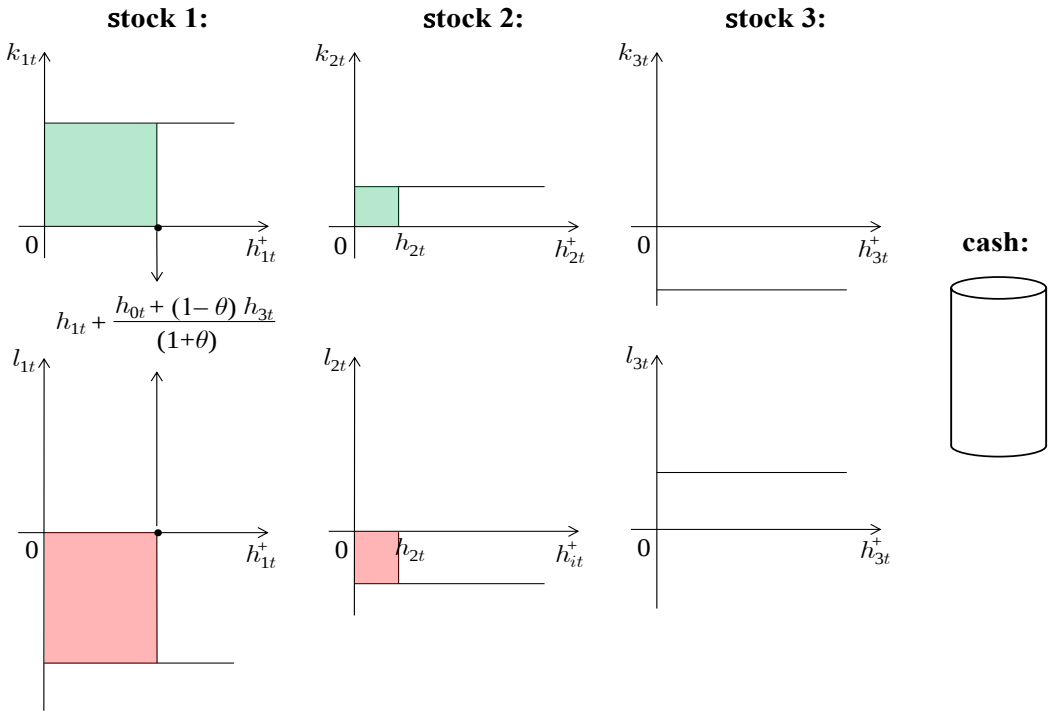


Figure 6.5: Step 2. Buy stock 1 with cash

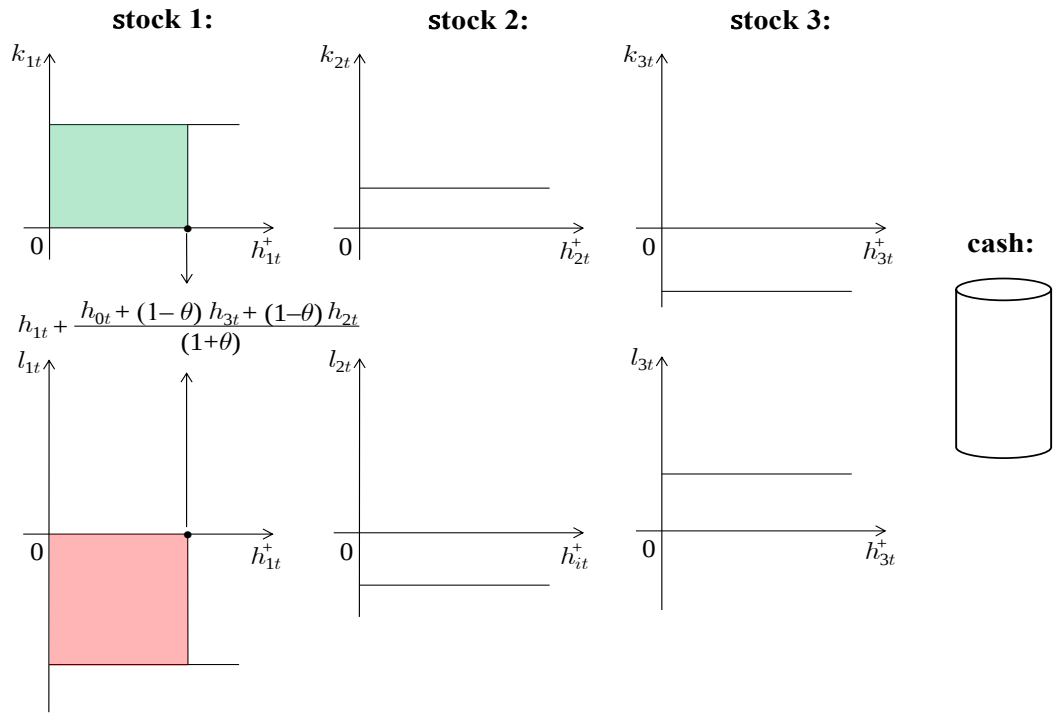


Figure 6.6: Step 3. Sell stock 2 and buy stock 1

Complexity

The linear program (6.6), which we solve with Algorithm 6.1, has a total of $2N$ decision variables and $N+1$ constraints excluding the $2N$ non-negativity constraints.

In Algorithm 6.1, it takes us $O(N)$ time to find the assets that have positive selling slopes, $O(N)$ time to find the best buying asset $j^\star$ and $O(N)$ time to detect the Sell-to-buy-assets. This gives us a time complexity of $O(N)$ for Algorithm 6.1. For S T-period realizations of the stochastic process of random returns, the above gives us a polynomial time complexity of $S \cdot T \cdot O(N) + \sum_{s=1}^{S} slogs$ for the LADP algorithm (see section 4.3 for a discussion on the complexity of the General ADP algorithm).

6.3 The Subproblem of LADP-UB

As in section 6.2, in this section we write the linear programming formulation of each subproblem of LADP-UB as a linear program and solve it.

Linear Programming Formulation of the Subproblem of LADP-UB

In each period t, we impose upper bounds on how much we can hold in each risky asset by requiring that:

$$h_{it}^{+} \leq w_0, \;\; i \in \mathcal{N} \tag{6.15}$$

where $w_0 = \alpha \sum_{i=0}^{N} h_{i0}$ and simply tells us that in every risky asset we cannot have more than a fraction α of the total initial wealth and parameter α takes values in $[0, \infty)$. Note that for very high values of parameter α, constraints of type (6.15) would be of no use as they would allow us to have in each risky asset as much wealth as we want. Further, note that for very low values of parameter α, we would end up keeping most of our wealth, if not all, tied up in cash. To avoid the above two extremes, in our experiments in chapter 8 we select values in range $(0, 1)$ that are neither too high nor too low.

Regarding cash, we leave it free assuming that we can have as much cash as we want.

If we substitute in (6.15) $h_{it}^{+} = R_{it} h_{i(t-1)}^{+} + x_{it} - y_{it}$ from (4.5), then (6.15) becomes:

$$x_{it} - y_{it} \leq w_0 - R_{it} h_{i(t-1)}^{+}, \;\; i \in \mathcal{N} \tag{6.16}$$

We will see later on that constraint (6.16) forces us to sell those assets for which we hold more than the upper bound w_0, i.e. those assets for which we have $R_{it} h_{i(t-1)}^{+} > w_0$, and allows us to buy those assets for which we hold less than the upper bound w_0, i.e. those assets for which we have $h_{it} = R_{it} h_{i(t-1)}^{+} < w_0$.

After introducing constraints (6.16), what changes in the dynamic programming formulation is action space (4.14), which is now expanded to include the new constraints and is given by:

$$\mathcal{A}_t = \{(\boldsymbol{x}_t, \boldsymbol{y}_t) : \; (4.11) - (4.13), (6.16) \text{ hold}\} \tag{6.17}$$

Using action space (6.17), the subproblem of LADP-UB is given by:

$$\left.\begin{array}{ll} \max\limits_{(\boldsymbol{x}_t,\boldsymbol{y}_t)} & \sum\limits_{i=1}^{N} k_{it}x_{it} + \sum\limits_{i=1}^{N} l_{it}y_{it} + \sum\limits_{i=0}^{N} u_{it}R_{it}h^{+}_{i(t-1)} \\ \text{s.t.} & -x_{it} + y_{it} \leq R_{it}h^{+}_{i(t-1)},\ \ i \in \mathcal{N} \\ & x_{it} - y_{it} \leq w_0 - R_{it}h^{+}_{i(t-1)},\ \ i \in \mathcal{N} \\ & \sum\limits_{i=1}^{N} x_{it} \leq \dfrac{1}{1+\theta}R_{0t}h^{+}_{0(t-1)} + \dfrac{1-\theta}{1+\theta}\sum\limits_{i=1}^{N} y_{it} \\ & x_{it}, y_{it} \geq 0,\ \ i \in \mathcal{N} \end{array}\right\} \tag{6.18}$$

where $k_{it} = u_{it} - (1+\theta)u_{0t}$ and $l_{it} = -u_{it} + (1-\theta)u_{0t}$ were called in chapter 6 respectively the buying and selling slopes.

Buying and Selling Slopes

Slopes k_{it} and l_{it} satisfy the properties of Theorem 6.2.1 and as in section 6.2 above, the relationships between the buying and selling slopes allow the classification of the assets into one of the categories (sets) of Figure 6.2 which are the Buy-assets, the Sell-assets, the Neutral-assets and Cash. Further, as in section 6.2, we will denote the best buying asset with $j^\star$ and will be given by $j^\star = \{\arg\max_{i\in\mathcal{N}} k_{it} :\ (k_{it} > 0)\ \&\,(h_{it} <\ w_0)\}$. The main difference from the best buying asset $j^\star$ of the simple linear approximation is that here we need to change to a new asset $j^\star$ whenever its holdings become equal to upper bound w_0.

Solution to the Subproblem of LADP-UB

We are now ready to solve problem (6.18).

Using $h_{it} = R_{it}h^{+}_{i(t-1)}$, we will need the following sets:

- $\mathcal{I}$ = Sell-assets $\&\ (h_{it} > 0)$: As in chapter 6, this set contains all Sell-assets that have positive holdings (these assets will be sold).
- $\mathcal{F} = (\text{Buy-assets} \cup \text{Neutral-assets})\,\&\,(h_{it} > w_0)$: All Buy- and Neutral-assets whose current holdings exceed upper bound w_0. As we will see, these assets are sold in order to satisfy the new constraints (6.16) and will be called the *Forced-to-sell-assets*. Note that after fixing the holdings of these assets, some of them may become Sell-to-buy-assets so we might need to continue selling them.

We propose the following algorithm:

Begin with setting $x_{it}, y_{it} := 0$ for every asset $i \in \mathcal{N}$, assign risky assets to one of the non-overlapping categories of Figure 6.2, initialize sets $\mathcal{I}$ and $\mathcal{F}$ and pick asset $j^\star$ by setting:

$$j^\star := \left\{ \arg\max_{i \in \mathcal{N}} k_{it} : \ (k_{it} > 0) \,\&\, (h_{it} < w_0) \right\} \tag{6.19}$$

This is Step 0.

If sets $\mathcal{I}$ and $\mathcal{F}$ are non-empty, then take all assets in sets $\mathcal{I}$ and $\mathcal{F}$, sell respectively h_{it} and $h_{it} - w_0$ units and update cash by setting:

$$\begin{aligned} y_{it} &:= h_{it}, \ \forall i \in \mathcal{I} \\ y_{it} &:= h_{it} - w_0, \ \forall i \in \mathcal{F} \\ h_{0t} &:= h_{0t} + (1-\theta) \sum_{i \in \mathcal{I}} h_{it} + (1-\theta) \sum_{i \in \mathcal{F}} (h_{it} - w_0) \end{aligned} \tag{6.20}$$

This is Step 1.

Next, after Step 1, if there exist an asset $j^\star$ and cash, then use cash to buy $\min\left(\frac{1}{1+\theta} h_{0t}, w_0 - h_{j^\star t}\right)$ units of asset $j^\star$ and update cash by setting:

$$\begin{aligned} x_{j^\star t} &:= \min\left(\frac{1}{1+\theta} h_{0t}, w_0 - h_{j^\star t} \right) \\ h_{0t} &:= h_{0t} - (1+\theta) \min\left(\frac{1}{1+\theta} h_{0t}, w_0 - h_{j^\star t} \right) \end{aligned} \tag{6.21}$$

Every time asset $j^\star$ is filled up, i.e. whenever $\min\left(\frac{1}{1+\theta} h_{0t}, w_0 - h_{j^\star t}\right) = w_0 - h_{j^\star t}$, pick another asset $j^\star$ using (6.19) and continue buying the new asset $j^\star$, if any.

This is Step 2 and it terminates either when there exists no other asset $j^\star$ or when we run out of cash.

Then, after Step 2 and given that there exists an asset $j^\star$, if there exist Buy- or Neutral-assets with positive holdings such that amount $\frac{1-\theta}{1+\theta} k_{j^\star t} + l_{it}$ is positive, then among these assets pick the one that maximizes this amount by setting:

$$i^\star := \left\{ \arg\max_{i \in \mathcal{N}} l_{it} : \ (l_{it} \leq 0) \ \& \ (h_{it} > 0) \ \& \ \left(\frac{1-\theta}{1+\theta} k_{j^\star t} + l_{it} > 0 \right) \right\}, \tag{6.22}$$

and simultaneously increase assets $i^\star$ and $j^\star$ by setting:

$$
\begin{aligned}
y_{i^\star t} &:= \min\left(h_{i^\star t}, \frac{1+\theta}{1-\theta}\left(w_0 - h_{j^\star t}\right)\right), \\
x_{j^\star t} &:= x_{j^\star t} + \frac{1-\theta}{1+\theta}\min\left(h_{i^\star t}, \frac{1+\theta}{1-\theta}\left(w_0 - h_{j^\star t}\right)\right), \\
h_{i^\star t} &:= h_{i^\star t} - x_{i^\star t}, \\
h_{j^\star t} &:= h_{j^\star t} + y_{j^\star t}
\end{aligned}
\tag{6.23}
$$

If the minimization in (6.23) is given by the first term (this is when we run out of holdings in asset $i^\star$), then pick another asset $i^\star$ using (6.22) and continue simultaneously selling the new asset $i^\star$ and buying asset $j^\star$ using (6.23). If the minimization in (6.23) is given by the second term (this is when asset $j^\star$ is filled up), then pick another asset $j^\star$ using (6.19) and continue simultaneously selling asset $i^\star$ and buying the new asset $j^\star$ using (6.23).

This is Step 3 and it terminates when there exists no other asset $i^\star$ or $j^\star$.

Algorithm 6.2 summarizes the proposed algorithm to solve problem (6.18).

Algorithm 6.2 Allocation Algorithm for Linear Approximation with Strict Control of Flows

---

Input: w_0, k_{it}, l_{it} $\forall i \in \mathcal{N}$, h_{it} $\forall i \in \mathcal{N} \cup \{0\}$

Step 0. Initialization:
assign risky assets to one of the non-overlapping categories of Figure 6.2
set $x_{it}, y_{it} := 0$ $\forall i \in \mathcal{N}$
set $\mathcal{I} :=$ Sell-assets $\&$ $(h_{it} > 0)$
set $\mathcal{F} :=$ (Buy-assets $\cup$ Neutral-assets) $\&$ $(h_{it} > w_0)$
set $j^\star := \{\arg\max_{i \in \mathcal{N}} k_{it} :\ (k_{it} > 0) \,\&\, (h_{it} < w_0)\}$

Step 1. Sell Sell-assets and Forced-to-sell-assets and update cash:
if $(\mathcal{I} \neq \emptyset) \vee (\mathcal{J} \neq \emptyset)$ **then**
  set $y_{it} := h_{it}$ $\forall i \in \mathcal{I}$
  set $y_{it} := h_{it} - w_0$ $\forall i \in \mathcal{F}$
  set $h_{0t} := h_{0t} + (1-\theta)\sum_{i \in \mathcal{I}} h_{it} + (1-\theta)\sum_{i \in \mathcal{F}} h_{it}$
end

Step 2. Buy with cash:
while $(j^\star \neq \emptyset)$ $\&$ $(h_{0t} > 0)$ **do**
  set $x_{j^\star t} := \min\left(\frac{1}{1+\theta} h_{0t}, w_0 - h_{j^\star t}\right)$
  set $h_{0t} := h_{0t} - (1+\theta)\min\left(\frac{1}{1+\theta} h_{0t}, w_0 - h_{j^\star t}\right)$
  if $\min\left(\frac{1}{1+\theta} h_{0t}, w_0 - h_{j^\star t}\right) = w_0 - h_{j^\star t}$ **then**

---

set $j^\star := \{\arg\max_{i\in\mathcal{N}} k_{it} : \ (k_{it} > 0) \,\&\, (h_{it} < w_0)\}$
set $h_{j^\star t} := w_0$
else
set $h_{j^\star t} := h_{j^\star t} + \frac{1}{1+\theta} h_{0t}$
set $h_{0t} := 0$
end
end

Step 3. Perform sell-to-buy-transactions:
set $i^\star := \left\{\arg\max_{i\in\mathcal{N}} l_{it} : \ (l_{it} \le 0) \ \&\ (h_{it} > 0) \ \&\ \left(\frac{1-\theta}{1+\theta} k_{j^\star t} + l_{it} > 0\right)\right\}$
while $(j^\star \neq \emptyset) \ \&\ (i^\star \neq \emptyset)$ **do**
set $y_{i^\star t} := y_{i^\star t} + \min\left(h_{i^\star t}, \frac{1+\theta}{1-\theta}(w_0 - h_{j^\star t})\right)$
set $x_{j^\star t} := x_{j^\star t} + \frac{1-\theta}{1+\theta} \min\left(h_{i^\star t}, \frac{1+\theta}{1-\theta}(w_0 - h_{j^\star t})\right)$
set $h_{i^\star t} := h_{i^\star t} - y_{i^\star t}$
set $h_{j^\star t} := h_{j^\star t} + x_{j^\star t}$
if $\min\left(h_{i^\star t}, \frac{1+\theta}{1-\theta}(w_0 - h_{j^\star t})\right) = h_{i^\star t}$ **then**
set $i^\star := \left\{\arg\max_{i\in\mathcal{N}} l_{it} : \ (l_{it} \le 0) \ \&\ (h_{it} > 0) \ \&\ \left(\frac{1-\theta}{1+\theta} k_{j^\star t} + l_{it} > 0\right)\right\}$
else
set $j^\star := \{\arg\max_{i\in\mathcal{N}} k_{it} : \ (k_{it} > 0) \,\&\, (h_{it} < w_0)\}$
end
end

Output: $x_{it}, y_{it} \ \forall i \in \mathcal{N}$

Theorem 6.3.1. *Algorithm 6.2 solves problem (6.18) optimally.*

Proof

As in the proof of Theorem 6.2.2, we begin with setting all variables to zero and replacing $-x_{it} + y_{it} \le h_{it}$ with $y_{it} \le h_{it}$ for every $i \in \mathcal{N}$, where $h_{it} = R_{it} h^+_{i(t-1)}$. Further, recall that for every $i \in \mathcal{I}$ (these are the Sell-assets with positive holdings) the optimal decisions are $y^\star_{it} = h_{it}$ and $x^\star_{it} = 0$.

Looking now at the new constraints: $x_{it} - y_{it} \le w_0 - h_{it}$, we examine the following cases:

1. $w_0 < h_{it}$: Setting $x_{it} = 0$ gives us $y_{it} \ge h_{it} - w_0$, while cases $y_{it} = 0$ and $x_{it} = y_{it} = 0$ lead to contradictions. That is, for each asset i whose current holdings exceed w_0 we must sell at least $h_{it} - w_0$ units. Therefore, for every asset $i \in \mathcal{F}$ inequality $x_{it} - y_{it} \le w_0 - h_{it}$ can be replaced by $x_{it} = 0$ and $y_{it} \ge h_{it} - w_0$.

2. $w_0 \geq h_{it}$: Setting $y_{it} = 0$ gives us $x_{it} \leq w_0 - h_{it}$, while cases $x_{it} = 0$ and $x_{it} = y_{it} = 0$ are redundant. That is, when current holdings are less than w_0 we can buy at most $w_0 - h_{it}$. Therefore, for every $i \in \mathcal{N} \setminus \mathcal{F}$, inequality $x_{it} - y_{it} \leq w_0 - h_{it}$ can be replaced by $x_{it} \leq w_0 - h_{it}$.

Considering the above and after fixing x_{it} and y_{it} for every $i \in \mathcal{I} \cup \mathcal{F}$, problem (6.18) becomes:

$$\left.\begin{aligned} \max_{(\mathbf{x}_t, \mathbf{y}_t)} \quad & \sum_{i \in \mathcal{N} \setminus (\mathcal{I} \cup \mathcal{F})} k_{it} x_{it} + \sum_{i \in \mathcal{N} \setminus \mathcal{I}} l_{it} y_{it} + \sum_{i \in \mathcal{I}} l_{it} h_{it} + \sum_{i=0}^{N} u_{it} h_{it} \\ \text{s.t.} \quad & \sum_{i \in \mathcal{N} \setminus (\mathcal{I} \cup \mathcal{F})} x_{it} \leq \frac{1}{1+\theta} h_{0t} + \frac{1-\theta}{1+\theta} \sum_{i \in \mathcal{I}} h_{it} + \frac{1-\theta}{1+\theta} \sum_{i \in \mathcal{N} \setminus \mathcal{I}} y_{it} \\ & h_{it} - w_0 \leq y_{it} \leq h_{it}, \;\; i \in \mathcal{F} \\ & 0 \leq y_{it} \leq h_{it}, \;\; i \in \mathcal{N} \setminus (\mathcal{I} \cup \mathcal{F}) \\ & 0 \leq x_{it} \leq w_0 - h_{it}, \;\; i \in \mathcal{N} \setminus (\mathcal{I} \cup \mathcal{F}) \end{aligned}\right\} \tag{6.24}$$

In problem (6.24), for every $i \in \mathcal{F}$ we replace variable y_{it} with a new variable $\bar{y}_{it}$ so that the left bound of the new variable becomes zero, i.e. we set $y_{it} = \bar{y}_{it} + h_{it} - w_0$, and rewrite problem (6.24) as follows:

$$\left.\begin{aligned} \max_{(\mathbf{x}_t, \mathbf{y}_t)} \quad & \sum_{i \in \mathcal{N} \setminus (\mathcal{I} \cup \mathcal{F})} k_{it} x_{it} + \sum_{i \in \mathcal{N} \setminus (\mathcal{I} \cup \mathcal{F})} l_{it} y_{it} + \sum_{i \in \mathcal{F}} l_{it} \bar{y}_{it} + \sum_{i \in \mathcal{I}} l_{it} h_{it} \\ & + \sum_{i \in \mathcal{F}} l_{it} \left(h_{it} - w_0\right) + \sum_{i=0}^{N} u_{it} h_{it} \\ \text{s.t.} \quad & \sum_{i \in \mathcal{N} \setminus (\mathcal{I} \cup \mathcal{F})} x_{it} \leq \frac{1}{1+\theta} h_{0t} + \frac{1-\theta}{1+\theta} \left(\sum_{i \in \mathcal{I}} h_{it} + \sum_{i \in \mathcal{F}} \left(h_{it} - w_0\right) \right) \\ & + \frac{1-\theta}{1+\theta} \sum_{i \in \mathcal{N} \setminus (\mathcal{I} \cup \mathcal{F})} y_{it} + \frac{1-\theta}{1+\theta} \sum_{i \in \mathcal{F}} \bar{y}_{it} \\ & 0 \leq \bar{y}_{it} \leq w_0, \;\; i \in \mathcal{F} \\ & 0 \leq y_{it} \leq h_{it}, \;\; i \in \mathcal{N} \setminus (\mathcal{I} \cup \mathcal{F}) \\ & 0 \leq x_{it} \leq w_0 - h_{it}, \;\; i \in \mathcal{N} \setminus (\mathcal{I} \cup \mathcal{F}) \end{aligned}\right\} \tag{6.25}$$

Looking now at the budget constraint of problem (6.25), we notice that x_{it}'s are constrained from above by constant $\frac{1}{1+\theta} h_{0t} + \frac{1-\theta}{1+\theta} \left(\sum_{i \in \mathcal{I}} h_{it} + \sum_{i \in \mathcal{F}} \left(h_{it} - w_0\right) \right)$ as

well as by y_{it}'s and $\bar{y}_{it}$'s, where every incremental increase in a y_{it} or $\bar{y}_{it}$ increases the upper bound of the sum of x_{it}'s by $\frac{1-\theta}{1+\theta}$ units.

Thus, a positive contribution in the objective can be achieved if we increase $x_{j^\star t}$, where $j^\star$ is the asset with the highest positive buying slope, by the constant amount of the budget constraint. However, due to $x_{j^\star t} \leq w_0 - h_{j^\star t}$, initially for asset $j^\star$ the optimal decisions are $y_{j^\star t} = 0$ and $x_{j^\star t} = \min\left(\frac{1}{1+\theta}h_{0t} + \frac{1-\theta}{1+\theta}\left(\sum_{i\in\mathcal{I}} h_{it} + \sum_{i\in\mathcal{F}}\left(h_{it} - w_0\right)\right), w_0 - h_{j^\star t}\right)$. If the minimum buying amount comes from asset $j^\star$ and there exists another asset $j^\star$, then the optimal decision is to continue buying the new asset $j^\star$ in the above manner as this will further increase the objective. Buying terminates either when we have used up the resources of the budget constraint or when there exists no other asset $j^\star$.

Further, a positive contribution in the objective can be achieved if we continue simultaneously increasing $x_{j^\star t}$ and a $y_{i^\star t}$ (or a $\bar{y}_{i^\star t}$), where $i^\star$ is the asset $i \in \mathcal{N} \setminus \mathcal{I}$ with the highest selling slope and positive holdings, if $k_{j^\star t} + \frac{1-\theta}{1+\theta} l_{i^\star t} > 0$. However, due to $x_{j^\star t} \leq w_0 - h_{j^\star t}$, initially the optimal decision for asset $i^\star$ is to increase $y_{i^\star t}$ (or $\bar{y}_{i^\star t}$) by $\min\left(h_{i^\star t}, \frac{1+\theta}{1-\theta}\left(w_0 - h_{j^\star t}\right)\right)$, while for asset $j^\star$ initially the optimal decision is to increase $x_{j^\star t}$ by $\frac{1-\theta}{1+\theta}\min\left(h_{i^\star t}, \frac{1+\theta}{1-\theta}\left(w_0 - h_{j^\star t}\right)\right)$ (note that asset $j^\star$ might have been used previously to buy with cash). If the minimum selling amount comes from asset $i^\star$ and there exists another asset $i^\star$, then the optimal decision is to continue selling the new asset $i^\star$ and buying asset $j^\star$ in the above manner as this will further increase the objective. If the minimum selling amount comes from asset $j^\star$ and there exists another asset $j^\star$, then the optimal decision is to continue selling asset $i^\star$ and buying the new asset $j^\star$ in the above manner as this will further increase the objective. Note that if we do not consider simultaneously increasing $x_{j^\star t}$ and $y_{i^\star t}$'s (or $\tilde{y}_{i^\star t}$'s), then we would end up with a suboptimal solution for problem (6.25).

Note that if all $k'_{it}s$ are negative, then there exists no asset $j^\star$ and we will have $x^\star_{it} = 0$ for every $i \in \mathcal{N}$. Further, note that for every $i \in \mathcal{F}$ we will have $y^\star_{it} = \tilde{y}^\star_{it} + h_{it} - w_0$. ■

Considering the above, after solving problem (6.18), the following new sets arise:

1. $\mathcal{C}$: All assets $j^\star$ that were used for buying. We assume that all assets in this set are sequenced in the order in which they were used, i.e. in a decreasing order in terms of their buying slopes, and asset c is the last asset in the set.

2. $\mathcal{D}$: All assets $i^\star$ that were sold in order to buy assets $j^\star$ and here will be simply called the Sell-to-buy-assets. We assume that all assets in this set are sequenced in the order in which they were used, i.e. in a decreasing order in terms of their selling slopes, and asset d is the last asset in the set. Note that this set may contain assets that were previously Forced-to-sell-assets.

3. $\bar{\mathcal{F}}$: All Forced-to-sell-assets from which we sold exactly $h_{it} - w_0$ units. Note that $\mathcal{D} \cap \bar{\mathcal{F}} = \emptyset$.

Given the above sets, we will now provide a sketch of the optimal solution of problem (6.18) which, after solving problem (6.18), breaks down into one of the following cases:

1. **Case 1**: No buying occurred, i.e. $\mathcal{C} = \emptyset$, either because there was no asset $j^\star$ or because we had no cash resources (including cash that comes from Sell-assets and Forced-to-sell-assets) and no Sell-to-buy-assets. The optimal decisions are given by:

$$y_{it}^\star = \begin{cases} h_{it}, & \text{if } i \in \mathcal{I} \\ h_{it} - w_0, & \text{if } i \in \bar{\mathcal{F}} \\ 0, & \text{if } i \notin \mathcal{I} \cup \bar{\mathcal{F}} \end{cases} \tag{6.26}$$

$$x_{it}^\star = 0, \ \ \forall i \in \mathcal{N} \tag{6.27}$$

   Note that if buying did not occur because we had no cash resources, then sets $\mathcal{I}$ and $\bar{\mathcal{F}}$ are empty.

2. **Case 2**: Buying occurred, i.e. $\mathcal{C} \neq \emptyset$, but we stopped buying because:

   (a) **Sub-case 2.1**: we filled up all Buy-assets in set $\mathcal{C}$ either only with cash resources (including cash that comes from Sell-assets and Forced-to-sell-assets) or with cash resources and Sell-to-buy-assets. The optimal decisions are given by:

$$y_{it}^{\star} = \begin{cases} h_{it}, & \text{if } i \in (\mathcal{I} \cup \mathcal{D}) \setminus \{d\} \\ h_{it} - w_0, & \text{if } i \in \bar{\mathcal{F}} \\ \frac{1+\theta}{1-\theta} \sum_{i \in \mathcal{C}} (w_0 - h_{it}) - \frac{1}{1-\theta} h_{0t} & \\ - \sum_{i \in (\mathcal{I} \cup \mathcal{D}) \setminus \{d\}} h_{it} - \sum_{i \in \bar{\mathcal{F}}} (h_{it} - w_0), & \text{if } i = d \\ 0, & \text{if } i \notin \mathcal{I} \cup \bar{\mathcal{F}} \cup \mathcal{D} \end{cases} \tag{6.28}$$

$$x_{it}^{\star} = \begin{cases} w_0 - h_{it}, & \text{if } i \in \mathcal{C} \\ 0, & \text{if } i \notin \mathcal{C} \end{cases} \tag{6.29}$$

Variable $y_{dt}^{\star}$ in (6.28) is given by how much we bought in total using the Buy-assets of set $\mathcal{C}$ minus the cash resources that we used for buying and which come from cash, Sell-assets, Forced-to-sell-assets as well as from the other Sell-to-buy-assets of set $\mathcal{D}$ (where note that the total buying amount and cash are projected to asset d using respectively projection coefficients $\frac{1+\theta}{1-\theta}$ and $\frac{1}{1-\theta}$ of Table 6.2). Further, note that set $\mathcal{D}$ is empty and asset d does not exist if the Buy-assets of set $\mathcal{C}$ were filled up without using any Sell-to-buy-assets.

(b) **Sub-case 2.2**: we run out of resources which are either only cash (including cash that comes from Sell-assets and Forced-to-sell-assets) or cash and Sell-to-buy-assets. The optimal decisions are given by:

$$y_{it}^{\star} = \begin{cases} h_{it}, & \text{if } i \in \mathcal{I} \cup \mathcal{D} \\ h_{it} - w_0, & \text{if } i \in \bar{\mathcal{F}} \\ 0, & \text{if } i \notin \mathcal{I} \cup \bar{\mathcal{F}} \cup \mathcal{D} \end{cases} \tag{6.30}$$

$$x_{it}^{\star} = \begin{cases} w_0 - h_{it}, & \text{if } i \in \mathcal{C} \setminus \{c\} \\ \frac{1}{1+\theta} h_{0t} + \frac{1-\theta}{1+\theta} \sum_{i \in \mathcal{I} \cup \mathcal{D}} h_{it} + \frac{1-\theta}{1+\theta} \sum_{i \in \bar{\mathcal{F}}} & \\ (h_{it} - w_0) - \sum_{i \in \mathcal{C} \setminus \{c\}} (w_0 - h_{it}), & \text{if } i = c \\ 0, & \text{if } i \notin \mathcal{C} \end{cases} \tag{6.31}$$

Variable $x_{ct}^{\star}$ in (6.31) is given by the cash resources that we used for

buying and which come from cash, Sell-assets, Forced-to-sell-assets as well as from Sell-to-buy-assets minus how much we bought from the other Buy-assets of set $\mathcal{C}$ (where note that cash and the other resources are transformed to asset c using respectively transformation coefficients $\frac{1}{1+\theta}$ and $\frac{1-\theta}{1+\theta}$ of Table 6.1). Further, note that set $\mathcal{D}$ is empty if we stopped buying without using any Sell-to-buy-assets.

We will now provide an example to illustrate how we perform selling and buying using Algorithm 6.2.

Example 6.2. Suppose we have three stocks with buying slopes, selling slopes, holdings and cash as shown in figure 6.7, where we assume that in every stock we cannot have more than b units of wealth.

Figure 6.8 shows the updated holdings of the assets after applying Step 1 of Algorithm 6.2, where we sell Sell-asset stock 3 and Forced-to-sell-asset stock 2. This increases cash by $(1-\theta)h_{3t} + (1-\theta)(h_{2t} - b)$ (monetary) units.

Figure 6.9 shows the updated holdings of the assets after applying Step 2 of Algorithm 6.2, where we use all cash to buy stock 1 since this is the only Buy-asset that we can buy. This increases the holdings of stock 1 by $\frac{h_{0t}+(1-\theta)h_{3t}+(1-\theta)(h_{2t}-b)}{1+\theta}$ (monetary) units.

Lastly, figure 6.10 shows the updated holdings of the assets after applying Step 3 of Algorithm 6.2, where, assuming that $k_{1t} + \frac{1-\theta}{1+\theta} l_{2t} > 0$, we sell stock 2 until we fill up stock 1. This brings up the holdings of stock 1 to level b and decreases the holdings of stock 2 by $\frac{1+\theta}{1-\theta}\left(b - h_{1t} - \frac{h_{0t}+(1-\theta)h_{3t}+(1-\theta)(h_{2t}-b)}{1+\theta}\right)$ (monetary) units.

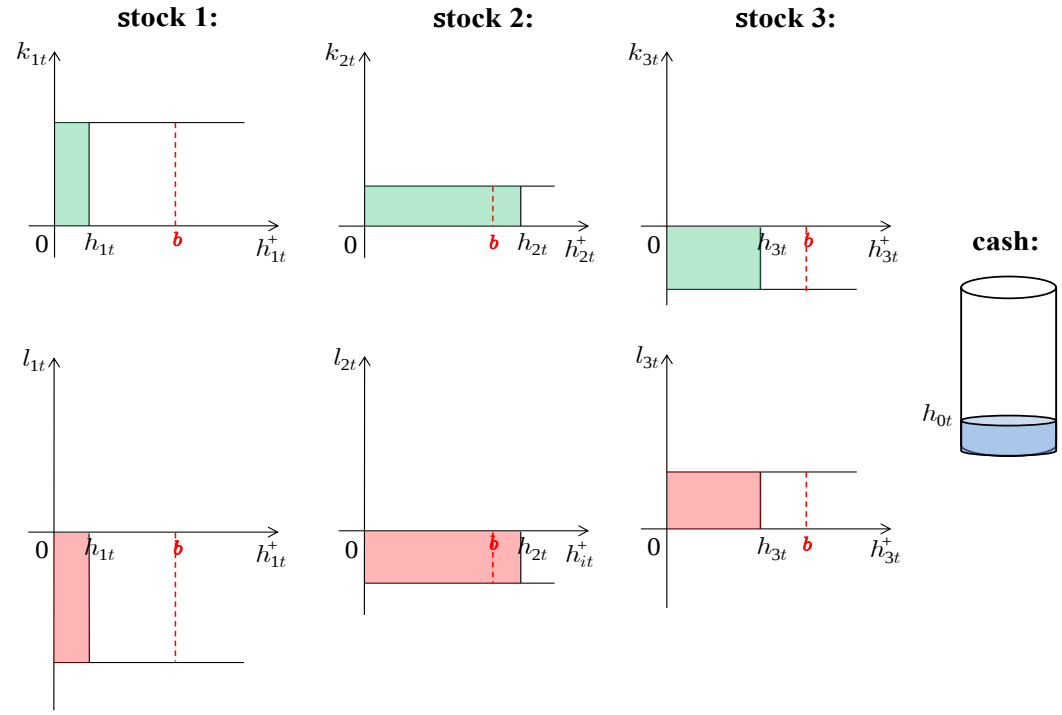


Figure 6.7: Buying and selling slopes vs current holdings before allocation

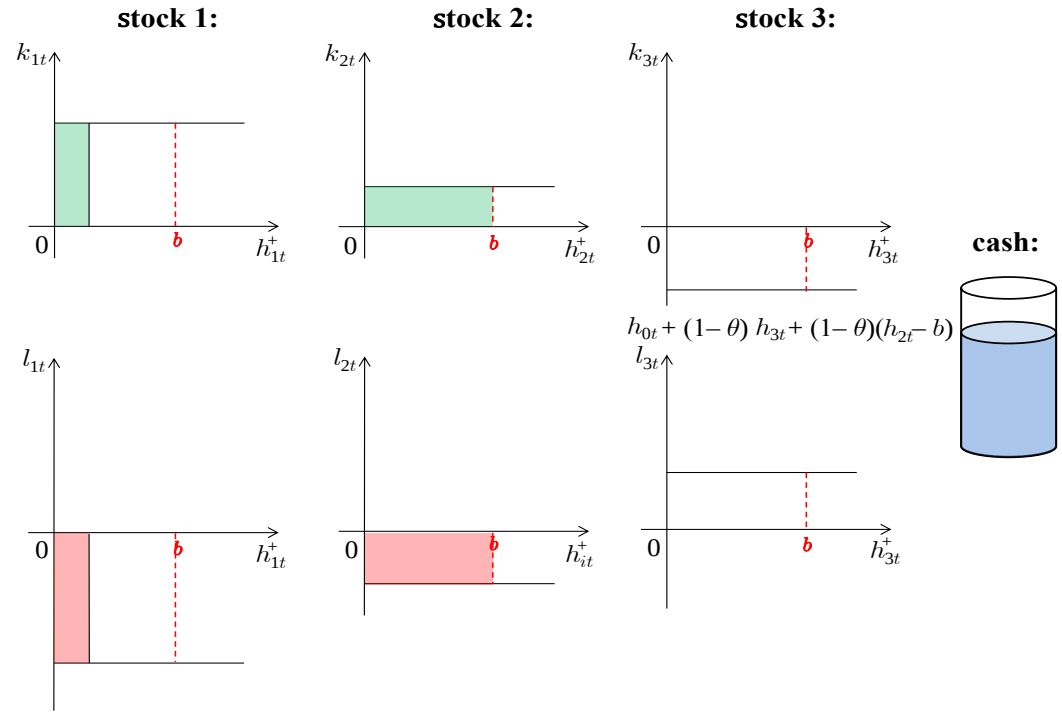


Figure 6.8: Step 1. Sell stock 2 and stock 3 and update cash

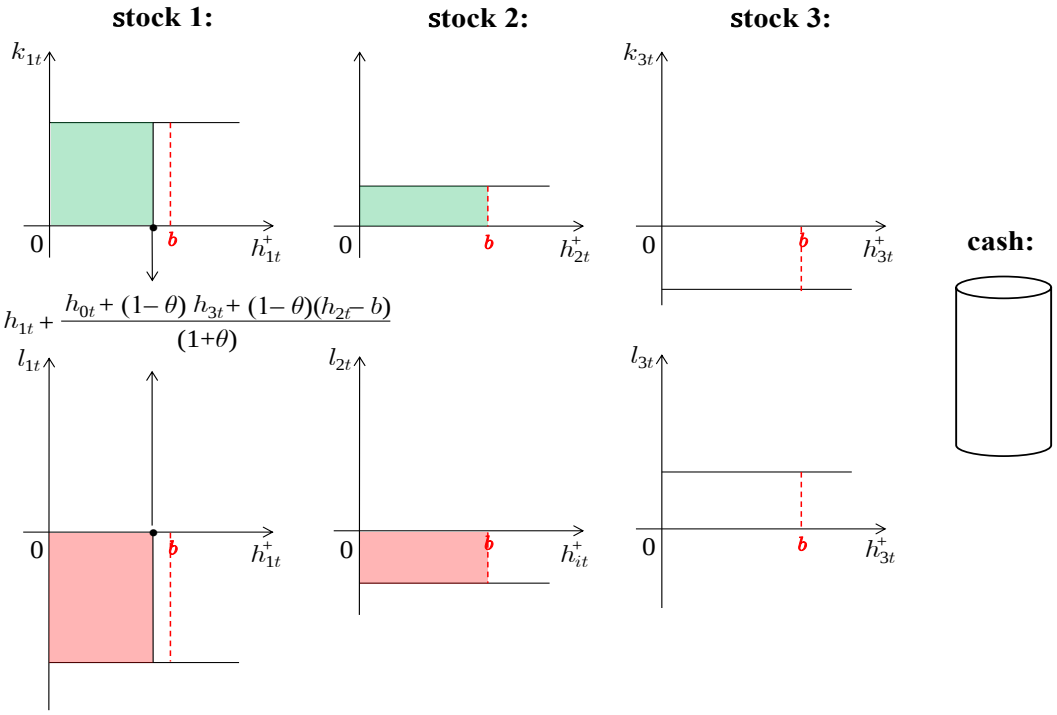


Figure 6.9: Step 2. Buy stock 1 with cash

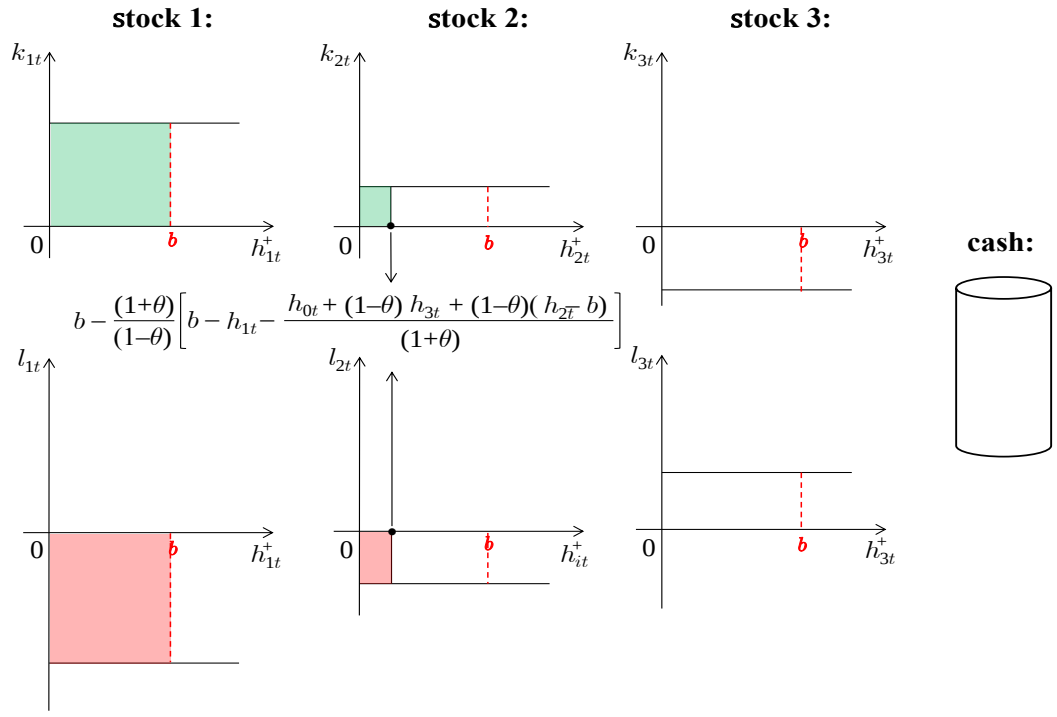


Figure 6.10: Step 3. Sell stock 2 and buy stock 1

Complexity

The linear program (6.18), which we solve with Algorithm 6.2, has a total of $2N$ decision variables and a total of $2N+1$ constraints excluding the $2N$ non-negativity constraints.

In Algorithm 6.2, it takes us $O(N)$ time to find the assets that have positive selling slopes, $O(N)$ time to divert cash and the wealth in Sell-assets and Forced-to-sell-assets to the Buy-assets and $O(N \log N)$ time to sort the assets in a decreasing order with respect to their buying slopes (note that we sort the assets only once and we take the best buying asset $j^\star$ from the top of the list and the best selling asset $i^\star$ from the bottom of the list). This gives us a time complexity of $O(N \log N)$ for Algorithm 6.2. For S T-period realizations of the stochastic process of random returns, the above gives us a polynomial time complexity of $S \cdot T \cdot O(N \log N) + \sum_{s=1}^{S} slogs$ for the LADP-UB algorithm (see section 4.3 for a discussion on the complexity of the General ADP algorithm).

6.4 Discussion

Note the following:

1. In Algorithms 6.1 and 6.2, none of the steps requires that we first visit the previous steps. For example, it might be the case that while solving problem (6.18) with Algorithm 6.2 there exist neither Sell-assets with positive holdings nor Forced-to-sell-assets but we have cash which we use for buying in Step 2. Also, it might be the case that there exist neither Sell-assets with positive holdings nor Forced-to-sell-assets and we have no cash, but there exist assets $i^\star$ and $j^\star$ between which we perform sell-to-buy transactions in Step 3.

2. In order to transform a risky asset into cash, we need to multiply its holdings with $(1-\theta)$, while the reverse transformation requires that we multiply cash with $\frac{1}{1+\theta}$. Further, to transform one risky asset into another one, we need to multiply the holdings of the first with $\frac{1-\theta}{1+\theta}$. These are called the *transformation coefficients* and are summarized in Table 6.1.

Type of asset	**risky**	**cash**
risky	$\frac{1-\theta}{1+\theta}$	$(1-\theta)$
cash	$\frac{1}{1+\theta}$	-

Table 6.1: Transformation coefficients

3. In order to have one additional unit of a risky asset, we either need to sell $\frac{1+\theta}{1-\theta}$ units of another risky asset or use $1+\theta$ units of cash. Further, in order to have one more unit of cash we need to sell $\frac{1}{1-\theta}$ units of a risky asset. These are called the *projection coefficients* and are summarized in Table 6.2. Note that the projection coefficients are the inverse of the transformation coefficients.

Type of asset	**risky**	**cash**
risky	$\frac{1+\theta}{1-\theta}$	$\frac{1}{1-\theta}$
cash	$1+\theta$	-

Table 6.2: Projection coefficients

4. While solving problems (6.6) and (6.18) using respectively Algorithms 6.1 and 6.2, if there exist more than one Buy-assets that have simultaneously the highest buying slope and thus can become the best buying asset $j^\star$, then we have multiple optimal allocations. For the sake of diversification, we resolve this tie by requiring that wealth is equally split among all the assets that have the same highest positive buying slope. However, due to the random returns, it is unlikely that the buying slopes of any assets will ever take the same values.

5. While solving problem (6.6) with Algorithm 6.1, we have at most one best buying asset $j^\star$ which results in the selected portfolios comprising of only one asset. While solving problem (6.18) with Algorithm 6.2, due to the upper bound constraints on the holdings of the assets, the selected portfolios comprise of more than one asset. The number of assets in the selected portfolios depends on the value of parameter α. The smaller the value of parameter α the more the assets in the selected portfolios.

6. Changing the order of buying in Algorithm 6.1 would make no difference to the optimal solution since we buy only one asset. Changing the order of buying in Algorithm 6.2 could result in suboptimal allocations.

6.5 Gradient Information

In sections 6.2 and 6.3, we solved respectively the subproblem of LADP and the subproblem of LADP-UB, where the optimal value of each of the subproblems gives us $\tilde{V}_{t-1}$. Further, recall that we have removed iteration indexing to simplify notation.

As explained in section 4.4, $\hat{V}_{t-1}^{s-1}$ and $\tilde{V}_{t-1}^{s}$ are used to estimate value function approximations $\hat{V}_{t-1}^{s}$ through an update function $U\left(\hat{V}_{t-1}^{s-1}, \Delta\tilde{V}_{i(t-1)}^{s}\right)$. Here, due to the simplicity of the assumed separable linear value function approximations, where the only parameters that need to be estimated are the slopes of the assets, our update function $U\left(\hat{V}_{t-1}^{s-1}, \Delta\tilde{V}_{i(t-1)}^{s}\right)$ takes the following form of simply smoothing on the previous slope estimate $\hat{u}_{i(t-1)}^{s-1}$:

$$\hat{u}_{i(t-1)}^{s} = \left(1 - \alpha_{i(t-1)}^{s}\right) \hat{u}_{i(t-1)}^{s-1} + \alpha_{i(t-1)}^{s} \Delta\tilde{V}_{i(t-1)}^{s}, \tag{6.32}$$

where $\hat{u}_{i(t-1)}^{s-1}$, $\Delta\tilde{V}_{i(t-1)}^{s}$ and $\hat{u}_{i(t-1)}^{s}$ are respectively the *old slope*, the *observed slope* and the *new slope*, and term $\alpha_{i(t-1)}^{s}$ is a quantity between 0 and 1 known as the *stepsize*. For convergence, a stepsize sequence $\{\alpha_{it}^{s}\}_{s=1,2,...}$ must satisfy the following conditions:

$$\left.\begin{array}{l} \alpha_{it}^{s} \geq 0,\ s = 1, 2, ..., \inf \\ \sum_{s=1}^{\inf} \alpha_{it}^{s} = \inf \\ \sum_{s=1}^{\inf} (\alpha_{it}^{s})^2 < \inf \end{array}\right\} \tag{6.33}$$

The first condition requires that stepsizes are non-negative. The second condition requires that the algorithm does not stop prematurely. The third condition requires that stepsizes converge. For a review of the stepsize rules that exist in the literature we refer the reader to [33] and chapter 6 of [65].

Computing the observed slope $\Delta\tilde{V}_{i(t-1)}^{s}$ of asset i for $t < T$ is straightforward and can be done via substituting the optimal solutions of the subproblems of LADP and LADP-UB in their objectives. For a derivation of the gradients in the linear approximate methods, we refer the reader to Appendix C. Here, we only provide a summary of the observed slopes for the different types of assets. To compute observed slopes $\Delta\tilde{V}_{i(T-1)}^{s}$ we use (4.41).

Tables 6.3 and 6.4 summarize the observed slopes $\Delta\tilde{V}_{i(t-1)}^{s}$ for the different types of assets in LADP and LADP-UB respectively.

Type of asset i	**Buy-assets $= \emptyset$**	**Buy-assets $\neq \emptyset$**
Sell-asset	$(1-\theta)\,\hat{u}_{0t}^{s-1} R_{it}^{s}$	$\frac{1-\theta}{1+\theta}\hat{u}_{j^{\star}t}^{s-1} R_{it}^{s}$
Sell-to-buy-asset	-	$\frac{1-\theta}{1+\theta}\hat{u}_{j^{\star}t}^{s-1} R_{it}^{s}$
Cash	$\hat{u}_{0t}^{s-1} R_{0t}^{s}$	$\frac{1}{1+\theta}\hat{u}_{j^{\star}t}^{s-1} R_{it}^{s}$
other asset	$\hat{u}_{it}^{s-1} R_{it}^{s}$	

Table 6.3: Observed slopes $\Delta\tilde{V}_{i(t-1)}^{s}$ in LADP

Remark 6.2. Note the following:

- Through the observed slopes of Tables 6.3 and 6.4, the returns of the assets are passed from the current time period of the current iteration to the previous time period of the next iteration in a multiplicative manner, which means that it takes T iterations for the LADP algorithms to pass the information from the last time period back to the first one.

- As we have previously explained in section 4.4, the gradients that we obtain from the terminal value function using (4.41) are positive. Due to this and the way returns are communicated to previous time periods through the observed slopes of Tables 6.3 and 6.4, if we start with positive initial slope estimates then it easy to verify by induction that all slope estimates will take positive values in all iterations of the LADP algorithms.

Type of asset i		$\mathcal{C}=\emptyset$		$\mathcal{C}\neq\emptyset$		
		$j^\star=\emptyset$	$j^\star\neq\emptyset$	$h^+_{ct}=w_0, \mathcal{D}=\emptyset$	$h^+_{ct}=w_0, \mathcal{D}\neq\emptyset$	$h^+_{ct}<w_0$
Buy-assets	$1:c-1$	-	-	$(1+\theta)\hat{u}^{s-1}_{0t}R^s_{it}$	$\frac{1+\theta}{1-\theta}\hat{u}^{s-1}_{dt}R^s_{it}$	$\hat{u}^{s-1}_{ct}R^s_{it}$
	c	-	$\hat{u}^{s-1}_{ct}R^s_{ct}$	$(1+\theta)\hat{u}^{s-1}_{0t}R^s_{ct}$	$\frac{1+\theta}{1-\theta}\hat{u}^{s-1}_{dt}R^s_{ct}$	$\hat{u}^{s-1}_{ct}R^s_{ct}$
Sell-assets		$(1-\theta)\hat{u}^{s-1}_{0t}R^s_{it}$	$\frac{1-\theta}{1+\theta}\hat{u}^{s-1}_{ct}R^s_{it}$	$(1-\theta)\hat{u}^{s-1}_{0t}R^s_{it}$	$\hat{u}^{s-1}_{dt}R^s_{it}$	$\frac{1-\theta}{1+\theta}\hat{u}^{s-1}_{ct}R^s_{it}$
Forced-to-sell-assets		$(1-\theta)\hat{u}^{s-1}_{0t}R^s_{it}$	$\frac{1-\theta}{1+\theta}\hat{u}^{s-1}_{ct}R^s_{it}$	$(1-\theta)\hat{u}^{s-1}_{0t}R^s_{it}$	$\hat{u}^{s-1}_{dt}R^s_{it}$	$\frac{1-\theta}{1+\theta}\hat{u}^{s-1}_{ct}R^s_{it}$
Sell-to-buy-assets	$1:d-1$	-	-	-	$\hat{u}^{s-1}_{dt}R^s_{it}$	$\frac{1-\theta}{1+\theta}\hat{u}^{s-1}_{ct}R^s_{it}$
	d	-	$\frac{1-\theta}{1+\theta}\hat{u}^{s-1}_{ct}R^s_{dt}$	-	$\hat{u}^{s-1}_{dt}R^s_{dt}$	$\frac{1-\theta}{1+\theta}\hat{u}^{s-1}_{ct}R^s_{dt}$
Cash		$\hat{u}^{s-1}_{0t}R^s_{0t}$	$\frac{1}{1+\theta}\hat{u}^{s-1}_{ct}R^s_{0t}$	$\hat{u}^{s-1}_{0t}R^s_{0t}$	$\frac{1}{1-\theta}\hat{u}^{s-1}_{dt}R^s_{0t}$	$\frac{1}{1+\theta}\hat{u}^{s-1}_{ct}R^s_{0t}$
Other assets		$\hat{u}^{s-1}_{it}R^s_{it}$	$\hat{u}^{s-1}_{it}R^s_{it}$	$\hat{u}^{s-1}_{it}R^s_{it}$	$\hat{u}^{s-1}_{it}R^s_{it}$	$\hat{u}^{s-1}_{it}R^s_{it}$

Table 6.4: Observed slopes $\Delta\hat{V}^s_{i(t-1)}$ in LADP-UB

Chapter 7

Separable Piecewise Linear Approximation

In chapter 6, we considered separable linear approximations for the unknown value functions in the dynamic programming formulation of the portfolio selection problem.

As explained in chapter 6, the simple linear approximation does not provide a good fit for the true value functions since when the holdings of the assets increase beyond a certain amount their distance from the linear approximate value functions becomes too large. Further, in this approximation every additional unit of wealth in a risky asset is valued the same which results in selecting portfolios that comprise of only one asset per time period. To improve the simple linear approximation we imposed upper bounds on the holdings of the risky assets. In this manner, we excluded points from the approximate linear value functions of the risky assets that were far from the true value functions and we had more diversified portfolios per time period. However, as explained in chapter 6, excluding points from the approximate value functions entails excluding states which might be good but we never visit.

In this chapter, we consider separable piecewise linear concave approximations for the unknown value functions of the risky assets. These approximate functions are closer to the true value functions (see Figure 7.1) and as a result they capture the expected attitude of the investor towards risk (the investor becomes risk-averse). The piecewise linear approximate functions allow wealth to flow naturally from one asset to another without us needing to impose diversification by using upper bound constraints. As far as cash is concerned, we assume that the respective value

functions follow the simple linear approximation.

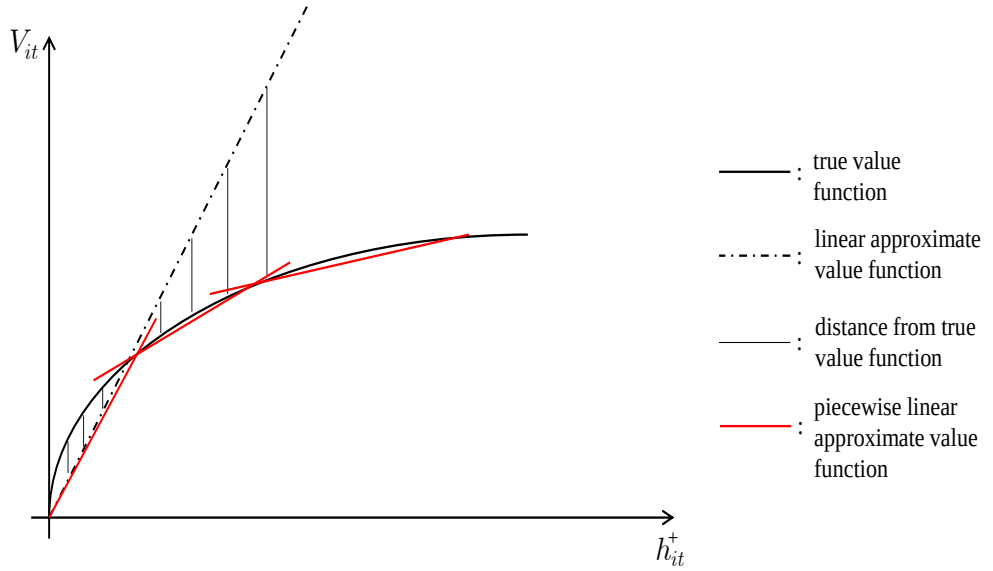


Figure 7.1: Piecewise linear approximate value function of risky asset i in period $t+1$

In the literature, there exist several separable piecewise linear approximate dynamic programming algorithms for problems with either continuous state spaces (see for example [34] and [35] for applications in fleet management, [36] for applications in inventory and distribution problems and [60] for an application in the mutual fund cash problem), or discrete state spaces (see tutorial [67] for a theoretical discussion and references therein for applications). What all these approximations differ in is the update rule of the value functions, which is nevertheless defined so that the monotonicity of the slopes in the approximate functions is preserved in all iterations.

To our knowledge, the problems considered in the existing applications of approximate dynamic programming with piecewise linear value functions are low-dimensional. However, in large-scale problems with continuous state spaces, the number of dimensions together with the number of slopes in each dimension increase significantly the size of the underlying subproblem of ADP and as a result the computational effort to solve it. To circumvent this difficulty, in this study, we propose an update rule that allows us to keep control over the number of slopes in the piecewise linear value functions at every iteration of the dynamic programming algorithm.

This chapter is structured as follows: In section 7.1, we provide representations

of the approximate piecewise linear value functions. Then, in section 7.2 we formulate the subproblem of ADP as a linear program and solve it. After solving the subproblem of ADP, in section 7.3, we propose an update rule for the piecewise linear value functions that preserves the monotonicity of the slopes (for piecewise linear concave functions this requires that slopes are monotone decreasing with respect to state) and maintains the number of slopes below a specified threshold in every approximate piecewise linear value function.

7.1 Piecewise Linear Approximations for the Value Functions

In this section, we implement separable piecewise linear concave functional approximations for the unknown value functions. Specifically, after removing iteration indexing to simplify notation, we replace every $\hat{V}_t\left(\boldsymbol{h}_t^+\right)$ in optimality equations (4.30) and (4.32) with a separable approximation of the following type:

$$\hat{V}_t\left(\boldsymbol{h}_t^+\right) = \sum_{i=0}^{N} \hat{V}_{it}\left(h_{it}^+\right) \tag{7.1}$$

where for every risky asset i function $\hat{V}_{it}\left(h_{it}^+\right)$ is a piecewise linear concave function of m slopes. Every piecewise linear function $\hat{V}_{it}\left(h_{it}^+\right)$ can be represented by a finite set of ordered *breakpoints* $\{\langle u_{it}^\kappa, a_{it}^\kappa\rangle : \kappa \in \mathcal{M}\}$, where:

- $\mathcal{M} = \{1, 2, \ldots, m\}$ is the index set of slopes.
- u_{it}^κ is the κ^{th} slope of the piecewise linear function and due to the concavity assumption we have $u_{it}^1 \geq u_{it}^2 \geq \ldots \geq u_{it}^m > 0$.
- a_{it}^κ is the breakpoint where slope changes from $u_{it}^{\kappa-1}$ to u_{it}^κ. That is, the slope in segment $\left[a_{it}^\kappa, a_{it}^{\kappa+1}\right)$ is u_{it}^κ. We assume that $a_{it}^1 \equiv 0$ and $a_{it}^{m+1} \equiv \infty$ and that the breakpoints are ordered in such a way so that $0 \equiv a_{it}^1 < a_{it}^2 < \ldots < a_{it}^m < a_{it}^{m+1}$.

Regarding cash, we assume that $\hat{V}_{0t}\left(h_{0t}^+\right) = u_{0t} h_{0t}^+$, where $u_{0t} > 0$.
In the discussion that follows, we use Figure 7.2 as a guide to our notation.

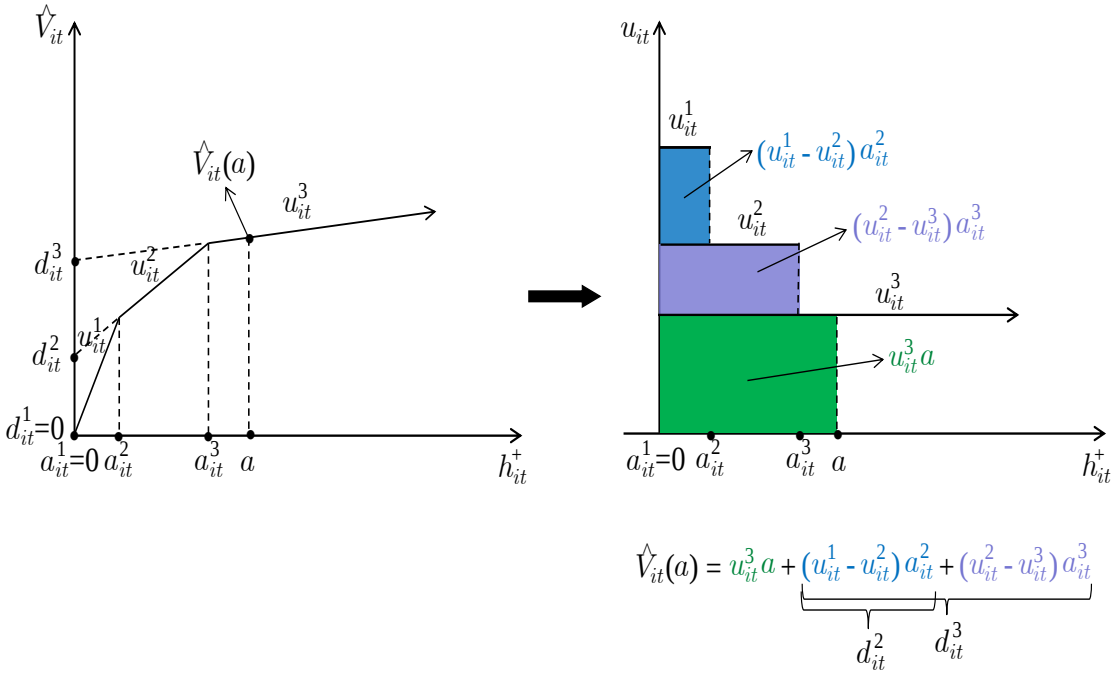


Figure 7.2: Equivalent representations of a piecewise linear function with 3 slopes

In line with [17], we express every piecewise continuous linear function $\hat{V}_{it}\left(h_{it}^+\right)$ in the following form:

$$\hat{V}_{it}\left(h_{it}^+\right) = u_{it}^\kappa h_{it}^+ + d_{it}^\kappa, \text{ for } a_{it}^\kappa \le h_{it}^+ \le a_{it}^{\kappa+1}, \ \kappa \in \mathcal{M} \tag{7.2}$$

where d_{it}^κ is a constant, its value is given by $d_{it}^\kappa = \hat{V}_{it}(0)$ and $d_{it}^1 \equiv 0$. Note that due to the concavity assumption we have $d_{it}^\kappa \le d_{it}^{\kappa+1}$ for all $\kappa \in \mathcal{M}$. Further, for $\kappa > 1$ from the continuity assumption at $h_{it}^+ = a_{it}^\kappa$ we have $u_{it}^{\kappa-1} a_{it}^\kappa + d_{it}^{\kappa-1} = u_{it}^\kappa a_{it}^\kappa + d_{it}^\kappa$ and constants d_{it}^κ can be computed by the following recursion:

$$\left.\begin{aligned} d_{it}^\kappa &= a_{it}^\kappa \left(u_{it}^{\kappa-1} - u_{it}^\kappa\right) + d_{it}^{\kappa-1}, \kappa > 1 \\ d_{it}^1 &= 0 \end{aligned}\right\} \tag{7.3}$$

where from backward substitution we get:

$$d_{it}^\kappa = \sum_{j=2}^{\kappa} \left(u_{it}^{j-1} - u_{it}^j\right) a_{it}^j, \ k > 1 \tag{7.4}$$

If we now substitute (7.4) in (7.2), then the latter becomes:

$$\hat{V}_{it}\left(h_{it}^{+}\right)=u_{it}^{\kappa}h_{it}^{+}+\sum_{j=2}^{\kappa}\left(u_{it}^{j-1}-u_{it}^{j}\right)a_{it}^{j},\ \text{ for } a_{it}^{\kappa}\leq h_{it}^{+}\leq a_{it}^{\kappa+1},\ \kappa\in\mathcal{M}, \tag{7.5}$$

Note that (7.5) gives us the cumulative area under the slopes-holdings plots. In Figure 7.2, we provide an example of a piecewise linear continuous function $\hat{V}_{it}\left(h_{it}^{+}\right)$ with 3 slopes (left plot) and its equivalent representation as the cumulative area in the slopes-holdings plot (right plot), assuming that $h_{it}^{+}=a$ where $a>a_{it}^{3}$. Note that constant d_{it}^{3} is equal to the sum of the areas of the rectangles that are above the rectangle with area $u_{it}^{3}a$.

Substituting separable approximation (7.2) in optimality equations (4.30) and (4.32) using transition equations (4.5), we can write the optimality equations in the following form:

$$\left.\begin{aligned}\tilde{V}_{0}^{-}\left(\boldsymbol{h}_{0}\right)=&\max_{(\boldsymbol{x}_{0},\boldsymbol{y}_{0})\in\mathcal{A}_{0}}\ \sum_{i=1}^{N}\hat{V}_{i0}\left(h_{i0}+x_{i0}-y_{i0}\right)\\&+\hat{V}_{00}\left(h_{00}-(1+\theta)\sum_{i=1}^{N}x_{i0}+(1-\theta)\sum_{i=1}^{N}y_{i0}\right)\\\tilde{V}_{t-1}\left(\boldsymbol{h}_{t-1}^{+}\right)=&\max_{(\boldsymbol{x}_{t},\boldsymbol{y}_{t})\in\mathcal{A}_{t}}\ \sum_{i=1}^{N}\hat{V}_{it}\left(R_{it}h_{i(t-1)}^{+}+x_{it}-y_{it}\right)\\&+\hat{V}_{0t}\left(R_{0t}h_{0(t-1)}^{+}-(1+\theta)\sum_{i=1}^{N}x_{it}+(1-\theta)\sum_{i=1}^{N}y_{it}\right)\end{aligned}\right\} \tag{7.6}$$

In (7.6), $\hat{V}_{it}\left(h_{it}^{+}\right)$ is of type (7.5) for every $i\in\mathcal{N}$ and $\hat{V}_{0t}\left(h_{0t}^{+}\right)=u_{0t}h_{0t}^{+}$. From this point on, we will call in the thesis the General ADP Algorithm 4.1 after we replace $\hat{V}_{t}\left(\boldsymbol{h}_{t}^{+}\right)$ with separable piecewise linear functional approximations of type (7.5) the *Piecewise Linear Approximate Dynamic Programming Algorithm (PLADP)*.

Further, as in chapter 6, the maximization problems in (7.6) for $t=0$ and $t>0$ in (7.6) are similar and differ only in the current holdings which are given by constant h_{i0} if $t=0$ and constant $R_{it}h_{i(t-1)}^{+}$ if $t>0$.

7.2 The Subproblem of PLADP

The optimality equations of (7.6) consist of many maximization problems, each one associated with the respective decision variables $(\boldsymbol{x}_t, \boldsymbol{y}_t)$. These are called the subproblems of PLADP. In this section, we show that each subproblem of PLADP can be written as a linear program. Specifically, in subsection 7.2.1, we discuss how we can represent piecewise linear curves in linear programs with upper and lower bounds on the variables. Then, in subsection 7.2.2, we write the linear programming formulation of the subproblem of PLADP and finally we solve it in subsection 7.2.3.

7.2.1 Representing Piecewise Linear Functions in Optimization Problems

In this section, we consider two examples to show how piecewise linear curves can be represented in optimization problems. In Example 9.1, we represent a piecewise linear curve using integer variables and explain why these can be avoided when we have diseconomies of scale. In Example 9.2, we consider two risky assets with piecewise linear concave value functions and cash with a linear value function, and, using the representation of Example 9.1, we explain how the sum of the three functions can be represented.

Example 7.1. Suppose we have the piecewise linear curve of Figure 7.3.

To model the non-linear function of Figure 7.3, we express variable x as the sum of three new variables x_1, x_2 and x_3, each associated with a marginal cost/reward, which in Figure 7.3 is given by the respective slope. The new variables are such that:

$$x = x_1 + x_2 + x_3,$$

where:

$$0 \leq x_1 \leq 2,$$
$$0 \leq x_2 \leq 5,$$
$$0 \leq x_3 \leq 2$$

and the total cost/reward is given by:

$$f(x) = 5x_1 + 2x_2 + 4x_3$$

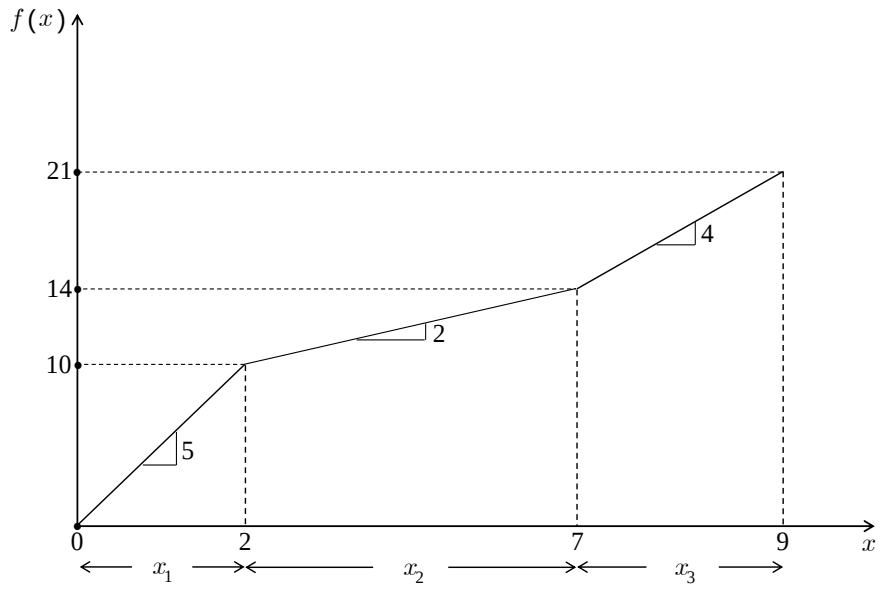


Figure 7.3: A piecewise linear curve with 3 slopes

For the new variables we require that $x_1 = 2$ whenever $x_2 > 0$ and $x_2 = 5$ whenever $x_3 > 0$. To model these conditional statements, we introduce binary variables δ_1 and δ_2 and define them as follows:

$$\delta_1 = \begin{cases} 1, & \text{if } x_1 \text{ at its upper bound} \\ 0, & \text{otherwise} \end{cases}$$

$$\delta_2 = \begin{cases} 1, & \text{if } x_2 \text{ at its upper bound} \\ 0, & \text{otherwise} \end{cases}$$

Using binary variables δ_1 and δ_2, the upper and lower bounds of variables x_1, x_2 and x_3 change as follows:

$$\begin{aligned} 2\delta_1 \leq x_1 \leq 2, \\ 5\delta_2 \leq x_2 \leq 5\delta_1, \\ 0 \leq x_3 \leq 2\delta_2 \end{aligned}$$

In the special case when we have *diseconomies of scale*, where the slopes are increasing/decreasing for a minimization/maximization problem, the binary variables can be ignored. Consider, for example, a maximization problem where the slopes of Figure 7.3 are 5, 4 and 3, which makes the objective:

$$f(x) = 5x_1 + 4x_2 + 3x_3$$

Due to the decreasing slopes and maximization, we first fill up x_1 before we start increasing x_2, and we first fill up x_2 before we start increasing x_3. Thus, $x_2 > 0$ only if x_1 is at its upper bound so δ_1 is not needed and $x_3 > 0$ only if x_2 is at its upper bound so δ_2 is not needed. An analogous reasoning applies for minimization problems, where the slopes are monotone increasing.

Example 7.2. Suppose we have the two piecewise linear curves and the linear curve of Figure 7.4 and let:

- $f(z + 5)$ be the value function of risky asset 1, where constant 5 gives us the current holdings of the asset, z gives us the change in the holdings of the asset after the transactions and $z + 5$ gives us the holdings of the asset after the transactions, for which we require that $z + 5 \geq 0$.

- $h(w + 11)$ be the value function of risky asset 2, where constant 11 gives us the current holdings in the asset, w gives us the change in the holdings of the asset after the transactions and $w + 11$ gives us the holdings of the asset after the transactions, for which we require that $w + 11 \geq 0$.

- $d(u + 1)$ be the value function of cash, where constant 1 gives us the current cash, u gives us the change in cash after the transactions and $u + 1$ gives us the cash after the transactions, for which we require that $u + 1 \geq 0$. We assume that an increase/decrease in the current holdings of the risky assets decreases/increases cash by the same amount, i.e. $u = -(z + w)$.

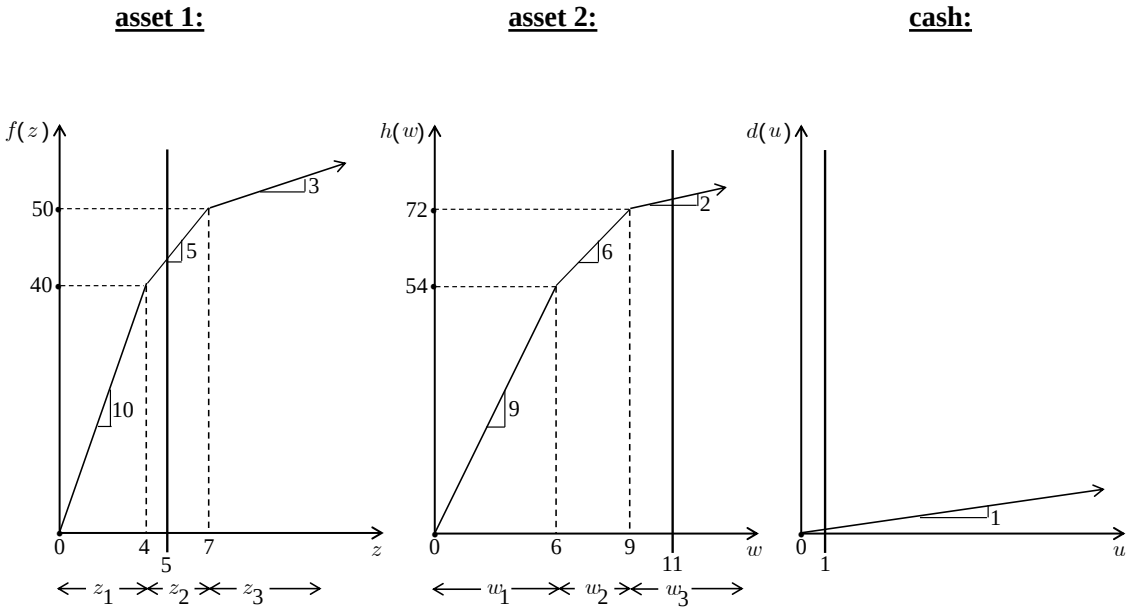


Figure 7.4: Maximizing the sum of two piecewise linear curves with 3 slopes each and one linear curve

Suppose now that we want to maximize the sum of the three functions, which is denoted with $g(z, w, u)$ and is given by:

$$g(z, w, u) = f(z + 5) + h(w + 11) + d(u + 1)$$

Following example 9.1, where we describe how a piecewise linear curve can be represented, for variable z we introduce three new variables z_1, z_2 and z_3, one for each slope, such that z_κ gives us by how much holdings of asset 1 change after the transaction in slope segment κ and $z + 5 = z_1 + z_2 + z_3 + 5$. Thus, if we buy, for example, 3 units of asset 1, i.e. $z = 3 > 0$, then there will be no change in slope segment 1, an increase of 2 units in slope segment 2 and an increase of 1 unit in slope segment 3. That is, $z_1 = 0$, $z_2 = 2$ and $z_3 = 1$. Similarly, if we sell 3 units of asset 1, i.e. $z = -3 < 0$, then there will be no change in slope segment 3, a decrease of 1 unit in slope segment 2 and a decrease of 2 units in slope segment 1. That is, $z_1 = -2$, $z_2 = -1$ and $z_3 = 0$.

Further, we decompose each z_κ into its positive and negative part as follows:

$$z_\kappa^+ = (z_\kappa)^+,$$
$$z_\kappa^- = (z_\kappa)^-,$$

where z_κ^+ and z_κ^- give us respectively the increase and the decrease in slope segment κ and $z_\kappa = z_\kappa^+ - z_\kappa^-$. Following the above example, an increase of 3 units in asset 1 will give us $z_1^+ = z_1^- = 0$, $z_2^+ = 2$, $z_2^- = 0$, and $z_3^+ = 1$, $z_3^- = 0$. Similarly, a decrease of 3 units in asset 1 will give us $z_1^+ = 0$, $z_1^- = 2$, $z_2^+ = 0$, $z_2^- = 1$, and $z_3^+ = z_3^- = 0$.

Considering the above, in Table 7.1 we summarize the lower and upper bounds for the new variables z_κ^+ and z_κ^- for $\kappa = 1, 2, 3$. Note that we always have $z_1^+ = 0$ since slope segment 1 will still be filled up after an increase and $z_3^- = 0$ since slope segment 3 will still be empty after a decrease. Also, note that slope segment 2 is neither filled up nor empty and as a result both z_2^+ and z_2^- have positive upper bounds.

z	
z^+	z^-
$z_1^+ = 0$	$0 \leq z_1^- \leq 4$
$0 \leq z_2^+ \leq 2$	$0 \leq z_2^- \leq 1$
$z_3^+ \geq 0$	$z_3^- = 0$

Table 7.1: Upper and lower bounds for variables z_κ^- and z_κ^+

In a similar manner, we repeat the above analysis for asset 2. Thus, for variable w we introduce three new variables w_1, w_2 and w_3, such that w_κ gives us by how much holdings of asset 2 change after the transaction in slope segment κ and $w + 11 = w_1 + w_2 + w_3 + 11$. If we decompose each w_κ into its positive and negative part as follows:

$$w_\kappa^+ = (w_\kappa)^+ ,$$
$$w_\kappa^- = (w_\kappa)^- ,$$

where w_κ^+ and w_κ^- give us respectively the increase and the decrease in slope segment κ and $w_\kappa = w_\kappa^+ - w_\kappa^-$, then Table 7.2 gives us the lower and upper bounds for the new variables w_κ^- and w_κ^+. Note that we always have $w_1^+ = w_2^+ = 0$ since slope segments 1 and 2 will still be filled up after an increase. Also, note that due to the current holdings of asset 2 expanding to all three slopes, all w^-s have positive upper bounds.

w	
w^+	w^-
$w_1^+ = 0$	$0 \leq w_1^- \leq 6$
$w_2^+ = 0$	$0 \leq w_2^- \leq 3$
$w_3^+ \geq 0$	$0 \leq w_3^- \leq 2$

Table 7.2: Upper and lower bounds for variables w_κ^- and w_κ^+

Note that in Tables 7.1 and 7.2 the sum of the upper bounds of the negative parts of the new variables give us the respective constant terms and this ensures that $z+5$ and $w+11$ are non-negative.

Further, to ensure that cash is always positive we need to have $u+1 \geq 0$. Given that $u = -(z+w)$, we get:

$$z_1^+ + z_2^+ + z_3^+ + w_1^+ + w_2^+ + w_3^+ \leq 1 + z_1^- + z_2^- + z_3^- + w_1^- + w_2^- + w_3^-, \tag{7.7}$$

which is the budget constraint.

We are now ready to express the objective, which is given by function $g(z, w, u)$, in terms of z_κ^+'s, z_κ^-'s, w_κ^+'s and w_κ^-'s.

Note the following:

$$\begin{aligned} f(z+5) &= f(z_1+z_2+z_3+5) = 10z_1 + 5z_2 + 3z_3 + f(5) \\ &= 10z_1^+ + 5z_2^+ + 3z_3^+ - 10z_1^- - 5z_2^- - 3z_3^- + f(5), \\ h(w+11) &= h(w_1+w_2+w_3+11) = 9w_1 + 6w_2 + 2w_3 + h(11) \\ &= 9w_1^+ + 6w_2^+ + 2w_3^+ - 9w_1^- - 6w_2^- - 2w_3^- + h(11), \end{aligned}$$

where $f(z+5)$, for example, is given by the value of $f(\cdot)$ at constant 5 plus the terms that give us the changes in each slope segment. Note that $f(5)$ and $h(11)$ can be computed by $f(5) = 4 \times 10 + 5 = 45$ and $h(11) = 6 \times 9 + 3 \times 6 + 2 \times 2 = 76$ since $f(\cdot)$ and $h(\cdot)$ are piecewise linear functions with given slopes.

Further, considering that $u = -(z+w)$, $d(u+1)$ is given by

$$\begin{aligned} d(u+1) &= u + d(1) = -z - w + d(1) \\ &= -z_1 - z_2 - z_3 - w_1 - w_2 - w_3 + d(1) \\ &= -z_1^+ - z_2^+ - z_3^+ + z_1^- + z_2^- + z_3^- \\ &- w_1^+ - w_2^+ - w_3^+ + w_1^- + w_2^- + w_3^- + d(1), \end{aligned}$$

where $d(1) = 1 \times 1 = 1$.

Considering the above, objective $g(z, w, u)$ is given by:

$$\begin{aligned} g(z,w,u) &= f(z+5) + h(w+11) + d(u+1) \\ &= 9z_1^+ + 4z_2^+ + 2z_3^+ + 8w_1^+ + 5w_2^+ + w_3^+ \\ &\quad - 9z_1^- - 4z_2^- - 2z_3^- - 8w_1^- - 5w_2^- - w_3^- + 122 \end{aligned} \tag{7.8}$$

7.2.2 Linear Programming Formulation of the Subproblem of PLADP

In this section, we formulate the subproblem of PLADP as a linear program. Specifically, we begin with writing the linear programming formulation of the problem of Example 9.2, where we had 2 risky assets and cash. Then, in an analogous manner, we expand this problem to include N risky assets and cash, where we define the decision variables, write the constraints and the objective and finally state the analogous linear programming formulation.

Example 9.2. (Continued) Combining objective (7.8) with the upper and lower bounds from Tables (7.1) and (7.2), as well as with inequality (7.7), the linear programming formulation of the problem described in Example 9.2 above is as follows:

$$\left.\begin{aligned} \max_{(z_\kappa^+, w_\kappa^+)} \quad & 4z_2^+ + 2z_3^+ + w_3^+ - 9z_1^- - 4z_2^- - 8w_1^- - 5w_2^- - w_3^- + 122 \\ \text{s.t.} \quad & z_2^+ + z_3^+ + w_3^+ \leq 1 + z_1^- + z_2^- + w_1^- + w_2^- + w_3^- \\ & 0 \leq z_1^- \leq 4 \\ & 0 \leq z_2^+ \leq 2 \\ & 0 \leq z_2^- \leq 1 \\ & 0 \leq w_1^- \leq 6 \\ & 0 \leq w_2^- \leq 3 \\ & 0 \leq w_3^- \leq 2 \\ & z_3^+, w_3^+ \geq 0 \end{aligned}\right\} \tag{7.9}$$

where note that we fixed the zero variables z_1^+, z_3^-, w_1^+ and w_2^+.

Figure 7.5 shows a piecewise linear value function with 3 slopes as well as the decisions associated with each slope and is used as a guide to the discussion that follows.

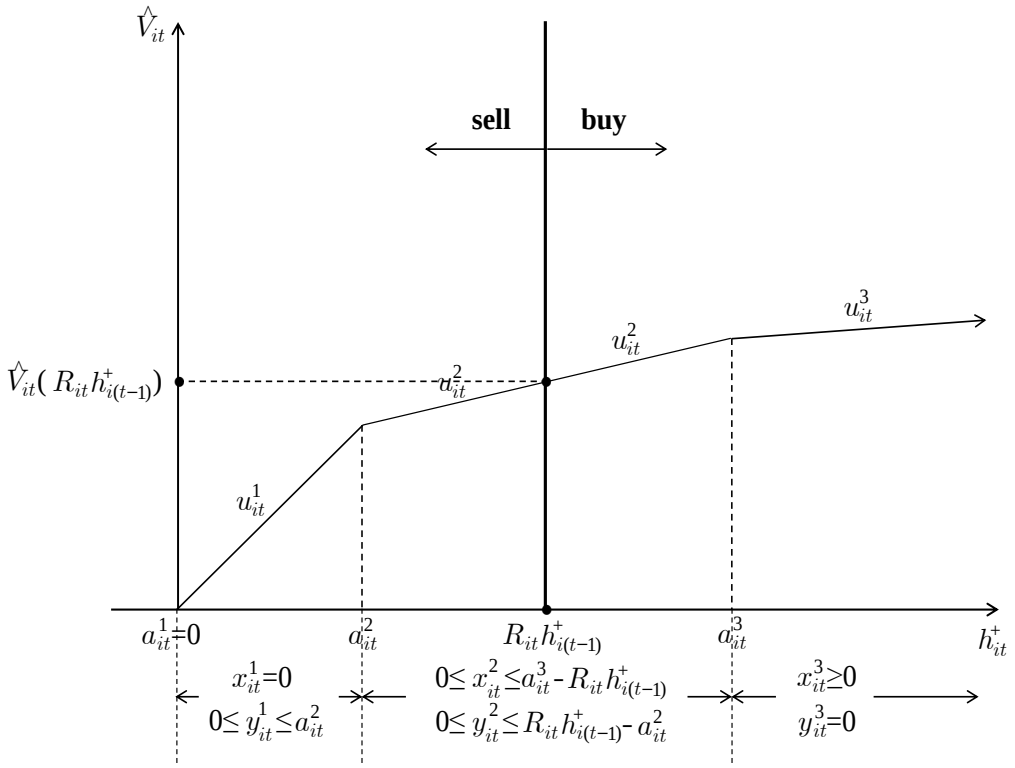


Figure 7.5: Decision variables for a piecewise linear value function with 3 slopes

Decision Variables

In each maximization problem of (7.6), we want to maximize the sum of N separable piecewise linear functions $\hat{V}_{it}\left(R_{it}h^+_{i(t-1)} + x_{it} - y_{it}\right)$ and linear function $\hat{V}_{0t}\left(R_{0t}h^+_{0(t-1)} - (1+\theta)\sum_{i=1}^{N} x_{it} + (1-\theta)\sum_{i=1}^{N} y_{it}\right)$. Considering Lemma 4.3.1, according to which when one of x_{it} and y_{it} is positive then the other one is zero, and that $R_{it}h^+_{i(t-1)}$ is a constant when we solve each maximization problem of (7.6), every variable x_{it} can be replaced by m new variables x^κ_{it}, one for each slope, such that:

$$x_{it} = \sum_{\kappa=1}^{m} x^\kappa_{it}, \tag{7.10}$$

where variables x^κ_{it} are analogous to the positive parts of the variables in Example 9.2.

Similarly, every variable y_{it} can be replaced by m new variables y^κ_{it}, one for each slope, such that:

$$y_{it} = \sum_{\kappa=1}^{m} y_{it}^{\kappa} \tag{7.11}$$

where variables y_{it}^{κ} are analogous to the negative parts of the variables in Example 9.2.

Note that the main difference between the above and Example 9.2 is the linear function $\hat{V}_{0t}\left(R_{0t}h_{0(t-1)}^{+} - (1+\theta)\sum_{i=1}^{N} x_{it} + (1-\theta)\sum_{i=1}^{N} y_{it}\right)$ of cash. Here the input is constant $R_{0t}h_{0(t-1)}^{+}$ plus a linear combination of the decision variables, where the coefficients involve transaction cost parameter θ.

From the above, our decision variables can be summarized as follows:

- x_{it}^{κ}, $i \in \mathcal{N}$, $\kappa \in \mathcal{M}$: the increase, i.e. how much we buy, in slope segment κ of risky asset i in period $t+1$
- y_{it}^{κ}, $i \in \mathcal{N}$, $\kappa \in \mathcal{M}$: the decrease, i.e. how much we sell, in slope segment κ of risky asset i in period $t+1$

Constraints

Recall that the lower and the upper bounds of the variables in Example 9.2 depend on the current holdings of the assets. In an analogous manner, to derive the upper and lower bounds for x_{it}^{κ} and y_{it}^{κ} we will use slopes $\pi(i,+)$ and $\pi(i,-)$, which give us the positions of the right and left slope at current holdings $R_{it}h_{i(t-1)}^{+}$, are called respectively the *active* right and left slope of asset i and are defined as follows:

$$\pi(i,+) = \left\{\kappa+1 \in \mathcal{M} :\ a_{it}^{\kappa+1} \leq R_{it}h_{i(t-1)}^{+} < a_{it}^{\kappa+2},\ i \in \mathcal{N}\right\}, \tag{7.12}$$

$$\pi(i,-) = \left\{\kappa \in \mathcal{M} :\ a_{it}^{\kappa} < R_{it}h_{i(t-1)}^{+} \leq a_{it}^{\kappa+1},\ i \in \mathcal{N}\right\}, \tag{7.13}$$

where we assume that $\pi(i,-) = 0$ if $R_{it}h_{i(t-1)}^{+} = 0$. As for the relationship between $\pi(i,+)$ and $\pi(i,-)$, note that when holdings $R_{it}h_{i(t-1)}^{+}$ lie between breakpoints, i.e. when $a_{it}^{\kappa} < R_{it}h_{i(t-1)}^{+} < a_{it}^{\kappa+1}$, then $\pi(i,+) = \pi(i,-)$, otherwise $\pi(i,+) = \pi(i,-) + 1$. For example, in Figure 7.5 we have $a_{it}^{2} < R_{it}h_{i(t-1)}^{+} < a_{it}^{3}$ and $\pi(i,+) = \pi(i,-) = 2$.

We are now ready to write the upper and lower bounds of our decision variables.

Recall that in Example 9.2 the positive parts of the variables were only meaningful at the right of the constants, while the negative parts were only meaningful at the left of the constants. In a similar manner, here x_{it}^{κ}'s are only meaningful for slopes that lie at the right of holdings $R_{it}h_{i(t-1)}^{+}$, i.e. for $\kappa \geq \pi(i,+)$, and satisfy the following constraints:

$$x_{it}^{\kappa} = 0, \text{ for } \kappa \leq \pi(i,+) - 1 \tag{7.14}$$

$$0 \leq x_{it}^{\kappa} \leq a_{it}^{\kappa+1} - R_{it}h_{i(t-1)}^{+}, \text{ for } \kappa = \pi(i,+) \tag{7.15}$$

$$0 \leq x_{it}^{\kappa} \leq a_{it}^{\kappa+1} - a_{it}^{\kappa}, \text{ for } \kappa \geq \pi(i,+) + 1 \tag{7.16}$$

Further, y_{it}^{κ}'s are only meaningful for slopes that lie at the left of holdings $R_{it}h_{i(t-1)}^{+}$, i.e. for $\kappa \leq \pi(i,-)$, and satisfy the following constraints:

$$0 \leq y_{it}^{\kappa} \leq a_{it}^{\kappa+1} - a_{it}^{\kappa}, \text{ for } \kappa \leq \pi(i,-) - 1 \tag{7.17}$$

$$0 \leq y_{it}^{\kappa} \leq R_{it}h_{i(t-1)}^{+} - a_{it}^{\kappa}, \text{ for } \kappa = \pi(i,-) \tag{7.18}$$

$$y_{it}^{\kappa} = 0, \text{ for } \kappa \geq \pi(i,-) + 1 \tag{7.19}$$

Constraints (7.14)-(7.19) simply tell us that at the left of holdings $R_{it}h_{i(t-1)}^{+}$ we only sell and at the right of holdings $R_{it}h_{i(t-1)}^{+}$ we only buy (see Figure 7.5). Also, note that, as in Example 9.2, when a slope is neither filled up nor empty the respective upper bounds of x_{it}^{κ} and y_{it}^{κ} are both positive. In Figure 7.5, for example, this is true for the decisions of the 2nd slope. Further, as in Example 9.2, it is easy to verify that the sum of the upper bounds of constraints (7.17)-(7.19) give us current holdings $R_{it}h_{i(t-1)}^{+}$, which means that for every risky asset we cannot sell more than what we currently have and this ensures non-negativity of holdings h_{it}^{+}.

We are now ready to write the action space $\mathcal{A}_t$, which in the piecewise linear approximation contains the upper and lower bounds of the variables that are due to the piecewise linear functions as well as the inequalities in the set $\mathcal{A}_t$ of section 4.1, which are as follows:

$$-x_{it} + y_{it} \leq R_{it}h_{i(t-1)}^{+}, \ i \in \mathcal{N}, \tag{7.20}$$

$$(1+\theta)\sum_{i=1}^{N} x_{it} - (1-\theta)\sum_{i=1}^{N} y_{it} \leq R_{0t}h^{+}_{0(t-1)}, \tag{7.21}$$

$$x_{it}, y_{it} \geq 0, \;\; i \in \mathcal{N}, \tag{7.22}$$

If we plug (7.10) and (7.11) in (7.20), then from (7.17)-(7.19) and due to $x^{\kappa}_{it} \geq 0$ for all κ we have:

$$-x_{it} + y_{it} \leq \sum_{\kappa=1}^{m} y^{\kappa}_{it} \leq R_{it}h^{+}_{i(t-1)},$$

i.e. inequality (7.20) is always true given the upper bounds of the variables in (7.14)-(7.19) and thus is not needed.

Further, it is easy to verify that given the lower bounds of the variables in (7.14)-(7.19) the non-negativity inequalities in (7.22) are always true and thus are also not needed.

Regarding the budget constraint, after substituting (7.10) and (7.11) in (7.21), the latter becomes:

$$\sum_{i=1}^{N}\sum_{\kappa=1}^{m} x^{\kappa}_{it} \leq \frac{1}{1+\theta}R_{0t}h^{+}_{0(t-1)} + \frac{1-\theta}{1+\theta}\sum_{i=1}^{N}\sum_{\kappa=1}^{m} y^{\kappa}_{it}, \tag{7.23}$$

Thus, as in Example 9.2, where the sum of the positive parts of the variables is constrained by the sum of the negative parts from above, in a similar manner here the sum of x^{κ}_{it}'s is constrained by the sum of y^{κ}_{it}'s from above. The only difference is the transaction costs.

From the above, x^{κ}_{it}'s and y^{κ}_{it}'s are feasible when they satisfy (7.14)-(7.19) and (7.23) and thus the action space $\mathcal{A}_t$ in the piecewise linear approximation is given by:

$$\mathcal{A}_t = \{(x^{\kappa}_{it}, y^{\kappa}_{it}) : \; (7.14) - (7.19) \text{ and } (7.23) \text{ hold}, \; i \in \mathcal{N}, \; \kappa \in \mathcal{M}\} \tag{7.24}$$

Objective

Similarly to Example 9.2, using $x_{it} = \sum_{\kappa=1}^{m} x^{\kappa}_{it}$ and $y_{it} = \sum_{\kappa=1}^{m} y^{\kappa}_{it}$ each piecewise linear function $\hat{V}_{it}\left(R_{it}h^{+}_{i(t-1)} + x_{it} - y_{it}\right)$ in (7.6) can be written as the sum of the following terms:

- $\hat{V}_{it}\left(R_{it}h^{+}_{i(t-1)}\right)$: This is the value of the function at current holdings $R_{it}h^{+}_{i(t-1)}$, which as explained earlier is treated as a constant, and can be computed by the respective cumulative area in the slopes-holdings plot (see figure 7.5) using:

$$\hat{V}_{it}\left(R_{it}h^{+}_{i(t-1)}\right) = \sum_{\kappa=1}^{\pi(i,-)-1} u^{\kappa}_{it}\left(a^{\kappa+1}_{it} - a^{\kappa}_{it}\right) + u^{\pi(i,-)}_{it}\left(R_{it}h^{+}_{i(t-1)} - a^{\pi(i,-)}_{it}\right) \tag{7.25}$$

- $\hat{V}_{it}\left(\sum_{\kappa=1}^{m} x^{\kappa}_{it} - \sum_{\kappa=1}^{m} y^{\kappa}_{it}\right)$: This is the change in value $\hat{V}_{it}\left(R_{it}h^{+}_{i(t-1)}\right)$ after we take decisions x^{κ}_{it} and y^{κ}_{it} and can be computed by the respective areas in the slopes-holdings plot using:

$$\hat{V}_{it}\left(\sum_{\kappa=1}^{m} x^{\kappa}_{it} - \sum_{\kappa=1}^{m} y^{\kappa}_{it}\right) = \sum_{\kappa=1}^{m}\left(u^{\kappa}_{it}x^{\kappa}_{it} - u^{\kappa}_{it}y^{\kappa}_{it}\right) \tag{7.26}$$

If we now expand the first N terms of the objective in (7.6) using (7.25) and (7.26), and substitute the value function of cash with:

$$u_{0t}\left(R_{0t}h^{+}_{0(t-1)} - (1+\theta)\sum_{i=1}^{N}\sum_{\kappa=1}^{m} x^{\kappa}_{it} + (1-\theta)\sum_{i=1}^{N}\sum_{\kappa=1}^{m} y^{\kappa}_{it}\right),$$

then the objective in (7.6) becomes:

$$\begin{aligned}&\sum_{i=1}^{N}\sum_{\kappa=1}^{m} u^{\kappa}_{it}\left(u^{\kappa}_{it}x^{\kappa}_{it} - u^{\kappa}_{it}y^{\kappa}_{it}\right) + \sum_{i=1}^{N}\sum_{\kappa=1}^{\pi(i,-)-1}\left(a^{\kappa+1}_{it} - a^{\kappa}_{it}\right) + \sum_{i=1}^{N} u^{\pi(i,-)}_{it}\left(R_{it}h^{+}_{i(t-1)}\right.\\ &\left.-a^{\pi(i,-)}_{it}\right) + u_{0t}\left(R_{0t}h^{+}_{0(t-1)} - (1+\theta)\sum_{i=1}^{N}\sum_{\kappa=1}^{m} x^{\kappa}_{it} + (1-\theta)\sum_{i=1}^{N}\sum_{\kappa=1}^{m} y^{\kappa}_{it}\right)\end{aligned} \tag{7.27}$$

After some algebra, objective (7.27) takes the following form:

$$\begin{aligned}&\sum_{i=1}^{N}\sum_{\kappa=1}^{m} u^{\kappa}_{it}\left(k^{\kappa}_{it}x^{\kappa}_{it} + l^{\kappa}_{it}y^{\kappa}_{it}\right) + \sum_{i=1}^{N}\sum_{\kappa=1}^{\pi(i,-)-1}\left(a^{\kappa+1}_{it} - a^{\kappa}_{it}\right)\\ &+ \sum_{i=1}^{N} u^{\pi(i,-)}_{it}\left(R_{it}h^{+}_{i(t-1)} - a^{\pi(i,-)}_{it}\right) + u_{0t}R_{0t}h^{+}_{0(t-1)},\end{aligned} \tag{7.28}$$

where $k^{\kappa}_{it} = u^{\kappa}_{it} - (1+\theta)u_{0t}$ and $l^{\kappa}_{it} = -u^{\kappa}_{it} + (1-\theta)u_{0t}$ are respectively the κ^{th} buying and selling slopes of asset i in period $t+1$. For the active right and left

slopes $u_{it}^{\pi(i,+)}$ and $u_{it}^{\pi(i,-)}$, slopes $k_{it}^{\pi(i,+)}$ and $l_{it}^{\pi(i,-)}$ are the analogous *active* buying and selling slopes.

Buying and Selling Slopes

Similarly to the linear approximations in chapter 6, every original slope defines two new slopes, one buying slope and one selling slope, which satisfy the properties of theorem 6.2.1. Due to $u_{it}^1 \geq u_{it}^2 \geq \ldots \geq u_{it}^m$, the associated buying slopes are *monotonically decreasing*, i.e. $k_{it}^1 \geq k_{it}^2 \geq \ldots \geq k_{it}^m$ and the associated selling slopes are *monotonically increasing*, i.e. $l_{it}^1 \leq l_{it}^2 \leq \ldots \leq l_{it}^m$.

Figure 7.6 shows the transition from the original slopes-holdings plot, where we have assumed that we have 3 slopes, to the buying and selling slopes-holdings plots for risky asset i in period $t+1$.

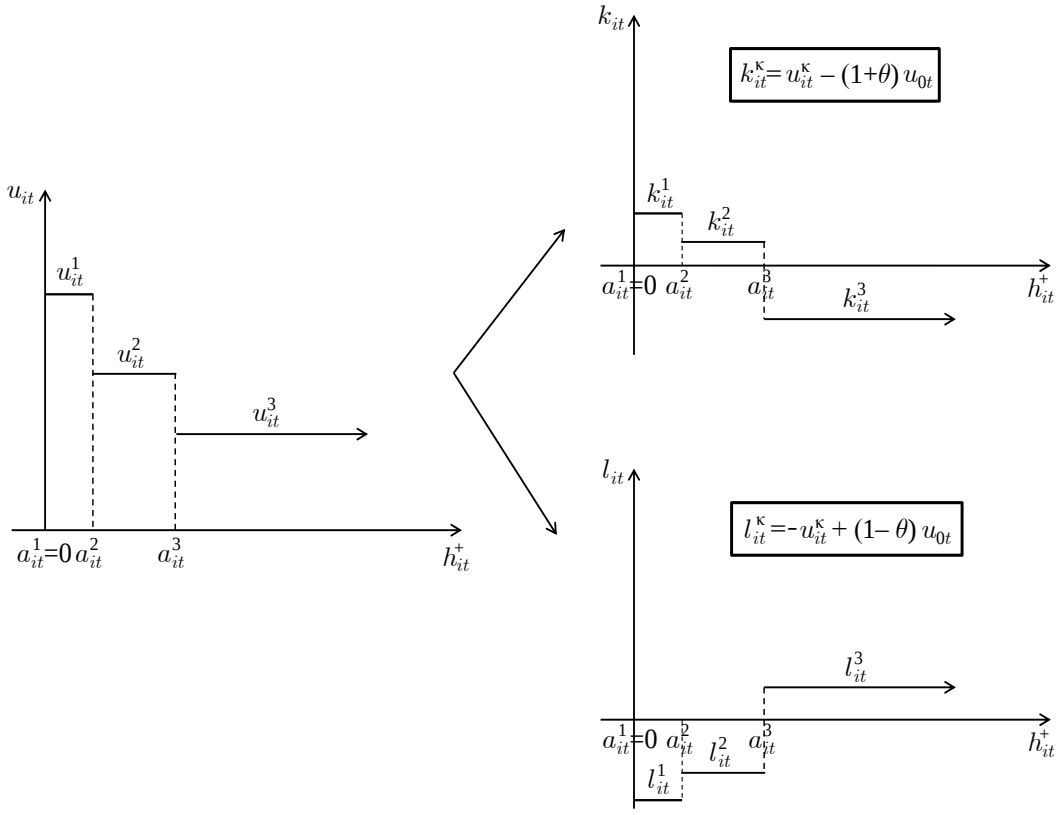


Figure 7.6: Transition from original slopes u_{it}^κ to buying slopes k_{it}^κ and selling slopes l_{it}^κ

Formulation

We are now ready to write the linear programming formulation of the subproblem of PLADP for period $t+1$.

Combining objective (7.28) with the constraints in (7.24), the subproblem of ADP can be expressed in the following form:

$$
\begin{aligned}
\max_{(x_{it}^{\kappa},y_{it}^{\kappa})} \quad & \sum_{i=1}^{N}\sum_{\kappa=1}^{m}\left(k_{it}^{\kappa}x_{it}^{\kappa}+l_{it}^{\kappa}y_{it}^{\kappa}\right)+\sum_{i=1}^{N}\sum_{\kappa=1}^{\pi(i,-)-1}u_{it}^{\kappa}\left(a_{it}^{\kappa+1}-a_{it}^{\kappa}\right) \\
& +\sum_{i=1}^{N}u_{it}^{\pi(i,-)}\left(R_{it}h_{i(t-1)}^{+}-a_{it}^{\pi(i,-)}\right)+u_{0t}R_{0t}h_{0(t-1)}^{+} \\
\text{s.t.} \quad & \sum_{i=1}^{N}\sum_{\kappa=1}^{m}x_{it}^{\kappa}\leq\frac{1}{1+\theta}R_{0t}h_{0(t-1)}^{+}+\frac{1-\theta}{1+\theta}\sum_{i=1}^{N}\sum_{\kappa=1}^{m}y_{it}^{\kappa} \\
& y_{it}^{\kappa}\leq a_{it}^{\kappa+1}-a_{it}^{\kappa},\ \ \kappa\leq\pi(i,-)-1,\ \ \kappa\in\mathcal{M},\ \ i\in\mathcal{N} \\
& y_{it}^{\kappa}\leq R_{it}h_{i(t-1)}^{+}-a_{it}^{r(i)},\ \ \kappa=\pi(i,-),\ \ \kappa\in\mathcal{M},\ \ i\in\mathcal{N} \\
& y_{it}^{\kappa}=0,\ \ \kappa\geq\pi(i,-)+1,\ \ \kappa\in\mathcal{M},\ \ i\in\mathcal{N} \\
& x_{it}^{\kappa}=0,\ \ \kappa\leq\pi(i,+)-1,\ \ \kappa\in\mathcal{M},\ \ i\in\mathcal{N} \\
& x_{it}^{\kappa}\leq a_{it}^{r(i)+1}-R_{it}h_{i(t-1)}^{+},\ \ \kappa=\pi(i,+),\ \ \kappa\in\mathcal{M},\ \ i\in\mathcal{N} \\
& x_{it}^{\kappa}\leq a_{it}^{\kappa+1}-a_{it}^{\kappa},\ \ \kappa\geq\pi(i,+)+1,\ \ \kappa\in\mathcal{M},\ \ i\in\mathcal{N} \\
& x_{it}^{\kappa},y_{it}^{\kappa}\geq 0,\ \ \kappa\in\mathcal{M},\ \ i\in\mathcal{N}
\end{aligned}
\tag{7.29}
$$

Note that problem (7.9) of Example 9.2 is an instance of problem (7.29) for $N=2$, $m=3$, $\theta=0$, the breakpoints of Figure 7.4 and buying and selling slopes the objective coefficients of z_{κ}^{+}'s, w_{κ}^{+}'s and z_{κ}^{-}'s, w_{κ}^{-}'s respectively.

After solving problem (7.29), we can compute optimal decisions $x_{it}^{\star}$'s and $y_{it}^{\star}$'s by plugging optimal decisions $x_{it}^{\kappa,\star}$'s and $y_{it}^{\kappa,\star}$'s into (7.10) and (7.11) respectively.

7.2.3 Solution to the Subproblem of PLADP

As explained in section 7.2.2, problem (7.9) of Example 9.2 is a particular instance of problem (7.29) without transaction costs. In this section, we first solve problem (7.9) of Example 9.2 and then, in an analogous manner, we solve the subproblem of PLADP.

Example 9.2. (Continued) We begin with reminding the reader of problem (7.9):

$$\left.\begin{aligned}
\max_{(z_\kappa^+, w_\kappa^+)} \quad & 4z_2^+ + 2z_3^+ + w_3^+ - 9z_1^- - 4z_2^- - 8w_1^- - 5w_2^- - w_3^- + 122 \\
\text{s.t.} \quad & z_2^+ + z_3^+ + w_3^+ \leq 1 + z_1^- + z_2^- + w_1^- + w_2^- + w_3^- \\
& 0 \leq z_1^- \leq 4 \\
& 0 \leq z_2^+ \leq 2 \\
& 0 \leq z_2^- \leq 1 \\
& 0 \leq w_1^- \leq 6 \\
& 0 \leq w_2^- \leq 3 \\
& 0 \leq w_3^- \leq 2 \\
& z_3^+, w_3^+ \geq 0
\end{aligned}\right\} \quad (7.30)$$

Initially, we set all variables to zero.

The constraints in problem (7.30) give us fixed lower and upper bounds for the variables and the relationship between them through the budget constraint.

Looking now at the budget constraint, we notice that z_κ^+'s and w_κ^+'s are constrained from above by current cash, which is 1 unit, as well as by z_κ^-'s and w_κ^-'s.

Since the coefficients of z_κ^-'s and w_κ^-'s in the objective of (7.30) are all negative, we will maximize z_κ^+'s and w_κ^+'s that have positive coefficients.

Thus, we will first increase variable z_2^+, which has the highest positive coefficient in the objective, up to the level of current cash, which is 1 unit. This will reduce the upper bound of variable z_2^+ by 1 unit and will increase the objective by 4 units.

After we exhaust the resources in the budget constraint, however, we will not stop. Instead, we will check if we can achieve a positive contribution in the objective function by simultaneously increasing one of z_κ^-'s and w_κ^-'s and one of z_κ^+'s and w_κ^+'s. Note that if there exists such a combination of variables, then stopping after exhausting current cash would lead to a suboptimal solution.

Considering the above, we will simultaneously increase variables z_2^+ and w_3^- as the contribution of this simultaneous increase in the objective is $4 - 1 = 3$ units and this is the highest contribution that can be achieved. The amount by which we increase these variables is given by the minimum of their upper bounds, which is 1 unit. After this, we can no longer increase variable z_2^+, so we fix it and problem (7.30) becomes:

$$\left.\begin{array}{ll} \max\limits_{(z_\kappa^+, w_\kappa^+)} & 2z_3^+ + w_3^+ - 9z_1^- - 4z_2^- - 8w_1^- - 5w_2^- - w_3^- + 129 \\ \text{s.t.} & z_3^+ + w_3^+ \leq z_1^- + z_2^- + w_1^- + w_2^- + w_3^- \\ & 0 \leq z_1^- \leq 4 \\ & 0 \leq z_2^- \leq 1 \\ & 0 \leq w_1^- \leq 6 \\ & 0 \leq w_2^- \leq 3 \\ & 0 \leq w_3^- \leq 1 \\ & z_3^+, w_3^+ \geq 0 \end{array}\right\} \quad (7.31)$$

Note that in (7.31) the constant term in the objective has increased by 7 units, from 122 to 129, where the first 4 units are due to increasing z_2^+ by the amount of current cash and the remaining 3 units are due to simultaneously increasing z_2^+ and w_3^-.

Looking now at problem (7.31), we notice that a simultaneous increase of variables z_3^+ and w_3^-, which are now the variables with the highest positive and negative coefficients in the objective respectively, by 1 unit will increase the objective by 1 unit. After this, variable w_3^- can no longer be increased and after fixing it problem (7.31) becomes:

$$\left.\begin{array}{ll} \max\limits_{(z_\kappa^+, w_\kappa^+)} & 2z_3^+ + w_3^+ - 9z_1^- - 4z_2^- - 8w_1^- - 5w_2^- + 130 \\ \text{s.t.} & z_3^+ + w_3^+ \leq z_1^- + z_2^- + w_1^- + w_2^+ \\ & 0 \leq z_1^- \leq 4 \\ & 0 \leq z_2^- \leq 1 \\ & 0 \leq w_1^- \leq 6 \\ & 0 \leq w_2^- \leq 3 \\ & z_3^+, w_3^+ \geq 0 \end{array}\right\} \quad (7.32)$$

In problem (7.32), note that, considering the allowable increases in the variables and the respective coefficients in the objective, any other simultaneous increase between the variables will lead to a smaller objective and thus we stop here by leaving all other variables equal to zero.

From the above, the optimal value of problem (7.30) is 130 units and the optimal

values of the decision variables are the ones summarized in Table (7.3).

$z^{+,\star}$	$z^{-,\star}$	$w^{+,\star}$	$w^{-,\star}$
$z_1^{+,\star} = 0$	$z_1^{-,\star} = 0$	$w_1^{+,\star} = 0$	$w_1^{-,\star} = 0$
$z_2^{+,\star} = 2$	$z_2^{-,\star} = 0$	$w_2^{+,\star} = 0$	$w_2^{-,\star} = 0$
$z_3^{+,\star} = 1$	$z_3^{-,\star} = 0$	$w_3^{+,\star} = 0$	$w_3^{-,\star} = 2$
$z^\star = z^{+,\star} - z^{-,\star} = 3$		$w^\star = w^{+,\star} - w^{-,\star} = -2$	

Table 7.3: Optimal decisions for the problem of Example 9.2

Improved Formulation of the Subproblem of PLADP

Recall that in Example 9.2 we had 2 risky assets with 3 slopes each and for each slope κ, where $\kappa = 1, 2, 3$, we had two variables, one for buying and one for selling. Thus, for asset 1 and slope 2, for example, we had variables z_2^+ and z_2^-, where the first one was for buying and the latter one was for selling. For slope segments that were filled up the optimal decisions were to set the respective z_κ^+'s and w_κ^+'s to zero, while for slope segments that were empty the optimal decisions were to set the respective z_κ^-'s and w_κ^- to zero.

Further, taking a closer look at the objective, we notice that the objective coefficients of z_κ^+'s (w_κ^+'s), which are the buying slopes, are monotone increasing with respect to κ, while the slopes of z_κ^-'s (w_κ^-), which are the selling slopes, are monotone decreasing with respect to κ. Thus, before we start increasing, for example, variable z_2^+ we must first have variable z_1^+ at its upper bound and before we start increasing variable z_3^+ we must first have variable z_2^+ at its upper bound. Similarly, before we start increasing variable z_1^- we must first have variable z_2^- at its upper bound and before we start increasing variable z_2^- we must first have variable z_3^- at its upper bound.

Considering the above, every risky asset of Example 9.2 can be replaced by three new risky assets, one for each slope segment. After introducing the new risky assets, the problem becomes similar to the subproblem of LADP-UB which is given by (6.18). The main difference is that the upper bounds on the holdings of the assets are not the same for all assets as in (6.18) but they are given by the distances between the breakpoints. For example, the new asset defined by the 2nd slope segment of asset 1 (this is the 2nd new risky asset) has an upper bound on its holdings of 3 units, the new asset defined by the 1st slope segment of asset 2 (this is the 4th new risky asset) has an upper bound of 6 units and the new assets defined by the 3rd slope segments of the two assets (these are respectively the 3rd and 6th new risky assets) are not restricted from above.

In an analogous manner, in our problem for every risky asset i we define m new assets, one for each slope segment. We let $j = (i, \kappa)$ be the new asset defined by the κ^{th} slope segment of asset i and $\tilde{\mathcal{N}}$ be the set that contains the new assets, i.e. we have $\tilde{\mathcal{N}} = \{j : j = (i, \kappa), i \in \mathcal{N}, \kappa \in \mathcal{M}\}$. The cardinality of set $\tilde{\mathcal{N}}$ is $|\tilde{\mathcal{N}}| = Nm$, i.e. set $\tilde{\mathcal{N}}$ contains Nm assets. The current holdings of every asset $j \in \tilde{\mathcal{N}}$, which we denote with $\tilde{h}_{jt}$, are given by:

$$\tilde{h}_{jt} = \begin{cases} a_{it}^{\kappa+1} - a_{it}^{\kappa}, & \text{if } \kappa \leq \pi(i,-) - 1,\ j = (i,\kappa) \\ R_{it} h_{i(t-1)}^{+} - a_{it}^{\kappa}, & \text{if } \kappa = \pi(i,-),\ j = (i,\kappa) \\ 0, & \text{if } \kappa \geq \pi(i,-) + 1,\ j = (i,\kappa) \end{cases} \tag{7.33}$$

and its upper bound, which we denote with $\tilde{w}_{jt}$, is given by:

$$\tilde{w}_{jt} = a_{it}^{\kappa+1} - a_{it}^{\kappa},\ j = (i,\kappa) \tag{7.34}$$

Table 7.4 summarizes the correspondence between the old variables and parameters and the new ones for every $i \in \mathcal{N}$ and $\kappa \in \mathcal{M}$.

Variables	
x_{it}^{κ}	$\tilde{x}_{jt}, j = (i,\kappa)$
y_{it}^{κ}	$\tilde{y}_{jt}, j = (i,\kappa)$
Parameters	
u_{it}^{κ}	$\tilde{u}_{jt}, j = (i,\kappa)$
k_{it}^{κ}	$\tilde{k}_{jt}, j = (i,\kappa)$
l_{it}^{κ}	$\tilde{l}_{jt}, j = (i,\kappa)$

Table 7.4: Correspondence between the old variables and parameters and the new ones

Considering the above, problem (7.29) can be now simplified as follows:

$$\left.\begin{aligned} \max_{(\tilde{x}_{it}, \tilde{y}_{it})} \quad & \sum_{i \in \tilde{\mathcal{N}}} \tilde{k}_{it} \tilde{x}_{it} + \sum_{i \in \tilde{\mathcal{N}}} \tilde{l}_{it} \tilde{y}_{it} + \sum_{i \in \tilde{\mathcal{N}}} \tilde{u}_{it} \tilde{h}_{it} + u_{0t} R_{0t} h_{0(t-1)}^{+} \\ \text{s.t.} \quad & \sum_{i \in \tilde{\mathcal{N}}} \tilde{x}_{it} \leq \frac{1}{1+\theta} R_{0t} h_{0(t-1)}^{+} + \frac{1-\theta}{1+\theta} \sum_{i \in \tilde{\mathcal{N}}} \tilde{y}_{it} \\ & \tilde{y}_{it} \leq \tilde{h}_{it},\ \ i \in \tilde{\mathcal{N}} \\ & \tilde{x}_{it} \leq \tilde{w}_{it} - \tilde{h}_{it},\ \ i \in \tilde{\mathcal{N}} \\ & \tilde{x}_{it}, \tilde{y}_{it} \geq 0,\ \ i \in \tilde{\mathcal{N}} \end{aligned}\right\} \tag{7.35}$$

Note that as in chapter 6, the buying and the selling slopes of the assets in set $\tilde{\mathcal{N}}$ allow the classification of the new assets into one of the categories (sets) of

Figure (6.2), which are the Buy-assets, the Sell-assets, the Neutral-assets and Cash. Further, we have the transformation and projection coefficients of Tables 6.1 and 6.2.

Solution

To solve problem (7.9) of Example 9.2, which as explained above is an instance of problem (7.35) without transaction costs, we began with noting that z_κ^+'s and w_κ^+'s are constrained by z_κ^-'s and w_κ^-'s from above and we proceeded as follows:

- First, we set all variables equal to zero.
- Second, we checked the coefficients of z_κ^-'s and w_κ^-'s in the objective and since these were all negative initially we did not increase any of them.
- Third, we checked the coefficients of z_κ^+'s and w_κ^+'s and increased the one with the highest positive coefficient in the objective by the amount of current cash since this was within the variable's upper bound.
- Fourth, after we exhausted current cash, we considered simultaneous increases of one of z_κ^-'s and w_κ^-'s and one of z_κ^+'s and w_κ^+'s and every time we increased the pair of variables that had the highest positive contribution in the objective.
- Finally, we stopped when there was no other pair of variables that could achieve a positive contribution in the objective.

In a similar manner, we will now solve problem (7.35). For this, we will need the following set:

- $\mathcal{I}$ = Sell-assets & $\Big(\tilde{h}_{it} > 0\Big)$: This set contains the Sell-assets with positive holdings (these assets will be sold).

We propose the following algorithm:

Begin with setting $\tilde{x}_{it}, \tilde{y}_{it} := 0$ for every asset $i \in \tilde{\mathcal{N}}$, assign risky assets to one of the non-overlapping categories of Figure 6.2, initialize set $\mathcal{I}$ and pick asset $j^\star$ by setting:

$$j^\star := \left\{ \arg\max_{i \in \tilde{\mathcal{N}}} \tilde{k}_{it} : \ \Big(\tilde{k}_{it} > 0\Big) \ \& \ \Big(\tilde{h}_{it} < \tilde{w}_{it}\Big) \right\} \tag{7.36}$$

This is Step 0.

If set $\mathcal{I}$ is non-empty, then take every asset i of this set, sell $\tilde{h}_{it}$ units and update cash by setting:

$$\begin{aligned} \tilde{y}_{it} &:= \tilde{h}_{it} \, \forall i \in \mathcal{I} \\ h_{0t} &:= h_{0t} + (1-\theta) \sum_{i \in \mathcal{I}} \tilde{h}_{it} \end{aligned} \tag{7.37}$$

This is Step 1.

Next, after Step 1, if there exist an asset $j^\star$ and cash, then buy $\min\left(\frac{1}{1+\theta} h_{0t}, \tilde{w}_{j^\star t} - \tilde{h}_{j^\star t}\right)$ units of asset $j^\star$ and update cash by setting:

$$\begin{aligned} \tilde{x}_{j^\star t} &:= \min\left(\frac{1}{1+\theta} h_{0t}, \tilde{w}_{j^\star t} - \tilde{h}_{j^\star t}\right), \\ h_{0t} &:= h_{0t} - (1+\theta) \min\left(\frac{1}{1+\theta} h_{0t}, \tilde{w}_{j^\star t} - \tilde{h}_{j^\star t}\right) \end{aligned} \tag{7.38}$$

Every time asset $j^\star$ is filled up, i.e. whenever $\min\left(\frac{1}{1+\theta} h_{0t}, \tilde{w}_{j^\star t} - \tilde{h}_{j^\star t}\right) = \tilde{w}_{j^\star t} - \tilde{h}_{j^\star t}$, pick another asset $j^\star$ using (7.36) and continue buying using the new asset $j^\star$, if any.

This is Step 2 and it terminates either when there exists no other asset $j^\star$ or when we run out of cash.

Then, after Step 2 and given that there exists an asset $j^\star$, if there exist Buy- or Neutral-assets with positive holdings such that amount $\frac{1-\theta}{1+\theta} \tilde{k}_{j^\star t} + \tilde{l}_{it}$ is positive, then among these assets select the one that maximizes this amount by setting:

$$i^\star := \left\{ \arg\max_{i \in \tilde{\mathcal{N}}} \tilde{l}_{it} : \left(\tilde{l}_{it} \leq 0\right) \ \& \ \left(\tilde{h}_{it} > 0\right) \ \& \ \left(\frac{1-\theta}{1+\theta} \tilde{k}_{j^\star t} + \tilde{l}_{it} > 0\right) \right\}, \tag{7.39}$$

and simultaneously increase assets $i^\star$ and $j^\star$ by setting:

$$\begin{aligned} \tilde{y}_{i^\star t} &:= \tilde{y}_{i^\star t} + \min\left(\tilde{h}_{i^\star t}, \frac{1+\theta}{1-\theta}\left(\tilde{w}_{j^\star t} - \tilde{h}_{j^\star t}\right)\right), \\ \tilde{x}_{j^\star t} &:= \tilde{x}_{j^\star t} + \frac{1-\theta}{1+\theta} \min\left(\tilde{h}_{i^\star t}, \frac{1+\theta}{1-\theta}\left(\tilde{w}_{j^\star t} - \tilde{h}_{j^\star t}\right)\right), \\ \tilde{h}_{i^\star t} &:= \tilde{h}_{i^\star t} - \tilde{y}_{i^\star t}, \\ \tilde{h}_{j^\star t} &:= \tilde{h}_{j^\star t} + \tilde{x}_{j^\star t} \end{aligned} \tag{7.40}$$

If the minimization in (7.40) is given by the first term (this is when we run out of wealth in asset $i^\star$), then pick another asset $i^\star$ using (7.39) and continue simultane-

ously selling the new asset $i^\star$ and buying asset $j^\star$ using (7.40). If the minimization in (7.40) is given by the second term (this is when asset $j^\star$ is filled up), then pick another asset $j^\star$ using (7.36) and continue simultaneously selling asset $i^\star$ and buying the new asset $j^\star$ using (7.40).

This is Step 3 and it terminates when there exists no other asset $i^\star$ or $j^\star$.

Note that we can return to the original buying and selling variables x_{it} and y_{it} using:

$$\begin{aligned} x_{it} &= \sum_{j:j=(i,\kappa)\in\tilde{\mathcal{N}}} \tilde{x}_{jt}, \;\; i \in \mathcal{N}, \kappa \in \mathcal{M} \\ y_{it} &= \sum_{j:j=(i,\kappa)\in\tilde{\mathcal{N}}} \tilde{y}_{jt}, \;\; i \in \mathcal{N}, \kappa \in \mathcal{M} \end{aligned} \tag{7.41}$$

Algorithm 7.1 summarizes the proposed algorithm to solve problem (7.35).

Algorithm 7.1 Allocation Algorithm for Piecewise Linear Approximation

Input: h_{0t}, $\tilde{k}_{it}$, $\tilde{l}_{it}$,$\tilde{h}_{it}$,$\tilde{w}_{it}$ $\forall i \in \tilde{\mathcal{N}}$

Step 0. Initialization:
set $\tilde{x}_{it}, \tilde{y}_{it} := 0 \; \forall i \in \tilde{\mathcal{N}}$
set $\mathcal{I} :=$ Sell-assets $\& \; \left(\tilde{h}_{it} > 0\right)$
set $j^\star := \left\{\arg\max_{i\in\tilde{\mathcal{N}}} \tilde{k}_{it} : \; \left(\tilde{k}_{it} > 0\right) \,\&\, \left(\tilde{h}_{it} < \tilde{w}_{it}\right)\right\}$

Step 1. Sell assets and update cash:
while $\mathcal{I} \neq \emptyset$ **then**
  set $\tilde{y}_{it} := \tilde{h}_{it} \; \forall i \in \mathcal{I}$
  set $h_{0t} := h_{0t} + (1-\theta)\sum_{i\in\mathcal{I}} \tilde{h}_{it}$
end

Step 2. Buy with cash:
while $(j \neq \emptyset) \;\&\; (h_{0t} > 0)$ **do**
  set $\tilde{x}_{j^\star t} := \min\left(\frac{1}{1+\theta}h_{0t}, \tilde{w}_{j^\star t} - \tilde{h}_{j^\star t}\right)$
  set $h_{0t} := h_{0t} - (1+\theta)\min\left(\frac{1}{1+\theta}h_{0t}, \tilde{w}_{j^\star t} - \tilde{h}_{j^\star t}\right)$
  if $\min\left(\frac{1}{1+\theta}h_{0t}, \tilde{w}_{j^\star t} - \tilde{h}_{j^\star t}\right) = \tilde{w}_{j^\star t} - \tilde{h}_{j^\star t}$ **then**
    set $j^\star := \left\{\arg\max_{i\in\tilde{\mathcal{N}}} \tilde{k}_{it} : \; \left(\tilde{k}_{it} > 0\right) \,\&\, \left(\tilde{h}_{it} < \tilde{w}_{it}\right)\right\}$
    set $\tilde{h}_{j^\star t} := \tilde{w}_{j^\star t}$
  else
    set $h_{0t} := 0$
  end

end

Step 3. Perform sell-to-buy-transactions:
set $i^\star := \left\{\arg\max_{i\in\tilde{\mathcal{N}}} \tilde{l}_{it} : \left(\tilde{l}_{it} \leq 0\right) \ \& \ \left(\tilde{h}_{it} > 0\right) \ \& \ \left(\frac{1-\theta}{1+\theta}\tilde{k}_{j^\star t} + \tilde{l}_{it} > 0\right)\right\}$
while $(i^\star \neq \emptyset) \ \& \ (j^\star \neq \emptyset)$ **do**
  set $\tilde{y}_{i^\star t} := \tilde{y}_{i^\star t} + \min\left(\tilde{h}_{i^\star t}, \frac{1+\theta}{1-\theta}\left(\tilde{w}_{it} - \tilde{h}_{j^\star t}\right)\right)$
  set $\tilde{x}_{j^\star t} := \tilde{x}_{j^\star t} + \frac{1-\theta}{1+\theta}\min\left(\tilde{h}_{i^\star t}, \frac{1+\theta}{1-\theta}\left(\tilde{w}_{it} - \tilde{h}_{j^\star t}\right)\right)$
  set $\tilde{h}_{i^\star t} := \tilde{h}_{i^\star t} - y_{i^\star t}$
  set $\tilde{h}_{j^\star t} := \tilde{h}_{j^\star t} + x_{j^\star t}$
  if $\min\left(\tilde{h}_{i^\star t}, \frac{1+\theta}{1-\theta}\left(\tilde{w}_{it} - \tilde{h}_{j^\star t}\right)\right) = \tilde{h}_{i^\star t}$ **then**
    set $i^\star := \left\{\arg\max_{i\in\tilde{\mathcal{N}}} \tilde{l}_{it} : \left(\tilde{l}_{it} \leq 0\right) \ \& \ \left(\tilde{h}_{it} > 0\right) \ \& \ \left(\frac{1-\theta}{1+\theta}\tilde{k}_{j^\star t} + \tilde{l}_{it} > 0\right)\right\}$
  else
    set $j^\star := \left\{\arg\max_{i\in\tilde{\mathcal{N}}} \tilde{k}_{it} : \left(\tilde{k}_{it} > 0\right) \& \left(\tilde{h}_{it} < \tilde{w}_{it}\right)\right\}$
  end
end

Output: $\tilde{x}_{it}, \tilde{y}_{it} \ \forall i \in \tilde{\mathcal{N}}$

Theorem 7.2.1. *Algorithm 7.1 solves problem (7.29) optimally.*

Proof

The proof is analogous to the proof of theorem 6.3.1 of chapter 6. The main difference here is that we do not have Forced-to-sell-assets since the holdings of the assets never exceed their upper bounds, which now differ from one asset to another and are determined by the breakpoints. ■

Considering the above, every subproblem of PLADP with N risky assets is similar to every subproblem of LADP-UB with Nm risky assets. Thus, depending on the transactions performed by Algorithm 7.1, we will have one of the cases of the sketch of the optimal solution provided in chapter 6 for the subproblem of LADP-UB, where the selling and buying decisions for each new risky asset will be given respectively by (6.26) and (6.27) if no buying occurred, (6.28) and (6.29) if buying occurred but we had to stop because all Buy-assets were filled up, and (6.30) and (6.31) if buying occurred but we had to stop because we had no more resources. Note that here set $\bar{\mathcal{F}}$ is empty since we do not have Forced-to-sell-assets.

We will now provide an example to illustrate how we perform selling and buying using Algorithm 7.1.

Example 7.3. Suppose we have three stocks with buying slopes, selling slopes, holdings and cash as shown in Figure 7.7.

Figure 7.8 shows the updated wealth in all assets after applying Step 1 of Algorithm 7.1, where, due to the second and third selling slopes of stock 2 being positive, we sell $h_{2t} - a^2_{2t}$ (monetary) units of stock 2. This increases cash by $(1-\theta)\left(h_{2t} - a^2_{2t}\right)$ (monetary) units.

Figure 7.9 shows the updated wealth in all assets after applying Step 2 of Algorithm 7.1, where we use all cash to buy stock 1. This increases the holdings of stock 1 by $\frac{h_{0t}+(1-\theta)\left(h_{2t}-a^2_{2t}\right)}{1+\theta}$ (monetary) units.

Finally, Figures 7.10 and 7.11 show how the wealth in assets changes after applying Step 3 of Algorithm 7.1, where, assuming that $l^3_{3t} > l^1_{2t}$, $k^3_{1t} + \frac{1-\theta}{1+\theta} l^3_{3t} > 0$ and $k^3_{1t} + \frac{1-\theta}{1+\theta} l^1_{2t} > 0$, we first sell $h_{3t} - a^2_{3t}$ (monetary) units of stock 3 and then we sell a^2_{2t} (monetary) units of stock 2 to buy stock 1. This increases the holdings of stock 1 by $\frac{1-\theta}{1+\theta}\left(h_{3t} - a^2_{3t}\right) + \frac{1-\theta}{1+\theta} a^2_{2t}$ (monetary) units.

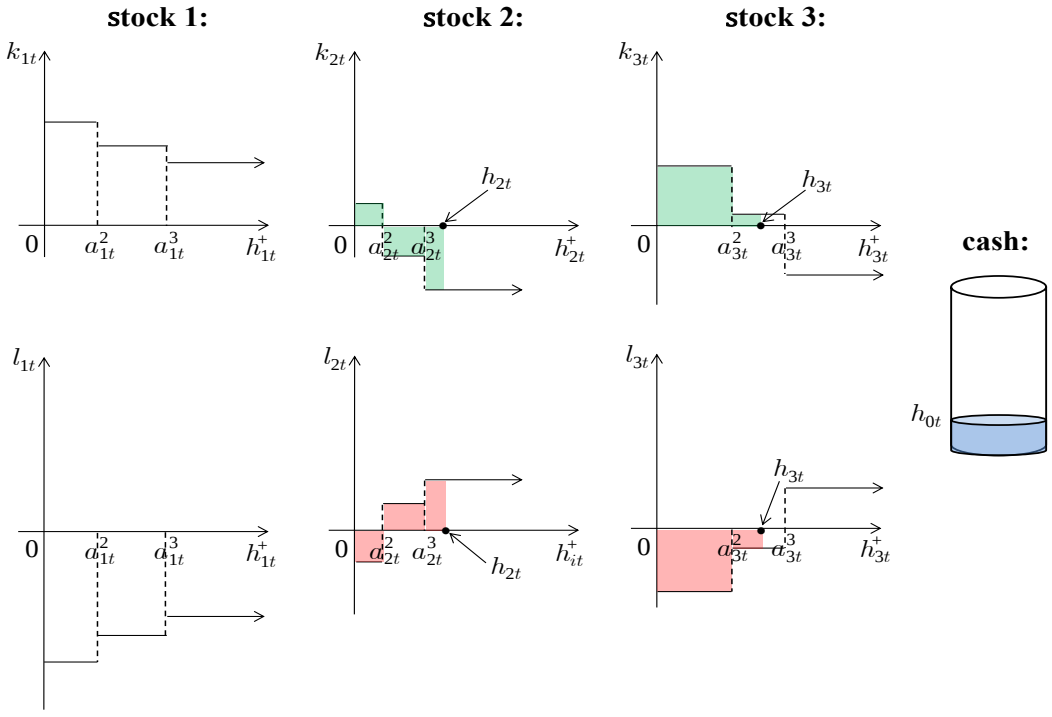


Figure 7.7: Buying and selling slopes vs current holdings before allocation

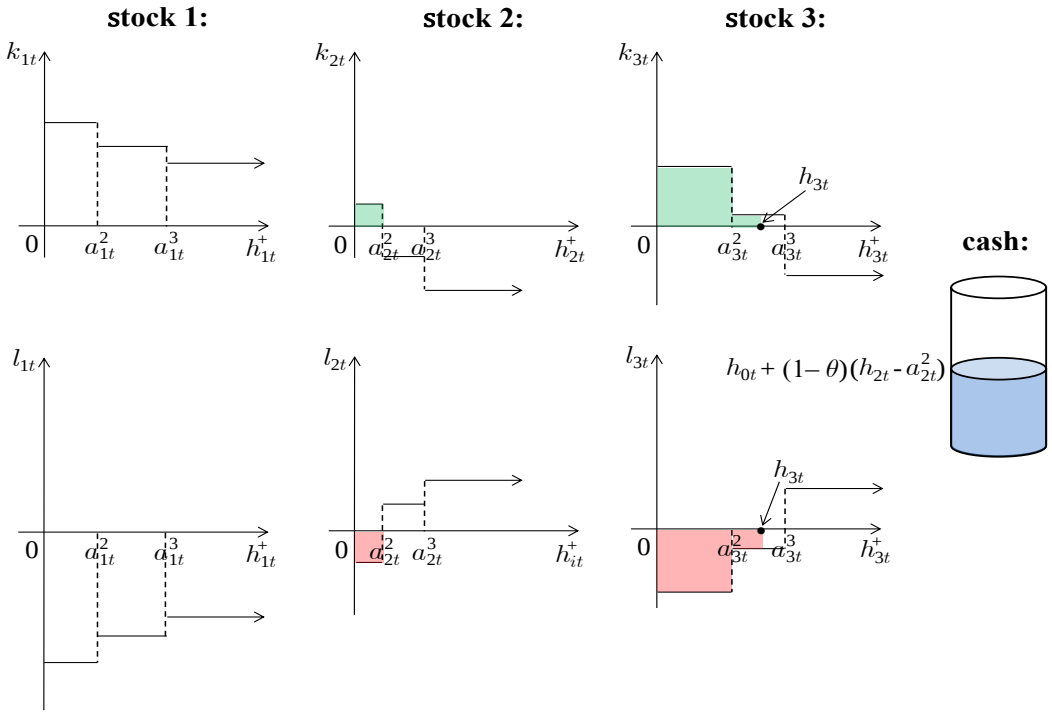


Figure 7.8: Step 1. Sell stock 2 and update cash

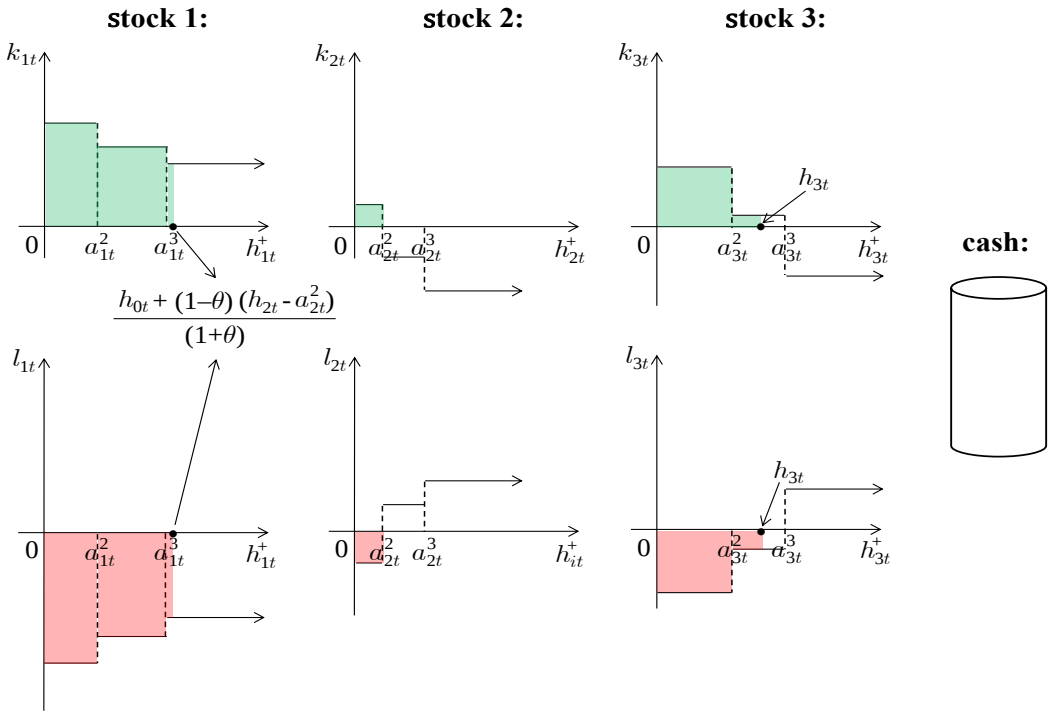


Figure 7.9: Step 2. Buy stock 1 with cash

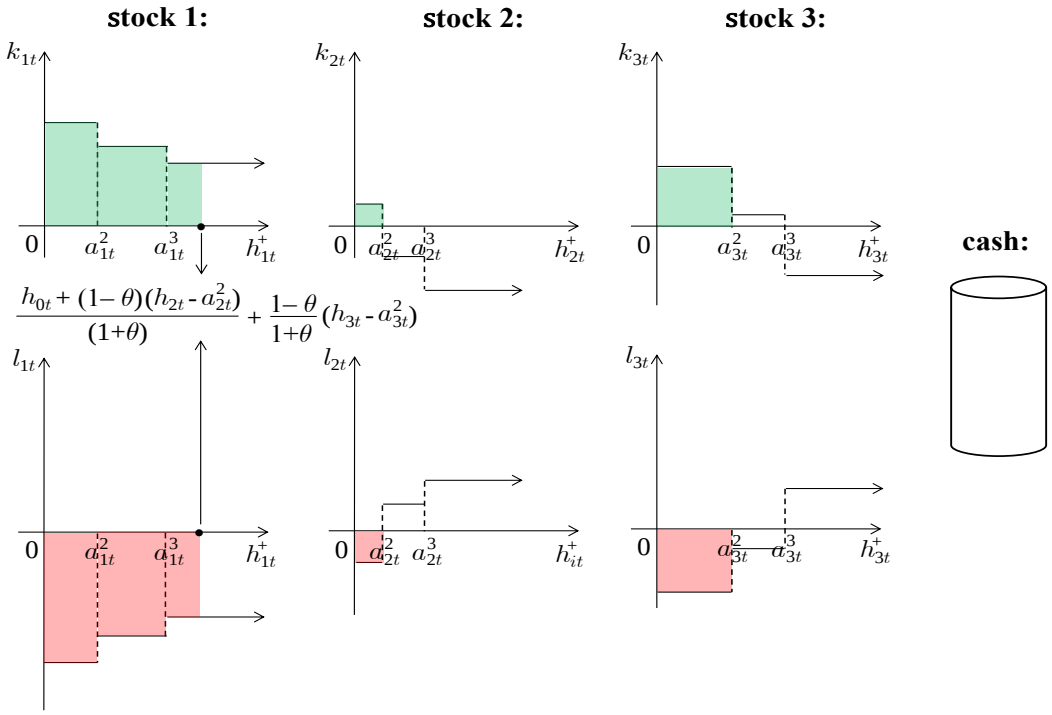


Figure 7.10: Step 3a. Sell stock 3 and buy stock 1

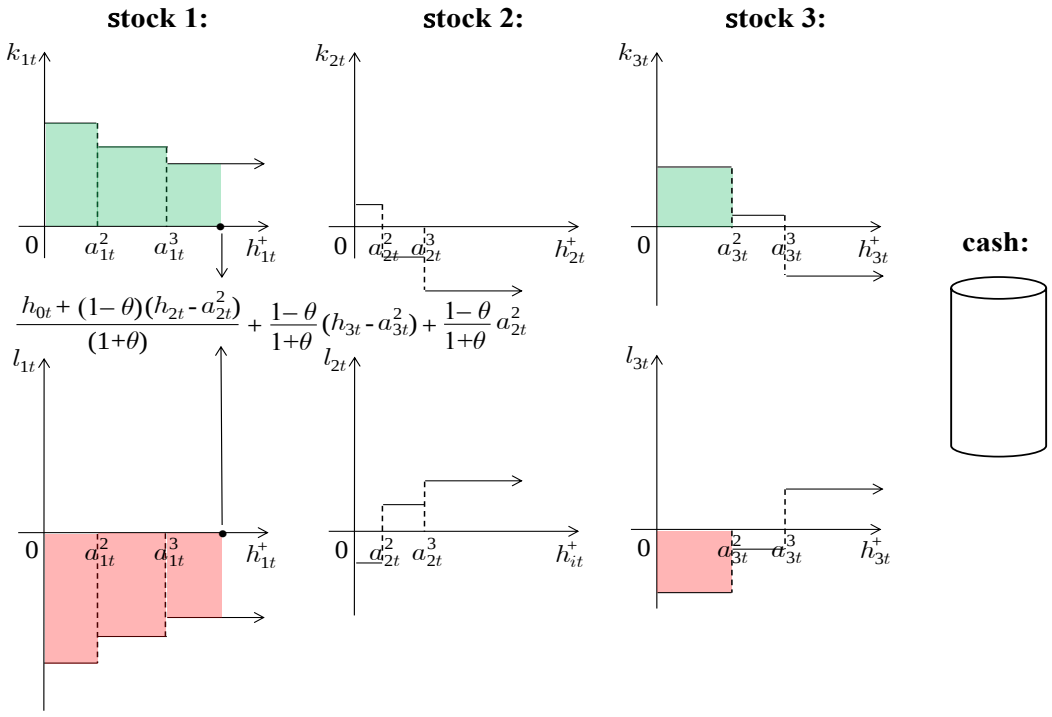


Figure 7.11: Step 3b. Sell stock 2 and buy stock 1

Complexity

The linear program (7.35), which we solve with Algorithm 7.1, has $2Nm$ non-negative decision variables, one fixed upper bound for each variable and the budget constraint which gives us the relationship between all variables.

Algorithm 7.1 which solves the subproblem of PLADP (7.35) differs from Algorithm 6.2 which solves the subproblem of LADP-UB (6.18) in Step 1 where in the latter we additionally sell the Forced-to-sell-assets. Therefore, as in Algorithm 6.2, where we have a time complexity of $O(N \log N)$ which comes from sorting N assets, in Algorithm 7.1 we have a time complexity of $O(Nm \log(Nm))$ which comes from sorting Nm assets. For S T-period realizations of the stochastic process of random returns, the above gives us a polynomial time complexity of $S \cdot T \cdot O(Nm \log(Nm)) + \sum_{s=1}^{S} slogs$ for the PLADP algorithm (see section 4.3 for a discussion on the complexity of the General ADP algorithm).

Discussion

Note the following:

1. Similarly to the linear approximation with upper bounds in chapter 6:
   (a) In Algorithm 7.1, none of the steps requires that we first visit the previous steps.
   (b) We have the transformation and projection coefficients of Tables 6.1 and 6.2.
   (c) While solving problem (6.18) using Algorithm 6.2, if we have multiple optimal allocations we resolve this tie by requiring that wealth is equally split among all the assets that have the same highest positive buying slope. However, as explained in chapter 6, due to the random returns we will never be faced with assets that have the same buying slopes.
   (d) Changing the order of buying in Algorithm 7.1 could result in a suboptimal allocation.
2. Unlike the linear approximation with upper bounds of chapter 6, where we had upper bound constraints on the holdings of the assets which forced us to sell the assets with holdings exceeding the respective upper bounds, here the upper bounds on the holdings of the assets are determined naturally by the breakpoints.

3. The observed slope of each original asset i is given by the observed slope of the new asset that is defined by its active right slope and we compute it using the gradients of Table 6.4. To compute observed slopes $\Delta\tilde{V}^{s}_{i(T-1)}$ we use (4.41).

7.3 Update Rule for Value Functions

In this section, we propose an update rule for the piecewise linear value functions to use in Step 3 of the General ADP algorithm (see Algorithm 4.1).

In the literature, there exist value function update rules that seem to work considerably well in practice. For problems with continuous state spaces, see for example the update rules in CAVE [34] [35] and SPAR-Mutual [60] algorithms. For problems with discrete state spaces, see for example the update rules in [50], [68], [81] and for an overview of them see tutorial [67].

In applications that use the above update rules, the authors start in the first iteration of the approximate dynamic programming algorithms either with discrete monotone decreasing initial slope estimates with fixed breakpoints or with flat initial slope estimates and they let the algorithm decide the number of slopes and breakpoints. Regarding the slopes, in both cases these are updated using gradient information and ensuring that the monotonicity of the slopes (i.e. concavity) is preserved. Regarding the breakpoints, in the first case we already have them from the first iteration, while in the latter case these are defined by the states visited in every iteration of the approximate dynamic programming algorithm. Thus, updating the piecewise linear value functions in every iteration of an approximate dynamic programming algorithm involves updating both the slopes and the breakpoints of the respective approximate value functions.

However, note that the above updates of slopes and breakpoints mean that in the first case, where we have discrete monotone decreasing initial slope estimates with fixed breakpoints, we will start in the first iteration with a large number of slopes and breakpoints, while in the latter case, where we have flat initial slope estimates, the number of slopes and breakpoints will increase quickly with the number of iterations and as a result the size of the subproblems of PLADP will grow significantly from the very early iterations. Due to this, the subproblems of PLADP, which are given by (7.35), grow considerably in size and can become computationally intractable. To circumvent this, we propose an update rule for Step 3 of the General ADP algorithm, where we require that the number of slopes in the approx-

imate value functions remains below a specified threshold at every iteration, thus maintaining problem (7.35) tractable.

For the above update methods, in problems with discrete state spaces there exist convergence results but in very limited settings where it is being assumed that the planning horizon consists of only two time periods and the state vector has only one dimension. In these problems, authors prove convergence by showing that if every state is visited infinitely many times, then the slope of the last iteration converges to its optimal value. For multi-dimensional (continuous) problems the above update rules are only approximate methods that seem to work well in practical applications.

In this study, at every iteration of the approximate dynamic programming algorithm we use the update rule of [68] (described below) to preserve monotonicity of the slopes and we propose the use of a correction rule to maintain the number of slopes below a specified threshold. In the discussion that follows, we begin with a short description of the update rule in [68] for problems with discrete state spaces. We discuss how this rule can be used in problems with continuous state spaces, such as the problem under study, and the we describe the correction rule. Finally, we conclude with the slopes update and correction routine, where we put together the update and correction rules.

Update Rule for Dynamic Programs with Discrete State Space

Here, we briefly discuss the update rule of [68], which was initially developed for problems with discrete state space, where the points of non-differentiability in the approximate value functions are subsets of non-negative integers. In the discussion that follows, we assume that currently we are at iteration s, we have visited state $h_{it}^{s,+}$ which is a discrete variable and we want to estimate the κ^{th} slope of function $\hat{V}_{it}^{s}$ which we denote with $\hat{u}_{it}^{s,\kappa}$ and will be used at iteration $s+1$.

In line with [68], function $\hat{V}_{it}^{s}$ can be represented by the sequence of slopes $\{\hat{u}_{it}^{s,\kappa} : \kappa = 1, 2, \ldots, m\}$, which are given by:

$$\hat{u}_{it}^{s,q} = \begin{cases} (1-\alpha_{it}^{s})\, \hat{u}_{it}^{s-1,\kappa} + \alpha_{it}^{s} \Delta\tilde{V}_{it}^{s}, & \text{if } \kappa = h_{it}^{s,+} + 1 \\ \hat{u}_{it}^{s-1,\kappa}, & \text{if } \kappa \in \left\{1, \ldots, h_{it}^{s,+}, h_{it}^{s,+} + 2, \ldots, m\right\} \end{cases} \tag{7.42}$$

where as explained in chapter 6 $\Delta\tilde{V}_{it}^{s}$ is the observed slope which we obtain using gradient information. Note that in (7.42) we update the slope that corresponds to state $h_{it}^{s,+} + 1$ (this is the first slope to the right of current state $h_{it}^{s,+}$) using the observed slope and leave the remaining slopes the same.

However, note that updating only slope $\hat{u}_{it}^{s-1,\kappa}$ for $\kappa = h_{it}^{s,+} + 1$ might lead to a violation of concavity if either $\hat{u}_{it}^{s,\kappa-1} < \hat{u}_{it}^{s,\kappa}$ or $\hat{u}_{it}^{s,\kappa} < \hat{u}_{it}^{s,\kappa+1}$. To correct monotonicity, it is proposed in [68] (see [50] and [81] for other methods) that the sequence of slopes $\{\hat{u}_{it}^{s,q} : \kappa = 1, 2, \ldots, m\}$ is chosen by applying a projection operation that involves solving the following non-linear minimization problem:

$$\left.\begin{aligned} \min_{z_\kappa} \sum_{\kappa=1}^{m} [z_\kappa - \hat{u}_{it}^{s,\kappa}]^2 \\ \text{s.t. } z_\kappa - z_{\kappa+1} \geq 0, \ \ \kappa \in \{1, 2, \ldots, m\} \end{aligned}\right\} \tag{7.43}$$

In (7.43), we replace the sequence of slopes $\{\hat{u}_{it}^{s,\kappa} : \kappa = 1, 2, \ldots, m\}$ with a new sequence of slopes, which we denote with $\{z_\kappa : \kappa = 1, 2, \ldots, m\}$ and so that the sum of the squared vertical distances of the new slopes from the old ones is minimized. Using the Karush-Kuhn-Tucker (KKT) conditions, we can come up with an optimal sequence of slopes $\{z_\kappa : \kappa = 1, 2, \ldots, m\}$ for the projection described in (7.43). For the derivation of the optimal sequence of slopes, we refer the reader to [68]. Here, we only provide a sketch of the optimal solution which breaks down in the following three cases:

- **Case 1:** After the update in (7.42), monotonicity is satisfied, i.e. $\hat{u}_{it}^{s,1} \geq \hat{u}_{it}^{s,2} \geq \ldots \geq \hat{u}_{it}^{s,m}$. In this case, the slopes remain the same, i.e. we have:

$$z_\kappa = \hat{u}_{it}^{s,\kappa}, \ \ \forall \kappa \in \{1, 2, \ldots, m\} \tag{7.44}$$

- **Case 2:** After the update in (7.42) and for $\kappa = h_{it}^{s,+} + 1$, monotonicity is violated due to $\hat{u}_{it}^{s,\kappa-1} < \hat{u}_{it}^{s,\kappa}$. In this case, all slopes that lie at the right of slope $\hat{u}_{it}^{s,\kappa}$ remain the same, while for the remaining slopes we have: Starting from slope $\hat{u}_{it}^{s,\kappa}$, we move to the left replacing as we go the respective slopes with the average of the slopes until monotonicity with previous slope is satisfied (for an example of correcting monotonicity in this case see Example 9.4 below). If slope $\kappa^\star$ is the last slope to the left that has been replaced by this average, then the new slopes $\{z_\kappa : \kappa = 1, 2, \ldots, m\}$ are given by:

$$z_\kappa = \begin{cases} \frac{1}{h_{it}^{s,+} - \kappa^\star + 2} \sum_{\kappa=\kappa^\star}^{h_{it}^{s,+}+1} \hat{u}_{it}^{s,\kappa}, & \text{if } \kappa \in \left\{\kappa^\star, \ldots, h_{it}^{s,+} + 1\right\} \\ \hat{u}_{it}^{s,\kappa}, & \text{if } \kappa \notin \left\{\kappa^\star, \ldots, h_{it}^{s,+} + 1\right\} \end{cases} \tag{7.45}$$

where $\kappa^\star < h_{it}^{+} + 1$.

- **Case 3:** After the update in (7.42) and for $\kappa = h_{it}^{s,+} + 1$, monotonicity is violated due to $\hat{u}_{it}^{s,\kappa} < \hat{u}_{it}^{s,\kappa+1}$. In this case, all slopes that lie at the left of slope $\hat{u}_{it}^{s,\kappa}$ remain the same, while for the remaining slopes we have: Starting from slope $\hat{u}_{it}^{s,\kappa}$, we move to the right replacing as we go the respective slopes with the average of the slopes until monotonicity with next slope is satisfied. If slope $\kappa^\star$ is the last slope to the right that has been replaced by this average, then the new slopes $\{z_\kappa : \kappa = 1, 2, \ldots, m\}$ are given by:

$$z_\kappa = \begin{cases} \frac{1}{\kappa^\star - h_{it}^{s,+}} \sum_{\kappa=h_{it}^{s,+}+1}^{\kappa^\star} \hat{u}_{it}^{s,\kappa}, & \text{if } \kappa \in \left\{h_{it}^{s,+}+1, \ldots, \kappa^\star\right\} \\ \hat{u}_{it}^{s,\kappa}, & \text{if } \kappa \notin \left\{h_{it}^{s,+}+1, \ldots, \kappa^\star\right\} \end{cases} \tag{7.46}$$

  where $\kappa^\star > h_{it}^{+} + 1$.

In the following example, we illustrate how projection operation is performed in problems with discrete state space when monotonicity is violated from the left (see case 2 above).

Example 7.4. Suppose that after updating the slopes according to (7.42), we have $m = 6$ slopes and $u_{it}^{s,4} < u_{it}^{s,5}$ (see top part of Figure 7.12 below).

Here, we need to correct the slopes that lie at the left of slope $u_{it}^{s,5}$ including slope $u_{it}^{s,5}$. These are slopes $\{u_{it}^{s,q} : q = 1, 2, 3, 4, 5\}$. Assuming that $u_{it}^{s,2} > \frac{u_{it}^{s,3}+u_{it}^{s,4}+u_{it}^{s,5}}{3}$ (i.e. the second slope satisfies monotonicity with the average of the next three slope) and $u_{it}^{s,3} < \frac{u_{it}^{s,4}+u_{it}^{s,5}}{2}$ (i.e. the third slope violates monotonicity with the average of the next two slopes), in the bottom part of Figure 7.12 we have replaced every slope of $u_{it}^{s,3}, u_{it}^{s,4}$ and $u_{it}^{s,5}$ with the average of the three.

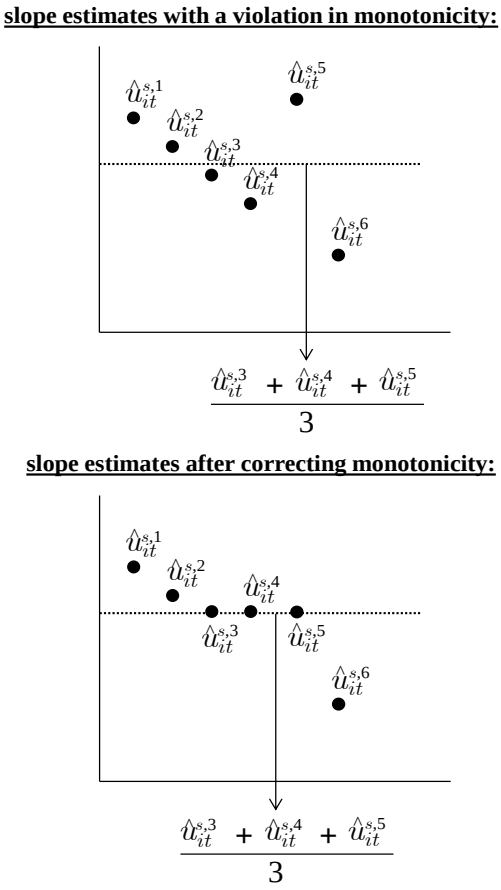


Figure 7.12: Update rule for problems with discrete state space and a violation in the monotonicity of the slopes from the left

Update Rule for Dynamic Programs with Continuous State Space

We are now ready to state the analogous projection operation rule for problems with continuous action spaces, where the points of non-differentiability in function $\hat{V}_{it}^{s}$ are a subset of positive real numbers. Function $\hat{V}_{it}^{s}$ is now represented by the sequence of slopes $\{\hat{u}_{it}^{s,\kappa} : \kappa = 1, 2, \ldots, m\}$, where slope $\hat{u}_{it}^{s,\kappa}$ is defined by breakpoints $a_{it}^{s,\kappa}$ and $a_{it}^{s,\kappa+1}$. Given that state variable $h_{it}^{s,+}$ is known when we update slopes $\hat{u}_{it}^{s-1}$, if κ^{+} denotes the part of the slope segment that corresponds to holdings $h_{it}^{s,+}$ and lies at the right of holdings $h_{it}^{s,+}$, then update rule (7.42) can be expressed as follows:

$$\hat{u}_{it}^{s,\kappa} = \begin{cases} (1-\alpha_{it}^{s})\,\hat{u}_{it}^{s-1,\kappa} + \alpha_{it}^{s}\Delta\tilde{V}_{it}^{s}, & \text{if } \kappa = \kappa^{+} \\ \hat{u}_{it}^{s-1,q}, & \text{if } \kappa \neq \kappa^{+} \end{cases} \tag{7.47}$$

Note that if holdings $h_{it}^{s,+}$ do not coincide with one of the breakpoints of function $\hat{V}_{it}^{s-1}$, then update rule (7.47) leads to one additional new slope for function $\hat{V}_{it}^{s}$. To account for this, we denote the number of slopes after applying update rule (7.47) with Q, where:

$$Q = \begin{cases} m+1, & \text{if } h_{it}^{s,+} \text{ is between breakpoints} \\ m, & \text{otherwise} \end{cases} \tag{7.48}$$

The analogous solution to the projection operation described by problem (7.43) can now be stated as follows:

- **Case 1:** After the update in (7.47), monotonicity is satisfied, i.e. $\hat{u}_{it}^{s,1} \geq \hat{u}_{it}^{s,2} \geq \ldots \geq \hat{u}_{it}^{s,m}$, and all slopes remain the same:

$$z_\kappa = \hat{u}_{it}^{s,\kappa}, \ \ \forall \kappa \in \{1, 2, \ldots, m\} \tag{7.49}$$

- **Case 2:** After the update in (7.47), monotonicity is violated due to $\hat{u}_{it}^{s,\kappa^+-1} < \hat{u}_{it}^{s,\kappa^+}$ (for an example of correcting monotonicity in this case see Example 9.5 below), and the new slopes $\{z_\kappa : \ \kappa = 1, 2, \ldots, Q\}$ are given by:

$$z_\kappa = \begin{cases} \frac{1}{\kappa^+ - \kappa^\star + 1} \sum_{\kappa=\kappa^\star}^{\kappa^+} \hat{u}_{it}^{s,\kappa}, & \text{if } \kappa \in \{\kappa^\star, \ldots, \kappa^+\} \\ \hat{u}_{it}^{s,\kappa}, & \text{if } \kappa \notin \{\kappa^\star, \ldots, \kappa^+\} \end{cases} \tag{7.50}$$

  where $\kappa^\star < \kappa^+$.

- After the update in (7.47), monotonicity is violated due to $\hat{u}_{it}^{s,\kappa^+} < \hat{u}_{it}^{s,\kappa^+ + 1}$, and the new slopes $\{z_\kappa : \ \kappa = 1, 2, \ldots, Q\}$ are given by:

$$z_\kappa = \begin{cases} \frac{1}{\kappa^\star - \kappa^+ + 1} \sum_{\kappa=\kappa^+}^{\kappa^\star} \hat{u}_{it}^{s,\kappa}, & \text{if } \kappa \in \{\kappa^+, \ldots, \kappa^\star\} \\ \hat{u}_{it}^{s,\kappa}, & \text{if } \kappa \notin \{\kappa^+, \ldots, \kappa^\star\} \end{cases} \tag{7.51}$$

  where $\kappa^\star > \kappa^+$.

In the following example, we illustrate how projection operation is performed in problems with continuous state space when monotonicity is violated from the left (see case 2 above).

Example 7.5. In the top part of Figure 7.13, we have the creation of a new break point due to visiting state $h_{it}^{s,+}$ (see top left plot) and the update of the part of the slope segment that corresponds to state $h_{it}^{s,+}$ and lies at the right of the new state $h_{it}^{s,+}$ (see top right plot). Note that due to $h_{it}^{s,+}$ lying between breakpoints, after the update of the slopes we end up with one more slope so now we have 6 instead of 5 slopes.

Assuming that after the creation of the new slope we have a violation in the monotonicity due to $\hat{u}_{it}^{s,4} < \hat{u}_{it}^{s,5}$, in the bottom part of Figure 7.13 we correct this by replacing slopes $\hat{u}_{it}^{s,3}$, $\hat{u}_{it}^{s,4}$ and $\hat{u}_{it}^{s,5}$ with the average of the three.

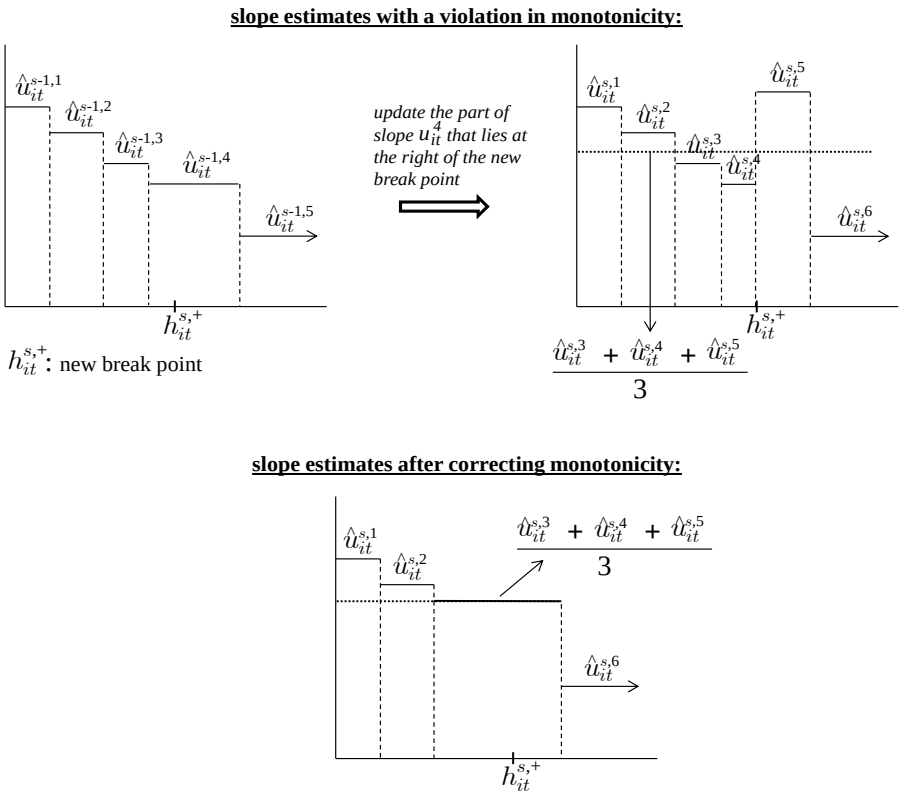


Figure 7.13: Update rule for problems with continuous state space and a violation in the monotonicity of the slopes from the left

Correction Rule for Dynamic Programs with Continuous State Space

If the number of slopes after the projection operation exceeds a specified threshold δ, then we amend the slopes by replacing the ones that are closest in value with their average. Thus, if $|Q| > \delta$ and $\hat{\kappa} = \arg\min_{\kappa \in \{1,\ldots,Q-1\}} (z_\kappa - z_{\kappa+1})$, then we

replace the sequence of slopes $\{z_\kappa : \kappa = 1, 2, \ldots, Q\}$ with a new sequence, which we denote with $\{z^\star_\kappa : \kappa = 1, 2, \ldots, Q\}$, such that:

$$z^\star_\kappa = \begin{cases} \frac{z_{\hat{\kappa}} + z_{\hat{\kappa}+1}}{2}, & \text{if } \kappa = \hat{\kappa}, \hat{\kappa}+1 \\ z_\kappa, & \text{if } \kappa \neq \hat{\kappa}, \hat{\kappa}+1 \end{cases} \tag{7.52}$$

Note that in every iteration we can have up to only one additional new slope and as a result the above replacement needs to be applied only once. Also, note that this operation does not distort the monotonicity of the slopes. An example for this operation follows.

Example 7.6. Suppose that the maximum number of slopes is set to $\delta = 5$.

In the top part of Figure 7.13, we notice that after the update of the slopes monotonicity is not violated but the number of slopes has gone up to 6.

Assuming that slopes $u^{s,5}_{it}$ and $u^{s,6}_{it}$ are the ones that are closest in value, in the bottom part of Figure 7.14 we have replaced slopes $u^{s,5}_{it}$ and $u^{s,6}_{it}$ with their average.

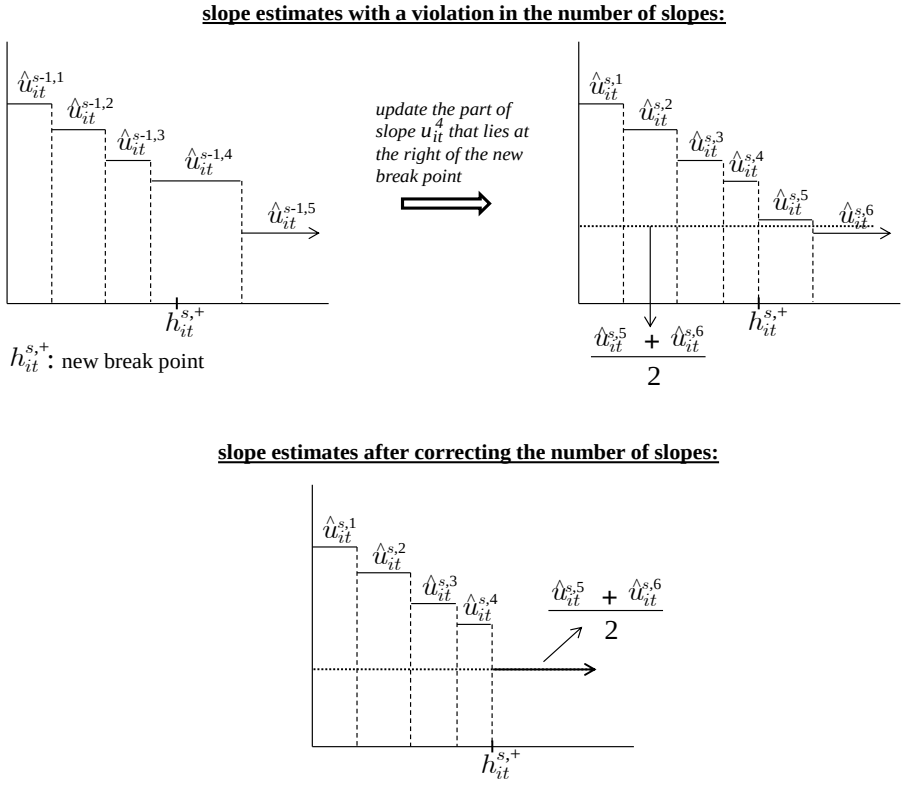


Figure 7.14: Correction rule for problems with continuous state space and a violation in the number of slopes

Slopes Update and Correction Routine for Dynamic Programs with Continuous Action Spaces

Algorithm 7.2 summarizes the slopes update and correction routine for separable piecewise linear and continuous value functions.

In Step 0, we use state $h_{it}^{s,+}$ of iteration s to create a new break point if $h_{it}^{s,+}$ lies between break points.

In Step 1, we update the part of the slope segment that corresponds to holdings $h_{it}^{s,+}$ and lies at the right of holdings $h_{it}^{s,+}$ using (7.47).

After updating the slopes, if there is a violation in the monotonicity, then in Step 2 we correct it by using projection operation (7.43).

Finally, in Step 3 and after correcting monotonicity, if the number of slopes exceeds threshold δ, then we correct the number of slopes using (7.52).

Algorithm 7.2 Slopes Update and Correction Routine

Input: $\left\{u_{it}^{s-1,\kappa} : \kappa = 1, 2, \ldots, m\right\}, \delta$

Step 0. Initialization:
insert a new break point at $h_{it}^{s,+}$
determine k^+

Step 1. Update slopes:

$$\text{set } \hat{u}_{it}^{s,\kappa} := \begin{cases} (1-\alpha_{it}^s)\,\hat{u}_{it}^{s-1,\kappa} + \alpha_{it}^s \Delta\tilde{V}_{it}^s, & \text{if } \kappa = \kappa^+ \\ \hat{u}_{it}^{s-1,\kappa}, & \text{if } \kappa \neq \kappa^+ \end{cases}$$

Step 2. Correct monotonicity:
if $\hat{u}_{it}^{s,\kappa^+-1} < \hat{u}_{it}^{s,\kappa^+}$ **then**

$$\text{set } z_\kappa := \begin{cases} \frac{1}{\kappa^+ - \kappa^\star + 1} \sum_{\kappa=\kappa^\star}^{\kappa^+} \hat{u}_{it}^{s,\kappa}, & \text{if } \kappa \in \{\kappa^\star, \ldots, \kappa^+\} \\ \hat{u}_{it}^{s,\kappa}, & \text{if } \kappa \notin \{\kappa^\star, \ldots, \kappa^+\} \end{cases}$$

elseif $\hat{u}_{it}^{s,\kappa^+} < \hat{u}_{it}^{s,\kappa^+ + 1}$ **then**

$$\text{set } z_\kappa := \begin{cases} \frac{1}{\kappa^\star - \kappa^+ + 1} \sum_{\kappa=\kappa^+}^{\kappa^\star} \hat{u}_{it}^{s,\kappa} & \text{if } \kappa \in \{\kappa^+, \ldots, \kappa^\star\} \\ \hat{u}_{it}^{s,\kappa} & \text{if } \kappa \notin \{\kappa^+, \ldots, \kappa^\star\} \end{cases}$$

else
set $z_\kappa := \hat{u}_{it}^{s,\kappa}\ \forall \kappa \in \{1, 2, \ldots, m\}$
end

Step 3. Correct the number of slopes:

if $|Q| > \delta$ **then**

set $\hat{\kappa} := \arg\min_{\kappa \in \{1,\ldots,Q-1\}} (z_\kappa - z_{\kappa+1})$

set $z^\star_\kappa := \begin{cases} \frac{z_{\hat{\kappa}} + z_{\hat{\kappa}+1}}{2}, & \text{if } \kappa = \hat{\kappa}, \hat{\kappa}+1 \\ z_\kappa, & \text{if } \kappa \neq \hat{\kappa}, \hat{\kappa}+1 \end{cases}$

else

set $z^\star_\kappa := z_\kappa \; \forall \kappa \in \{1, 2, \ldots, Q\}$

end

STOP

Output: $\{z^\star_\kappa : \; \kappa = 1, 2, \ldots, Q\}$

Part V

Experimental Results, Conclusions and Future Research

Chapter 8

Experimental Results

In this chapter, we present and analyze experimental results using real-world equity data from FTSE100 Index. Specifically, in section 8.1, we discuss how we evaluate the performance of the different portfolio selection methods and specify the output parameters to be reported for every method. Then, in section 8.2, we describe the testing environment, where we specify the data, the stepsize rules and the initial slope values used in the ADP methods, as well as the software. Finally, in section 8.3, we report the numerical results from our simulations.

8.1 Performance Evaluation Design

In this section, we begin with discussing how we evaluate the ADP methods and stating the analogous evaluation process for the other methods. Then, in subsection 8.1.1, we define the parameters that describe the composition of the selected portfolios and which will be reported for each method in the numerical results. Finally, in subsection 8.1.2 we provide our performance measures.

In approximate dynamic programming, we have two types of experiments: training and testing. Training iterations are used in order to estimate the value function approximations for all time periods, where, due to the assumed functional approximations, these are given by their slope estimates computed in Step 3 of the last iteration of the General ADP Algorithm 4.1 and are called the *terminal slope estimates*. Testing is used to evaluate for every ADP method the extracted policy, which is given by the sequence of the terminal slope estimates from time 0 up to time $T-1$. Training and testing are designed as follows:

First, we divide our data into the following two parts:

- the *in-sample data*, which is used to generate scenarios of T-period return vectors that we use in the training iterations (as we will see later on, in our experiments we use four datasets with in-sample horizon 104 weeks each), and

- the *out-of-sample data*, which consists of actual T-period return vectors that are used to test the actual performance of the extracted policies (as we will see later on, in our experiments we use four datasets with out-of-sample horizon 52 weeks each).

Then, using the in-sample data we generate scenario paths. As explained in section 2.4 and Appendix A, to generate scenarios of returns we forecast the variance-covariance matrix of the returns of the assets using the O-GARCH models.

After generating the scenario paths, we run the training iterations. In each training iteration we do a forward pass through time on one of the generated scenario paths using the General ADP Algorithm 4.1. Recall from section (4.3) that in every iteration of the General ADP Algorithm we compute buying/selling decision trajectories based on the slope estimates of the previous iteration and using these we update the slope estimates of the current iteration. In this manner, in the last iteration of the General ADP Algorithm we obtain the terminal slope estimates.

After obtaining the terminal slope estimates, we use them to perform testing, which involves performing one more iteration of the General ADP Algorithm 4.1 using the terminal slope estimates and the trajectory of the actual returns. Thus, starting at time $t = 0$ with known holdings $\boldsymbol{h}_0$, every time we solve the associated subproblem of ADP, which is given by (6.6) for the linear approximation, (6.18) for the linear approximation with strict control of flows and (7.29) for the piecewise linear approximation, and compute the respective buying and selling decisions. These are then plugged into transition equations (4.5), which now depend on the actual returns, and give us the post-decision holdings that are used to solve the subproblem of the next time period. While doing the forward pass through time on the scenario of the actual returns, we compute and store pre-decision holdings h_{it} for every $t = 1, 2, \ldots, T$ using $R_{it}h^{+}_{i(t-1)}$. In the above manner, at the end of the out-of-sample horizon we will have computed holdings $\boldsymbol{h}_T$. Note that in the testing iteration we skip Step 3 of the General ADP Algorithm 4.1, which involves updating the slopes of the current iteration, since we already have the terminal slope estimates.

In order to evaluate the performance of the other methods and compare them

against the performance of the ADP methods, the generated scenario paths that were used in the training iterations of the ADP methods are also used as an input for the other methods (except for the equally-weighted method) since in the end we want to evaluate all methods on the same input information.

In the equally-weighted method, as explained in section 5.2 we consider two investment strategies. First, the buy-and-hold strategy where we solve problem (5.19) only once in the beginning of the investment horizon and then we hold the portfolio until the end of the horizon. Second, the fixed-mix strategy where in every time period the portfolio is rebalanced to equal target proportions. We evaluate the fixed-mix strategy as follows: We start at time $t = 0$ with known holdings $\boldsymbol{h}_0$ and solve problem (5.19), which gives us holdings $\boldsymbol{h}_0^+$. Then, these holdings are multiplied with the actual returns of the first period and give us holdings $\boldsymbol{h}_1$. In a similar manner, we move forward in time solving problem (5.19) for the consequent time periods until we reach the last time period, where we compute holdings $\boldsymbol{h}_T$. Note that problem (5.19) does not depend on the scenario returns and decisions are taken without us needing to use the generated scenario paths.

In the single-period method, we start at time $t = 0$ with known holdings $\boldsymbol{h}_0$ and solve problem (5.9) using the first-period returns of the generated scenario paths. After solving problem (5.9), we obtain post-decision holdings $\boldsymbol{h}_0^+$, which is then multiplied with the actual returns of the first period and give us pre-decision holdings $\boldsymbol{h}_1$. Given now the holdings of the assets at time $t = 1$, we move forward one time period, solve problem (5.9) using the second-period returns from the scenario paths, where now the current holdings of the assets are given by $\boldsymbol{h}_1$, and obtain post-decision holdings $\boldsymbol{h}_1^+$. These are then multiplied with the actual returns of the second period and give us pre-decision holdings $\boldsymbol{h}_2$. In a similar manner, we move forward in time solving problem (5.9) for the consecutive time periods until we reach the last time period, where we compute holdings $\boldsymbol{h}_T$.

In the multistage stochastic programming method, we solve repeatedly multistage stochastic programs with smaller horizons, where every time, considering the already achieved state of the system, we solve the associated multistage stochastic program and store the first-stage decisions (see for example [6]). Thus, we start at time $t = 0$ with constructing a scenario tree for the original T-period scenario paths and then, using the generated scenario tree, we solve the deterministic equivalent of the associated $(T + 1)$-stage stochastic program, which is given by (5.16). In line with the notation of chapter 5, after solving the first multistage stochastic program, we obtain first-stage decisions $X_0 = \left(\boldsymbol{h}_0^+, \boldsymbol{x}_0, \boldsymbol{y}_0\right)$, and then as in the single-period

method we multiply post-decision holdings $\boldsymbol{h}_0^+$ with the actual returns of the first period and compute holdings $\boldsymbol{h}_1$. After the first period returns are revealed, we move forward one time period as follows: We crop the returns of the first period from the original scenario paths, and using the new scenario paths which now have length $T-1$ periods each we construct a new scenario tree with T stages. Then, using the new T-stage scenario tree we solve the subsequent T-stage stochastic program, which now gives us decisions $X_1 = \left(\boldsymbol{h}_1^+, \boldsymbol{x}_1, \boldsymbol{y}_1\right)$, and using the actual returns of the second period we compute holdings $\boldsymbol{h}_2$. The above process is repeated until we reach the last time period, where we solve a 2-stage stochastic program and compute holdings $\boldsymbol{h}_T$. The scenario trees are constructed using the scenario tree construction methods of Appendix B.

Note that the above methods are also compared against a *market Index*, which we evaluate as follows: First, using the prices of the Index in the out-of-sample data, we compute the respective total returns of the Index. Then, using these returns and starting with known wealth at time 0, every time we multiply the current total wealth with the realized market portfolio return. In this manner, we obtain cumulative wealth estimates for the market portfolio at every point in time and as a result at time T.

We divide our analysis into the following two parts:

- the *characteristics of the selected portfolios and convergence of slopes*, where for every method we report parameters related to the composition of the selected portfolios and show how slopes converge in the ADP methods, and
- the *performance evaluation*, where for every method we report the respective *out-of-sample terminal wealth*, which is the cumulative wealth achieved by the selected portfolios at the end of the out-of-sample horizon. As we will explain later on, the out-of-sample terminal wealth is our criterion as to how each method performs.

8.1.1 Characteristic of Selected Portfolios and Convergence of Slopes

Regarding the characteristics of the selected portfolios, for each portfolio selection method we report the following parameters:

1. a range for the *number of assets per out-of-sample period* (label: "NA_t") in the selected portfolios: If S_t denotes the set of assets used at time t with

cardinality $|S_t|$, then this parameter reports a range of values $\mu - \nu$, where $\mu = \min_{t=0,\ldots,T-1} |S_t|$ and $\nu = \max_{t=0,\ldots,T-1} |S_t|$,

2. the *number of different assets* (label: "NA") used throughout the out-of-sample horizon,

3. a $0-1$ variable (label: Cash) that shows whether cash is part of the selected portfolios ("Cash$= 1$") or not ("Cash$= 0$"), and

4. a range for the *normalized Herfindahl Index per out-of-sample period* (label: "HI*") to show the degree of diversification in the selected portfolios.

   The Herfindahl Index is defined as the sum of the squared weights of the assets in the portfolio (see for example [83]). With cash being one of the assets in the portfolio, if w_{it} is the weight of asset i in the portfolio at time t, then the Herfindahl Index at time t is defined as follows:

$$\text{HI}_t = \sum_{i=0}^{N} w_{it}^2, \tag{8.1}$$

   where the weights of the assets are given by $w_{it} = \frac{h_{it}}{\sum_{i=0}^{N} h_{it}}$.

   Given now that the maximum number of assets that we can have in every portfolio is $N+1$, the Herfindahl Index takes values in the range $\left[\frac{1}{N+1}, 1\right]$. To obtain a range $[0,1]$, where 0 and 1 indicate respectively perfect and no diversification, we normalize the Herfindahl Index by subtracting from it $\frac{1}{N+1}$ and dividing the resulting figure with $1 - \frac{1}{N+1}$. We denote the normalized figure with $\text{HI}_t^\star$ and compute it using:

$$\text{HI}_t^\star = \frac{\text{HI}_t - \frac{1}{N+1}}{1 - \frac{1}{N+1}} \tag{8.2}$$

   Given the above definition, if $\text{HI}_t^\star$ is the normalized Herfindahl Index at time t, then here we report a range of values $\phi - \chi$, where $\phi = \min_{t=0,\ldots,T-1} \text{HI}_t^\star$ and $\chi = \min_{t=0,\ldots,T-1} \text{HI}_t^\star$.

Additionally, for the ADP methods we provide plots that show how slopes converge with the number of iterations.

8.1.2 Performance Evaluation

Regarding performance evaluation, in the single-period portfolio optimization literature authors assess the performance of the selected portfolios by comparing the cumulative portfolio returns at all points in time during the out-of-sample horizon against the corresponding cumulative returns of the market Index (see for example [38] and [40]). In this study, however, we concentrate on multi-period portfolio selection methods and use the myopic ones for comparison reasons only. Considering that in multi-period portfolio selection methods we might need to underperform in the early periods in order to outperform in the end, we use the out-of-sample terminal wealth as our *criterion* to assess how well every method is performing and we report the following parameters:

1. the *out-of-sample terminal wealth* which we compute using $\upsilon^T = \sum_{i=0}^{N} h_{iT}$,
2. the *number of times* the out-of-sample terminal wealth of each method (including the market Index) exceeds the corresponding terminal wealths of the other methods (including the market Index), and
3. the *computational time* it takes us to evaluate each method.

Plots of cumulative wealth against time are also provided.

8.2 Experimental Data and Design

In this section, we first describe the data that we used in our simulations and specify the values of the parameters. Then, we provide a brief overview of the different stepsize rules that we used in the training iterations of the ADP methods and specify the ones that we selected to report in the numerical results. Finally, we discuss the issue of selecting initial slope values for the ADP methods and we conclude by reporting the software used for our experiments.

Data and Parameters

We use data from the FTSE100 Index to test the performance of the different portfolio selection methods. Guastaroba, Mansini and Speranza [39] have constructed four datasets of weekly rates of returns (these were computed using closing stock prices adjusted for dividents) of FTSE100 Index and have divided each dataset into

two parts: the in-sample data that have a horizon of 104 weeks and the out-of-sample data that have a horizon of 52 weeks.

The four datasets differ in the trend of the market behavior in the in-sample and out-of-sample data. The trend of the Index is increasing in both the in-sample and the out-of-sample data in the first dataset (label: "Up-Up"), increasing in the in-sample data but decreasing in the out-of-sample data in the second dataset (label: "Up-Down"), decreasing in the in-sample data but increasing in the out-of-sample data in the third dataset (label: "Down-Up") and decreasing in both the in-sample and the out-of-sample data in the last dataset (label: "Down-Down").

Table 8.1 summarizes the dates of the in-sample and out-of-sample data of each dataset.

Dataset	**in-sample data**	**out-of-sample data**
Up-Up	$17/04/1995 - 14/04/1997$	$14/04/1997 - 13/04/1998$
Up-Down	$05/01/1998 - 03/01/2000$	$03/01/2000 - 01/01/2001$
Down-Up	$12/03/2001 - 10/03/2003$	$10/03/2003 - 08/03/2004$
Down-Down	$04/01/2000 - 01/01/2002$	$01/01/2002 - 31/12/2002$

Table 8.1: Dates of in-sample and out-of-sample data

Figure 8.1 shows the price Index and the four market periods considered.

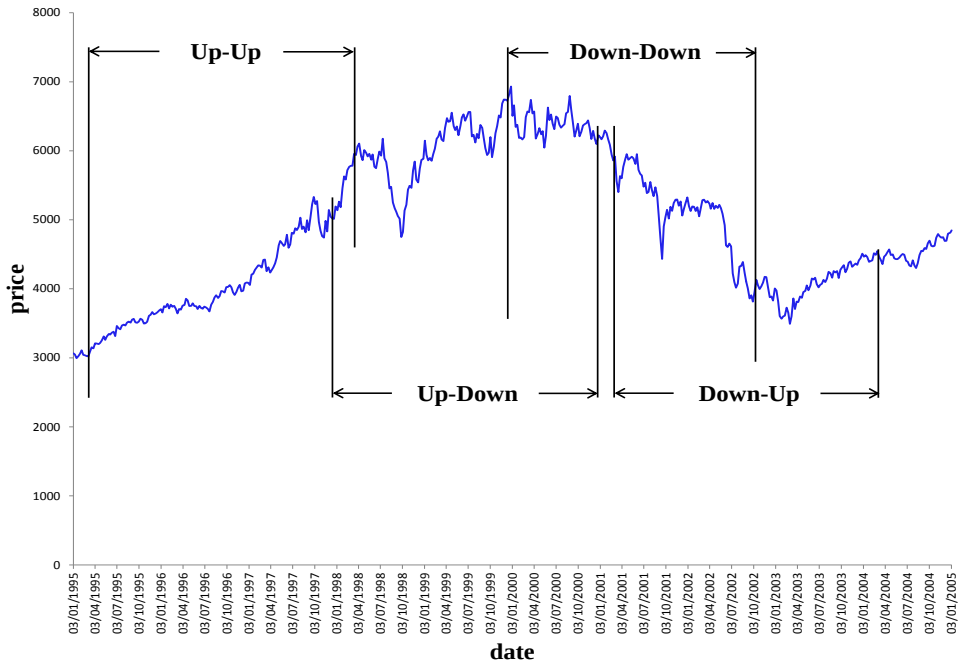


Figure 8.1: FTSE100 price Index from 03/01/1995 until 03/01/2005 and the four market periods

Regarding parameters, our first parameter was the weekly return of cash which in this study is assumed to be known. To compute the risk-free interest rate for each dataset we used the average annual base rates of the Bank of England during the out-of-sample horizon of each dataset. If r_{av} and r_{week} are respectively the average annual base rate for the 52 weeks out-of-sample and the corresponding weekly interest rate, then for each dataset we computed the weekly rate of return using $r_{\text{week}} = (1 + r_{\text{av}})^{\frac{1}{52}} - 1$. Table 8.2 summarizes the average annual base rates and the respective weekly interest rates for each dataset.

Dataset	**Average annual base rate** (r_{av})	**Weekly interest rate** (r_{week})
Up-Up	6.92%	0.129%
Up-Down	5.98%	0.112%
Down-Up	3.67%	0.069%
Down-Down	4%	0.075%

Table 8.2: Average annual base rates and weekly interest rates in the four datasets

Using the returns of the in-sample data of every dataset, we generated 4000 scenarios of returns, each scenario consisting of 52 weeks, using the OGARCH models (see section 2.4 and Appendix A). Note that these scenarios were generated for every dataset only *once*, the reason being that in the end we want to compare the different portfolio selection methods and thus we evaluate them on the same input information. The average computational time to generate an instance of 4000 scenario paths for every dataset was 226.66 seconds.

As explained above, in our experiments we used datasets from FTSE 100 Index which consists of the $N = 100$ companies in the London Stock Exchange with the highest market capitalization and for each of these datasets we evaluated all methods on out-of-sample horizons of $T = 52$ weeks each assuming an initial portfolio value of $h_{00} = £100000$ cash and zero holdings $h_{i0} = 0$ for $i = 1, 2, \ldots, 100$. Further, in our simulations we used a quantile parameter $\beta = 0.95$ and as in [39] we assumed proportional transaction costs equal to $\theta = 0.2\%$ for all assets. Using the generated scenarios of returns, we conducted several sets of simulations varying the level of the risk importance parameter γ from 0, where the investor is infinitely risk-averse, to 1, where the investor is infinitely risk-taking, with step 0.2.

Regarding the multistage stochastic programming method, in Appendix D we provide the scenario reduction parameters of the scenario trees that we used to evaluate this method. The scenario trees were generated in a backward fashion using the scenario tree construction methods of Appendix B. The amount by which the

original 4000 scenario paths were reduced in order to generate the scenario trees was selected in such a way so that each scenario tree preserves the maximum possible amount of the information contained in the original 4000 scenario paths and at the same time the corresponding deterministic equivalent remains computationally tractable. Note that to achieve this in many of our scenario trees we had to cut a significant amount of information. In the initial 53-stages scenario trees, for example, where each scenario path contained the forecasted returns of all 52 out-of-sample periods, for the sake of computational tractability we had to reduce the information contained in the 4000 scenario paths by 90%!

As for the ADP parameters, note that for the LADP-UB method we have considered two cases, one where we require that in each asset we cannot have more than one third of the total initial wealth of $\pounds 100000$ cash and one where we require that in each asset we cannot have more than one fifth of the total initial wealth of $\pounds 100000$. In line with the notation of chapter 6, the first case corresponds to $\alpha = 1/3$ and the second one corresponds to $\alpha = 1/5$. Further, note that for the PLADP methods we have considered two cases, one where the maximum number of slopes is $m = 3$ and one where the maximum number of slopes is $m = 5$.

Table 8.3 summarizes the values of the parameters used in our experiments.

Parameter	**Value**
N	100
T	52 weeks
h_{00}	£100000
h_{i0}	£0, i=1,2,..., 100
β	0.95
θ	0.2%
γ	0, 0.2, 0.4, 0.6, 0.8, 1
α	1/3, 1/5
m	3, 5

Table 8.3: Parameters and Values

Stepsize rules for ADP methods

In the literature (see [33]), there exist several stepsize rules that satisfy convergence conditions (6.33) and are grouped into the following two categories:

1. The *deterministic stepsize rules*, which depend on the iteration number as well as on adjustable parameters that control the rate of convergence. One

such stepsize rule is, for example, $\alpha_s = \frac{b}{(b-1)+s}$, where s is the iteration counter and b is a constant parameter to be specified. Figure 8.2 shows the different convergence rates for different values of parameter b. Note that the higher the value of parameter b the slower the rate of convergence.

2. The *stochastic stepsize rules*, where we take into account the prediction error which at iteration s is defined as follows: $\tilde{\epsilon}^s = \tilde{\Phi}^s - \hat{\Phi}^{s-1}$, where $\tilde{\Phi}^s$ are the observed estimates of the current iteration and the $\hat{\Phi}^{s-1}$ are mean estimates of the previous iteration. One such stepsize rule is, for example, *Kesten's rule* which is defined as follows:

$$\alpha^s = \alpha^0 \frac{\alpha}{\beta + K^s},\ s = 1, 2, \ldots, \tag{8.3}$$

In (8.3), α^0 and α are constant parameters to be specified and K^s is computed recursively by the following equation:

$$K^s = \begin{cases} s, & \text{if } s = 1, 2 \\ K^{s-1} + 1_{\{\tilde{\epsilon}^s \tilde{\epsilon}^{s-1} < 0\}}, & \text{if } s > 2 \end{cases} \tag{8.4}$$

where $1_{\{X\}}$ is the indicator function that gives one if condition X holds.

The idea in Kesten's rule is to decrease the stepsize and thus speed up convergence of the value function estimates if the successive prediction errors have opposite sign.

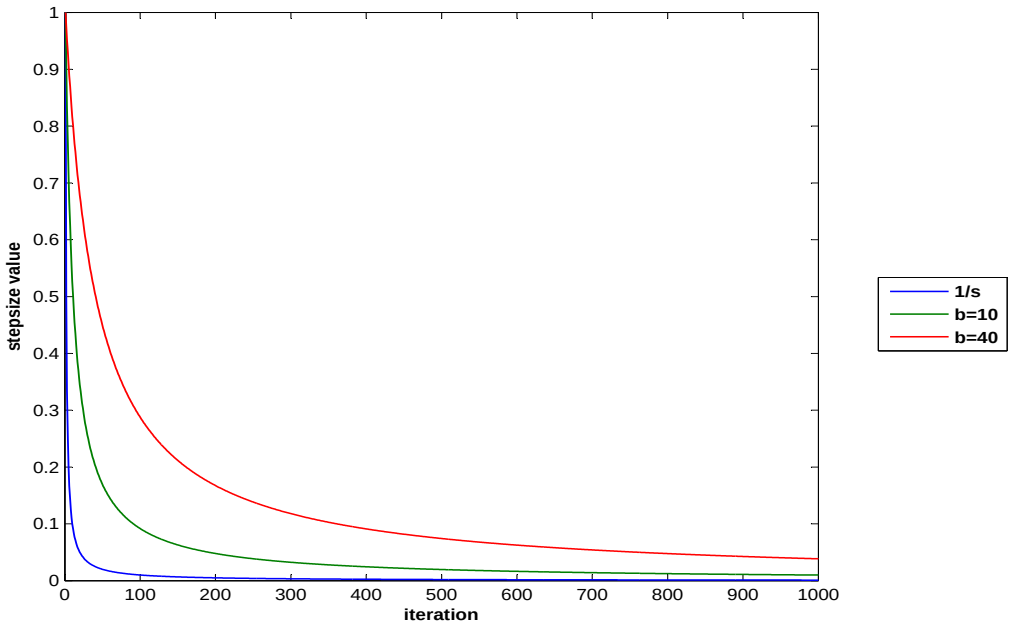


Figure 8.2: Different rates of convergence for stepsize rule $\alpha^s = \frac{b}{(b-1)+s}$

In this study, we would like to overcome the problem of deterministic stepsize rules where different dimensions have different convergence rates in the sense that the number of states explored differs from one dimension to another so we use a modified version of deterministic stepsize rule $\alpha_s = \frac{b}{(b-1)+s}$, where instead of an iteration counter s we use a counter n_t^s that measures the number of "states" visited so far at time t and iteration s (see for example [60]). Since we have continuous state variables, in order to count the number of "states" we use binary holdings vectors where 0 and 1 indicate respectively zero and positive holdings in an asset. We define these binary "states" as follows:

- In the linear approximation, we let our "state" be described by a $1 \times (N+1)$ vector of $0-1$ variables with one at position i if the holdings of asset i are positive. This gives us a total of 101 states for the simple LADP method since only one asset is selected per time period. In the LADP-UB method, the number of states that we can visit grows significantly since here, depending on how tight our upper bounds are, the selected portfolios comprise of more than one asset. For $\alpha = 1/3$, for example, we will see in the numerical results that in each time period we select between 2 and 5 assets which makes our "state" space much larger.

- In the piecewise linear approximation, we let our "state" be described by the following two vectors: a $1 \times (N+1)$ vector where the first $N+1$ positions are as in the linear approximation $0-1$ variables with one at position i if the holdings of asset i are positive and a $1 \times N$ vector of integers where position i denotes the active slope of asset i (i.e. the slope used for the current holdings). The main difference with the linear approximation is that here for each asset we can explore different slopes which increases significantly the size of our "state" space.

Therefore, we use the following stepsize rule:

$$\alpha_{it}^{s} = \frac{b}{(b-1)+n_t^s}, \;\; s = 1, 2, ..., \tag{8.5}$$

where b is a constant parameter to be specified. The advantage of using rule (8.5) is twofold: On the one hand, the slopes of the assets of every time period have the same rate of convergence since in each iteration these are updated with the same stepsize value. On the other hand, using the number of states visited instead of the iteration number in the denominator of (8.5) gives us the chance to explore more states before our slopes converge by assigning more weight to the observed slopes of the current iteration and less weight to the slope estimates of the previous iteration.

Note that before we resorted to stepsize rule (8.5) we had conducted several experiments trying a number of different deterministic and stochastic stepsize rules which achieved the desired convergence for the mean slope estimates but were rejected because they explored very few "states". In particular, from the deterministic stepsize rules we tried the *generalized harmonic stepsizes* as well as the *McClain's formula* and from the stochastic stepsize rules we tried *Kesten's rule* as well as *Kalman's filter*.

For our numerical results, we have selected to use stepsize $\frac{1}{n_t^s}$ for the simple LADP method and $\frac{25}{24+n_t^s}$ for the LADP-UB and the PLADP methods as these stepsizes provide a good compromise between visiting a sufficiently high number of "states" and achieving convergence for our slopes. In the end, we will also provide results for other values of parameter b.

Initial Slope Estimates for the ADP Methods

In approximate dynamic programming, ideally we would want to encourage exploration of actions/states before the value function estimates converge. In the Reinforcement Learning community, this is achieved by using a method known as *optimistic initial values* where we start with optimistic initial values for the value function estimates in the sense that these values are too high/low if we are maximizing/minimizing (see [77]).

The benefit of using optimistic initial values is that whatever the decisions taken initially, the achieved reward is always less than the initial value function estimates if we are maximizing. Due to this, the learner gets "disappointed" with the achieved reward and switches to other decisions trying to gain back his original value function estimates, thus visiting a fair amount of states before the value function estimates converge. The problem here is that if the initial value function estimates are too optimistic, i.e. too high if we are maximizing, then they might take a long time to come down to normal levels and this may affect convergence (not that they will not converge but they might converge to worse values).

For our simulations, we have constructed optimistic initial values for the slope estimates using the returns of the assets from the generated scenario paths. The catch here is that through the observed slopes (see Tables 6.3 and 6.4) the returns of the scenario paths are passed to previous time periods in a multiplicative manner. Thus, the slope of each asset at time t is a cumulative return that gives us the value of holding $\pounds 1$ of the asset from time t until the end of the time horizon. Considering this, we computed our initial slope estimates as follows: Starting at time $T = 52$, for each asset we computed its maximum return on the generated scenario paths. Then, we computed the average of these maximum returns and we set initial slopes $\hat{u}^0_{i(T-1)}$ equal to this average for all $i = 1, 2, \ldots, N + 1$. After computing the initial slope estimates of the last period, we stepped back to period 51 and, as in the last period, we computed the average of the maximum returns of the assets on the scenario paths at period 51. The new average return at period 51 was then multiplied with one of $\hat{u}^0_{i(T-1)}$ to give us slopes $\hat{u}^0_{i(T-2)}$ for all $i = 1, 2, \ldots, N + 1$. In a similar manner, we stepped back to previous time periods computing as we go optimistic estimates for the initial slopes of the assets in all time periods.

Note the following:

1. Each time we step back one time period to estimate the new optimistic initial slopes of the assets, we do an averaging of the maximum returns of the assets

instead of using again a max operator. The reason for this is that in the latter case the slopes of the assets in the early periods of the out-of-sample horizon explode, thus resulting in too optimistic initial slope estimates.

2. All assets including cash are assigned the same initial slopes at every time period which results in us moving forward through time in the first iteration of each ADP algorithm only with cash (this is because the buying slopes of the assets become all negative in all time periods). The reason for doing this is that, considering the importance of the initial slope values in obtaining "good" value function estimates in the end, we want to avoid any initial bias coming from a "pre-defined" preference over some assets. Instead, we let the ADP algorithms decide which assets are "good" based on the returns of the assets in the scenario paths.

Software

All experiments were conducted on PCs with Intel(R) Core(TM) i7-2600S processors and 8GB RAM memory and all algorithms were implemented in Version 7.12 (R2011a) MATLAB. To solve the multistage stochastic programs, we used the General Algebraic Modeling System (GAMS), where we used the embedded solver Scenred2 to reduce the number of scenarios and construct scenario trees as well as the embedded optimizer IBM CPLEX 12 to solve the resulting deterministic equivalent problems. All other optimization problems were solved in MATLAB with IBM CPLEX 12.

8.3 Numerical Results

In this section, we present the numerical results from our simulations. In particular, we first report in subsection 8.3.1 the characteristics of the selected portfolios and provide plots that show how slopes converge in the ADP methods. Then, in subsection 8.3.2, we proceed with the performance evaluation where we report the performance measures and the computational times, and we provide performance tracking plots for all methods. Finally, in subsections 8.3.3, 8.3.4 and 8.3.5 we perform robustness checks by examining respectively the expected terminal wealth and the CVaR of the different portfolio policies, the impact of the length of the planning horizon and the impact of transaction costs.

8.3.1 Characteristics of Selected Portfolios and Convergence of Slopes

In this section, we report the characteristics of the selected portfolios described in section 8.1.1 above. In particular, for each method under the column labeled "NA_t" we report a range that gives the minimum and the maximum number of assets selected per out-of-sample period, under the column labeled "NA" we report the number of different assets used throughout the out-of-sample horizon, under the column labeled "Cash" we report $0-1$ if cash is used in some of the selected portfolios, 1 if we only use cash and 0 if we never use cash, and finally under the column labeled "HI*" we report a range that gives the minimum and the maximum normalized Herfindahl Index per out-of-sample period. Further, we provide plots that show how slopes converge in the ADP methods.

Characteristics of Selected Portfolios

Table 8.4, summarizes the characteristics of the selected portfolios for the equally-weighted, the single-period and the multistage stochastic programming methods.

In the equally-weighted method, the buy-and-hold and and the fixed-mix strategies are labeled respectively as "EW-BH" and "EW-FM" and in both strategies all assets including cash are used in all time periods out-of-sample. As explained above, the difference between the two strategies is that unlike the buy-and-hold strategy, where the portfolio is rebalanced to fixed equal target proportions only in the first time period, in the fixed-mix strategy the portfolio is rebalanced to equal proportions in all assets in all time periods.

In the single-period method labeled as "SP", the number of assets in the selected portfolios as well as the degree of diversification depend on the market expectations and the risk importance parameter γ. Thus, when we expect the market to go up out-of-sample (this is in the Up-Up and Up-Down datasets) we neglect cash and go for risky assets hoping to earn something better than the risk-free interest rate. On the contrary, when we expect the market to go down out-of-sample (this is in the Down-Up and Down-Down datasets) either we prefer to go for the safer risk-free interest rate or we select a few risky assets. Regarding the risk importance parameter, note that the higher the value of γ (this corresponds to more risk) the less the number of assets and the less the degree of diversification in the selected portfolios. In particular, for $\gamma = 1$ we are infinitely risk taking and a single risky

asset is selected everywhere.

In the multistage stochastic programming method labeled as "MSP", we observe a similar behavior as in the single-period method. The main difference here is that we have less diversified portfolios per time period. Overall, however, note that the number of different assets used out-of-sample is considerably higher in this method as compared to the single-period one.

Table 8.5 summarizes the characteristics of the selected portfolios for the LADP methods.

In the simple linear approximate dynamic programming method without upper bounds on the holdings of the risky assets labeled as "LADP", the risk importance parameter γ has no effect in the composition of the selected portfolios which comprise of only one asset per time period. As explained in chapter 6, this is due to the assumed linear approximations of the value functions which ignore the effect of CVaR and this leads to a similar behavior to the MSP method for $\gamma = 1$. In particular, in the simulation results we noticed that for $\gamma = 1$ the same asset was selected by both methods in most out-of-sample periods.

In the linear approximate dynamic programming method with upper bounds on the holdings of the risky assets labeled as "LADP-UB", the effect of CVaR is still ignored since the functional type of the approximate value functions remains linear. What changes from the simple LADP method without upper bounds is diversification. Recall from chapter 6 that the reason why we considered adding these upper bound constraints is that for low values of the state variables this becomes a "good" approximation (see Figure 6.1). Specifically, here the additional upper bound constraints on the holdings of the risky assets force us to include at least 2 assets in the selected portfolios. Any variation in the number of assets between different time periods is because the actual returns change the amount of the total wealth that can be allocated to assets in every time period. In the simulation results, we noticed that, as expected, when $\alpha = 1/5$ we have slightly more diversified portfolios as compared to when $\alpha = 1/3$ since $\alpha = 1/5$ gives us tighter upper bounds regarding how much we can have in each asset. Note that after adding the new constraints the number of different assets used out-of-sample is higher.

Table 8.6 summarizes the characteristics of the selected portfolios for the PLADP methods.

In the piecewise linear approximate dynamic programming methods labeled as "PLADP", the use of piecewise linear concave value functions results in well diversified portfolios since we estimate concave functions for the assets assuming

that the value of an extra unit of an asset deteriorates as the holdings of the asset grow and takes into account the effect of CVaR (this is communicated to previous time periods through the observed slopes). Note that for $m = 5$ slopes we have, as expected, more diversified portfolios as compared to $m = 3$ slopes since with more slopes we have more jumps between the slopes and as a result between the assets. Further, note that in most cases for $m = 5$ the number of different assets used throughout the out-of-sample horizon is higher than for $m = 3$ but in both approximations we use overall fewer assets than in multistage stochastic programming. This could be improved by adding more slopes in the approximate piecewise linear value functions.

Dataset	γ	EW-BH				EW-FM				SP				MSP			
		NA_t	NA	Cash	$HI^\star$	NA_t	NA	Cash	$HI^\star$	NA_t	NA	Cash	$HI^\star$	NA_t	NA	Cash	$HI^\star$
Up-Up	0	101	101	1	0	101	101	1	0	25-31	34	0	0.12-0.15	1-18	50	0	0.18-1
	0.2									22-27	30	0	0.14-0.18	1-15	39	0	0.24-1
	0.4									18-20	21	0	0.16-0.22	1-14	28	0	0.18-1
	0.6									12-13	13	0	0.27-0.35	1-7	17	0	0.30-1
	0.8									2	2	0	0.50-0.69	1-2	5	0	0.57-1
	1									1	1	0	1	1	3	0	1
Up-Down	0	101	101	1	0	101	101	1	0	33-37	44	0	0.06-0.13	1-32	68	0	0.11-1
	0.2									28-34	39	0	0.07-0.15	1-28	59	0	0.15-1
	0.4									27-31	34	0	0.09-0.19	1-24	51	0	0.19-1
	0.6									14-20	23	0	0.11-0.26	1-13	31	0	0.21-1
	0.8									7-9	9	0	0.20-0.34	1-7	20	0	0.32-1
	1									1	1	0	1	1	7	0	1
Down-Up	0	101	101	1	0	101	101	1	0	1	1	1	1	1-14	59	0-1	0.20-1
	0.2									1	1	1	1	1-12	49	0-1	0.20-1
	0.4									1	1	1	1	1-27	45	0-1	0.111-1
	0.6									1-27	30	0-1	0.10-1	1-16	26	0	0.20-1
	0.8									9-10	10	0	0.31-0.35	1-8	14	0	0.39-1
	1									1	1	0	1	1	6	0	1
Down-Down	0	101	101	1	0	101	101	1	0	1	1	1	1	1-28	70	0-1	0.11-1
	0.2									1	1	1	1	1-37	69	0-1	0.10-1
	0.4									1	1	1	1	1-36	60	0-1	0.05-1
	0.6									32-36	36	0	0.08-0.09	1-22	38	0	0.09-1
	0.8									16-18	20	0	0.13-0.15	1-14	23	0	0.13-1
	1									1	1	0	1	1	9	0	1

Table 8.4: Characteristics of selected portfolios for the equally-weighted, the single-period and the multistage stochastic programming methods

Dataset	γ	LADP				LADP-UB ($\alpha = 1/3$)				LADP-UB ($\alpha = 1/5$)			
		NA$_t$	**NA**	**Cash**	**HI***	**NA**$_t$	**NA**	**Cash**	**HI***	**NA**$_t$	**NA**	**Cash**	**HI***
Up-Up	0	1	3	0	1	2-5	12	0-1	0.25-0.55	2-9	12	0-1	0.14-0.68
	0.2						11			2-8	12		
	0.4						11			2-9	13		
	0.6						11			2-8	13		
	0.8						12			2-8	13		
	1						12			2-9	13		
Up-Down	0	1	5	0	1	2-4	13	0-1	0.25-0.60	2-5	14	0-1	0.19-0.70
	0.2						12				13		
	0.4						12				12		
	0.6						12				12		
	0.8						12				13		
	1						12				13		
Down-Up	0	1	5	0	1	2-4	20	0-1	0.25-0.59	2-5	19	0-1	0.19-71
	0.2												
	0.4												
	0.6												
	0.8												
	1												
Down-Down	0	1	7	0	1	2-4	13	0-1	0.30-0.56	2-6	12	0-1	0.18-0.69
	0.2						12	0-1					
	0.4						12	0-1					
	0.6						12	0-1					
	0.8						12	0-1					
	1						12	0-1					

Table 8.5: Characteristics of selected portfolios for the LADP methods

Dataset	γ	PLADP ($m=3$)				PLADP ($m=5$)			
		$\mathbf{NA}_t$	NA	Cash	HI*	$\mathbf{NA}_t$	NA	Cash	HI*
Up-Up	0	2-29	37	0	0.21-0.83	3-26	31	0	0.16-0.84
	0.2	3-33	33		0.18-0.83	3-22	31		0.18-0.86
	0.4	4-28	35		0.23-0.83	3-23	31		0.20-0.83
	0.6	3-25	38		0.18-0.84	4-29	34		0.21-0.83
	0.8	4-32	36		0.22-0.83	4-30	36		0.19-0.86
	1	3-30	31		0.22-0.83	4-34	40		0.17-0.86
Up-Down	0	2-21	31	0	0.20-0.82	1-33	40	0	0.21-1
	0.2	3-18	34		0.11-0.80	3-34	41		0.13-0.80
	0.4	3-23	35		0.15-0.81	3-34	40		0.19-0.79
	0.6	3-20	31		0.18-0.84	3-25	36		0.14-0.79
	0.8	2-21	33		0.14-0.93	2-27	33		0.13-0.78
	1	2-21	34		0.12-0.91	3-31	40		0.10-0.76
Down-Up	0	1-43	50	0	0.13-1	2-35	48	0	0.12-0.97
	0.2	2-19	33	0-1	0.20-0.97	2-27	45		0.12-0.95
	0.4	2-22	42	0-1	0.16-0.93	2-33	50		0.09-0.92
	0.6	2-21	39	0	0.21-0.88	3-30	45		0.14-0.85
	0.8	2-27	43	0-1	0.12-0.85	3-30	45		0.12-0.78
	1	2-25	40	0	0.16-0.92	2-29	43		0.09-0.80
Down-Down	0	4-20	37	0	0.15-0.63	3-34	47	0	0.16-0.65
	0.2	4-16	33		0.19-0.64	3-34	51		0.15-0.66
	0.4	3-16	35		0.20-0.66	3-42	52		0.13-0.66
	0.6	5-23	38		0.16-0.63	3-34	47		0.14-0.66
	0.8	5-20	36		0.14-0.64	4-39	52		0.14-0.63
	1	5-17	31		0.18-0.64	3-31	47		0.17-0.66

Table 8.6: Characteristics of selected portfolios for the PLADP methods

Convergence of Slopes

For the selected stepsizes (see section 8.2 above), Figures 8.3-8.5 show how slopes converge in the ADP methods for asset $i = 3$, time $t = 30$ and risk importance parameter $\gamma = 0.8$ in the Up-Up dataset, where the decrease in the mean slope values is due to visiting new "states" which increases parameter n_t^s in the stepsize formula $\frac{b}{(b-1)+n_t^s}$ and any noise in the slope values is due to the noise in the data.

In the simple LADP method (see Figure 8.3), we notice that the mean slope initially decreases slowly but converges very quickly after approximately 500 iterations. The reason for this is that, as explained earlier, with this method we can explore only 101 states, one for visiting each asset.

In the LADP-UB method (see Figure 8.4), we observe a slightly different pattern. Here, the mean slope initially decreases fast but it requires approximately 2000 iterations before it converges. The reason for this is that, as explained earlier, with this method due to diversification we can explore many more states.

In the PLADP method (see Figure 8.5 for an example with $m = 3$ slopes), the

mean slopes behave as in the LADP-UB methods but overall require many more iterations before they converge (for the given example we need approximately 3000 iterations). The reason for this is that with the PLADP method we do not explore only the assets but also the slopes of the assets which makes the state space even larger. Further, note that in the PLADP methods the earlier slopes of the assets converge faster as they are the first to explore when visiting the assets. In Figure 8.5, for example, we notice that slope 1 converges to its mean value only after approximately 100 iterations.

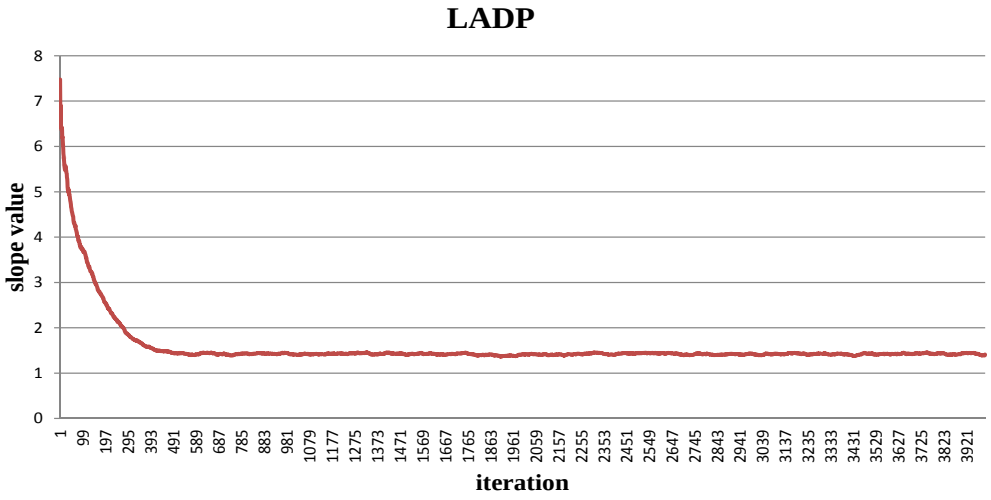


Figure 8.3: Slope versus iteration for asset $i = 3$ at time $t = 30$ and for $\gamma = 0.8$ in the Up-Up dataset in the LADP method

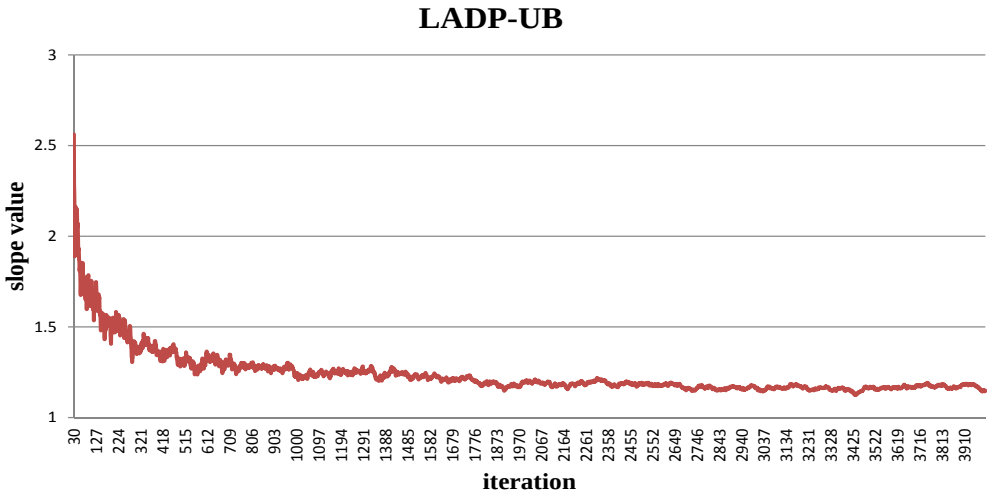


Figure 8.4: Slope versus iteration for asset $i = 3$ at time $t = 30$ and for $\gamma = 0.8$ in the Up-Up dataset in the LADP-UB method

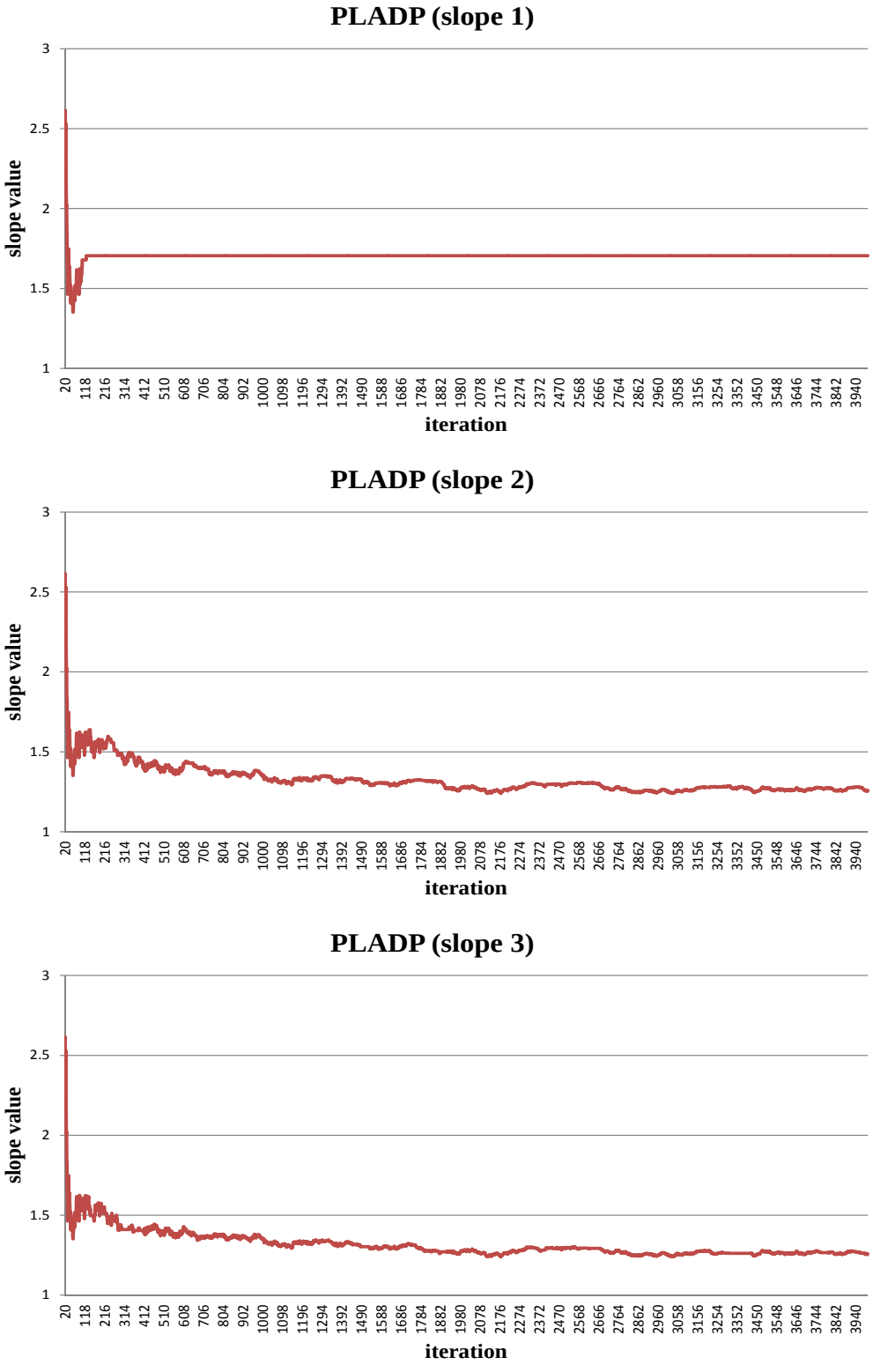


Figure 8.5: Slopes versus iteration for asset $i = 3$ at time $t = 30$ and for $\gamma = 0.8$ in the Up-Up dataset in the PLADP method with $m = 3$ slopes

8.3.2 Performance Evaluation

As explained earlier, the out-of-sample terminal wealth is our performance measure. In this section, we compare the portfolio selection methods with respect to their out-of-sample terminal wealths. Specifically, we first report the performance measures for each method and then based on them we compare all methods. Then, for each method and for each value of the risk importance parameter γ we compute the respective average out-of-sample terminal wealth across the four datasets (Up-Up, Up-Down, Down-Up, Down-Down) and use it as a measure of the average performance of each method. Particularly for the ADP methods, we provide additional results for their average performance for more stepsizes. Finally, we conclude with computational times.

Performance of Portfolio Selection methods

Here, we report the out-of-sample terminal wealths for each method, where note that as explained earlier in section 8.2 for the simple LADP method we use stepsize $\frac{1}{n_t^s}$ and for the other ADP methods we use stepsize $\frac{25}{24+n_t^s}$, and comment on them using as a reference the market. Table (8.7) summarizes the out-of-sample terminal wealths for all methods, all datasets and the different values of the risk importance parameter γ. Note that in total for each method we have 24 instances (4 datasets $\times$ 6 levels of γ).

The equally-weighted buy-and-hold and fixed-mix strategies labeled respectively as "EW-BH" and "EW-FM" consistently outperform the market in all datasets except for the Up-Up dataset, where, although we are below the market, our initial portfolio value grows in both methods by approximately 30%. Note that, although the discrepancies in the terminal wealth values in the two strategies are negligible, the equally-weighted buy-and-hold strategy is doing slightly better than the fixed-mix one in all datasets except for the Up-Down dataset, where the buy-and-hold strategy is slightly below the fixed-mix one.

The single-period portfolio method labeled as "SP" performs very poorly as in most instances its out-of-sample terminal wealth is significantly less than the market except for $\gamma = 0.6$ and 0.8 in the Down-Up dataset and everywhere in the Down-Down dataset where it is doing better.

The multistage stochastic programming method labeled as "MSP" is doing consistently worse than the market except for $\gamma \geq 0.4$ in the Down-Down dataset where it beats the market. Further, note that in most instances of this method our terminal

portfolio value falls bellow $\pounds 100000$.

The simple linear approximate dynamic programming method labeled as "LADP" ignores risk and as a result we have only one terminal wealth value for each dataset. Although in this method we are doing better than the market in two out of the four datasets, the Down-Up and Down-Down ones, by selecting only one asset per time period we risk experiencing dramatic losses (see for example the Up-Down dataset where our initial portfolio value drops by approximately 64%).

The linear approximate dynamic programming method with upper bound constraints on the holdings of the risky assets labeled as "LADP-UB" gives us in each dataset very similar terminal wealth values, in some instances identical, no matter what the value of the risk importance parameter γ is. The reason for this is that in this approximation we still have one slope for each slope which results in ignoring CVaR. Here, although we do not see a clear pattern as to whether the higher degree of diversification which is caused by the tighter upper bound constraints on the holdings of the risky assets for $\alpha = 1/5$ improves the performance, dirversification seems to protect us from experiencing high losses. Thus, when the market goes down out-of-sample, i.e. in the Up-Down and Down-Down datasets, we are clearly better off the market and above our initial portfolio value of $\pounds 100000$. When the market goes up out-of-sample, i.e. in the Up-Up and Down-Up datasets, although we are below the market, our initial portfolio grows above $\pounds 100000$ in value in all instances.

Unlike other methods that perform badly, the piecewise linear approximate dynamic programming method labeled as "PLADP" consistently outperforms the market in all datasets for both $m = 3$ slopes and $m = 5$ slopes except for the Up-Up dataset where, although our initial portfolio value grows significantly in most instances (note that especially for low values of γ including $\gamma = 0$ we are doing pretty good), the Index does better than every other method. Specifically for $m = 5$ slopes the performance measures look in general way better than for $m = 3$ slopes. Further, note that this method is doing well in datasets that other methods perform badly.

In Appendix E, we provide cumulative wealth plots that show how cumulative wealth changes out-of-sample with time for the different values of the risk importance parameter γ in all four datasets.

To compare now the different portfolio selection methods and the market Index, we compute in Table 8.8 in how many instances out of the 24 each method and the market Index outperform the other methods and the market Index with respect to

our performance measure.

Clearly the best performing methods are the PLADP method with $m = 5$ slopes, the PLADP method with $m = 3$ slopes and the equally-weighted methods since they outperform the other methods and the market in at least half of the instances. Among the four, the best performing is the PLADP method with $m = 5$ slopes which beats the PLADP method with $m = 3$ slopes and the equally-weighted methods in 16 out of the 24 instances, followed by the the PLADP method with $m = 3$ slopes which beats the buy-and-hold and the fixed-mix equally-weighted methods in 15 and 14 out of the 24 instances respectively.

Dataset	γ	FTSE	EW-BH	EW-FM	SP	MSP	LADP	LADP-UB		PLADP	
								$\alpha = 1/3$	$\alpha = 1/5$	$m = 3$	$m = 5$
Up-Up	0	143601.38	130011.93	128821.16	129528.15	71682.26	84312.51	112354.58	123466.09	125581.73	115829.62
	0.2				128867.97	94066.95		111929.10	123448.37	127064.39	123768.05
	0.4				126680.20	96615.72		111929.10	124440.07	110346.22	111473.22
	0.6				122376.75	91156.03		112795.76	124007.86	108087.36	120966.91
	0.8				103421.54	88651.21		112795.76	124007.86	119311.81	112486.75
	1				69912.77	82330.09		112795.76	124043.59	102292.24	113765.82
Up-Down	0	89788.17	105262.25	106652.31	58575.51	77641.30	35987.38	122124.22	108874.86	113208.79	127971.69
	0.2				58018.93	63322.20		114553.04	108247.48	105383.91	117407.22
	0.4				55443.25	75137.02		114764.44	108452.87	95402.15	106839.34
	0.6				47459.70	58158.09		115866.63	107699.59	90676.71	100336.32
	0.8				37189.91	46346.34		114824.56	104975.10	104760.61	114850.97
	1				40523.23	36693.78		114824.56	107689.99	108714.75	95624.64
Down-Up	0	132532.01	156160.11	155357.52	103673.08	117515.10	145029.84	120324.12	110525.84	181669.09	205900.70
	0.2				103673.08	82482.68		121309.89	110525.84	217338.56	222927.47
	0.4				103673.08	72953.63		121804.76	111007.55	181707.49	206173.46
	0.6				136968.21	80532.89		121644.94	111007.55	200196.25	204696.62
	0.8				156679.99	96396.33		121644.94	111007.55	179878.15	228856.90
	1				99805.38	110514.50		119186.49	111007.55	201382.49	196325.57
Down-Down	0	75524.21	84529.28	84227.82	104000.00	62579.01	125380.59	91222.07	101372.24	97336.07	115882.48
	0.2				104000.00	70376.29		92818.73	101795.00	88808.88	106934.62
	0.4				104000.00	83750.42		92581.79	101795.00	112862.12	111190.07
	0.6				90722.25	103974.97		92818.73	101648.98	114604.32	111470.92
	0.8				94180.21	110283.60		92051.04	101207.85	101348.79	112200.37
	1				114646.76	107590.91		92282.88	101207.85	107883.07	107448.80

Table 8.7: Out-of-sample terminal wealths

# of instances = 24	FTSE	EW-BH	EW-FM	SP	MSP	LADP	LADP-UB		PLADP	
							$\alpha = 1/3$	$\alpha = 1/5$	$m = 3$	$m = 5$
FTSE	-	6	6	16	20	12	12	12	6	6
EW-BH	18	-	18	17	21	18	12	13	9	8
EW-FM	18	6	-	15	21	18	12	13	10	8
SP	8	7	9	-	14	12	11	9	7	5
MSP	4	3	3	10	-	10	3	4	1	1
LADP	12	6	6	12	14	-	12	12	6	6
LADP-UB ($\alpha = 1/3$)	12	12	12	13	21	12	-	12	10	5
LADP-UB ($\alpha = 1/5$)	12	11	11	15	20	12	12	-	10	8
PLADP ($m = 3$)	18	15	14	17	23	18	14	14	-	8
PLADP ($m = 5$)	18	16	16	19	23	18	19	16	16	-

Table 8.8: In how many instances out of the 24 a row method outperforms a column method

Average Performance

Assuming an investor who enters the market with $£100000$ cash at the beginning of each out-of-sample horizon with equal probability 0.25, Tables 8.9 and 8.10 summarize the average out-of-sample terminal wealth values for each method across the four datasets and for each value of the risk importance parameter γ.

Looking at the average performance measures of Tables 8.9 and 8.10 and starting from the best performing method, our simulation results suggest the following ordering for the methods in terms of their average performance:

1. PLADP with $m = 5$ slopes
2. PLADP with $m = 3$ slopes
3. Equally-weighted strategies
4. LADP-UB with $\alpha = 1/3, 1/5$
5. Single-period method
6. LADP
7. Multistage stochastic programming

Note that for the LADP-UB methods we do not have a clear picture as to which of the two $\alpha = 1/3$ or $\alpha = 1/5$ performs better since their average performance values are very similar. Further, note that in the PLADP, the single-period and the multistage stochastic programming methods performance seems to improve significantly either when we are infinitely risk averse for $\gamma = 0$ or when we accept a reasonable amount of risk for $\gamma = 0.8$.

γ	FTSE	EW-BH	EW-FM	SP	MSP
0	110361.44	118990.89	118764.70	98944.19	82354.42
0.2				98640.00	77562.03
0.4				97749.13	82114.20
0.6				99381.73	83455.50
0.8				97867.91	85419.37
1				81222.03	84282.32

Table 8.9: Average out-of-sample terminal wealth for the market Index, the equally-weighted, the single-period and the multistage stochastic programming methods

γ	LADP	LADP-UB		PLADP	
		$\alpha = 1/3$	$\alpha = 1/5$	$m = 3$	$m = 5$
0	97677.58	111506.25	111059.76	129448.92	141396.12
0.2		110152.69	111004.17	134648.93	142759.34
0.4		110270.02	111423.87	125079.49	133919.02
0.6		110781.51	111090.99	128391.16	134367.69
0.8		110329.08	110299.59	126324.84	142098.75
1		109772.42	110987.24	130068.14	128291.21

Table 8.10: Average out-of-sample terminal wealth for the ADP methods

Figures 8.6-8.11 show how the average cumulative wealth changes with time for each method and for each level of risk and one can clearly see the above distinction between the methods with respect to their average performance. Note that the four best performers with respect to average performance, i.e. the two PLADP methods and the two equally-weighted strategies, tend to track the market Index.

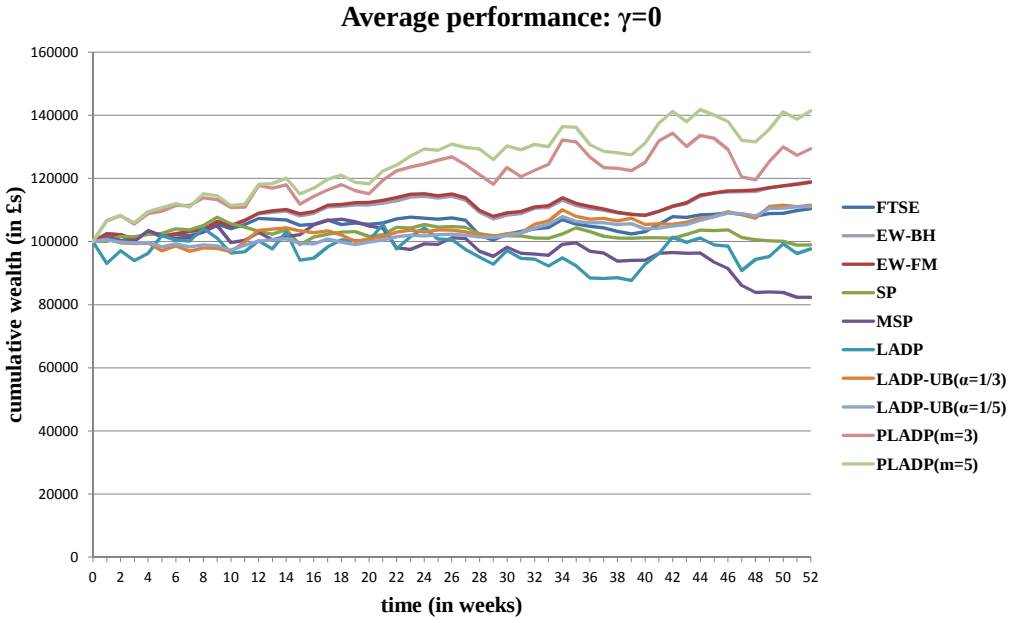


Figure 8.6: Average out-of-sample cumulative wealth against time for $\gamma = 0$

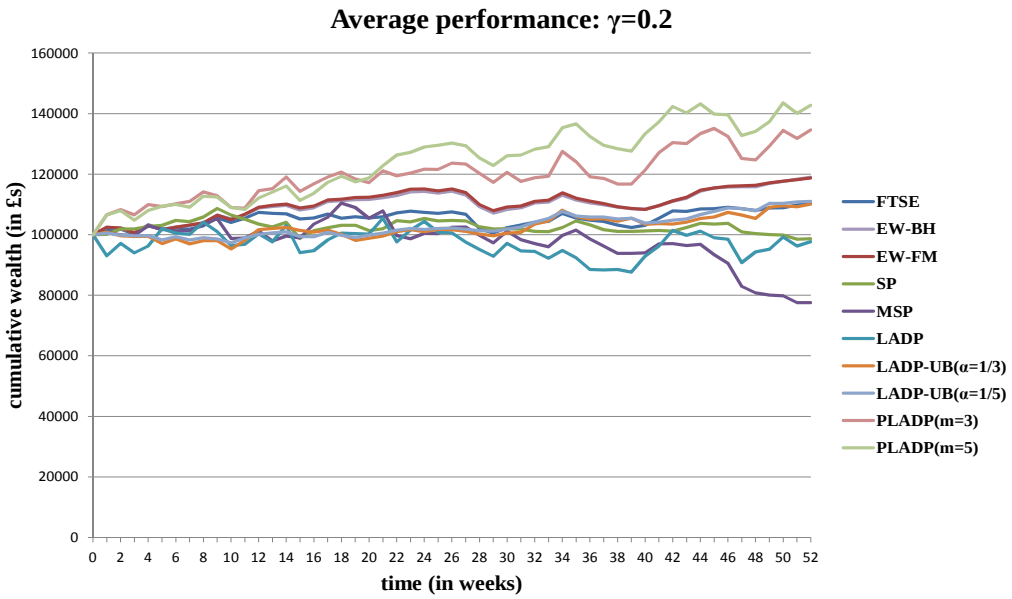


Figure 8.7: Average out-of-sample cumulative wealth against time for $\gamma = 0.2$

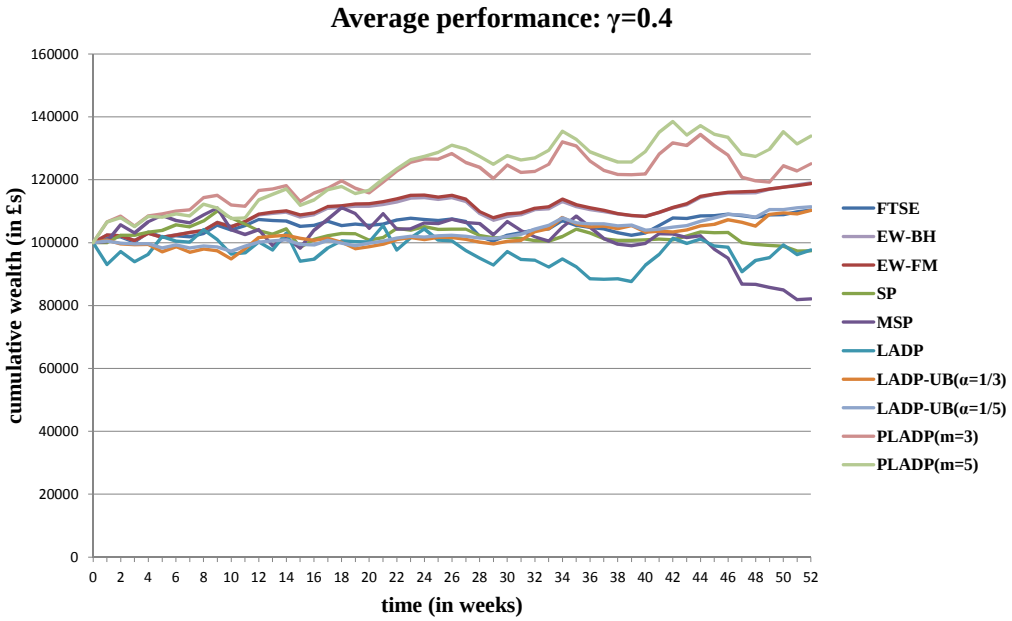


Figure 8.8: Average out-of-sample cumulative wealth against time for $\gamma = 0.4$

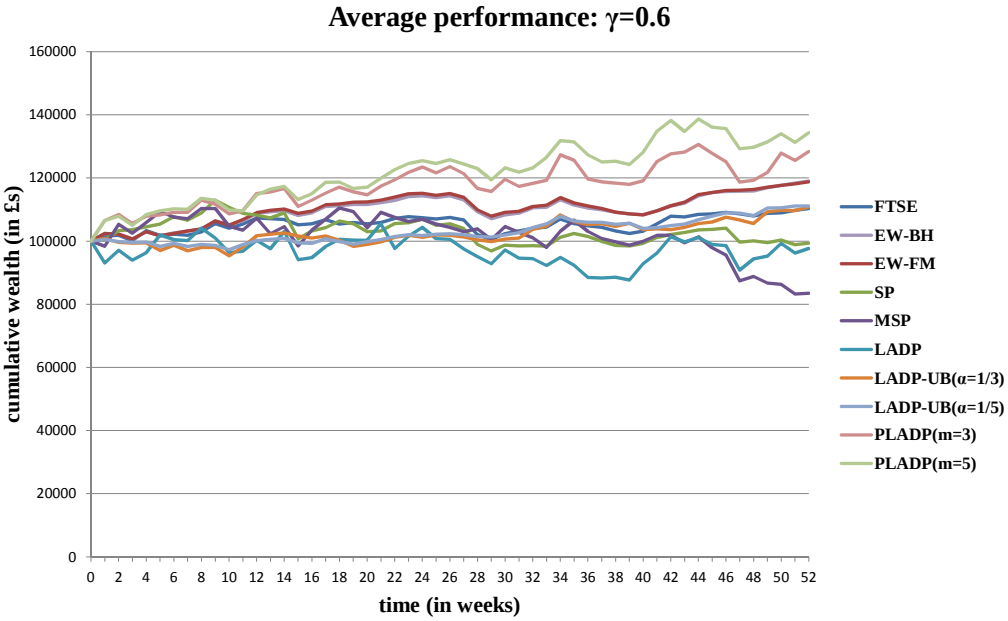


Figure 8.9: Average out-of-sample cumulative wealth against time for $\gamma = 0.6$

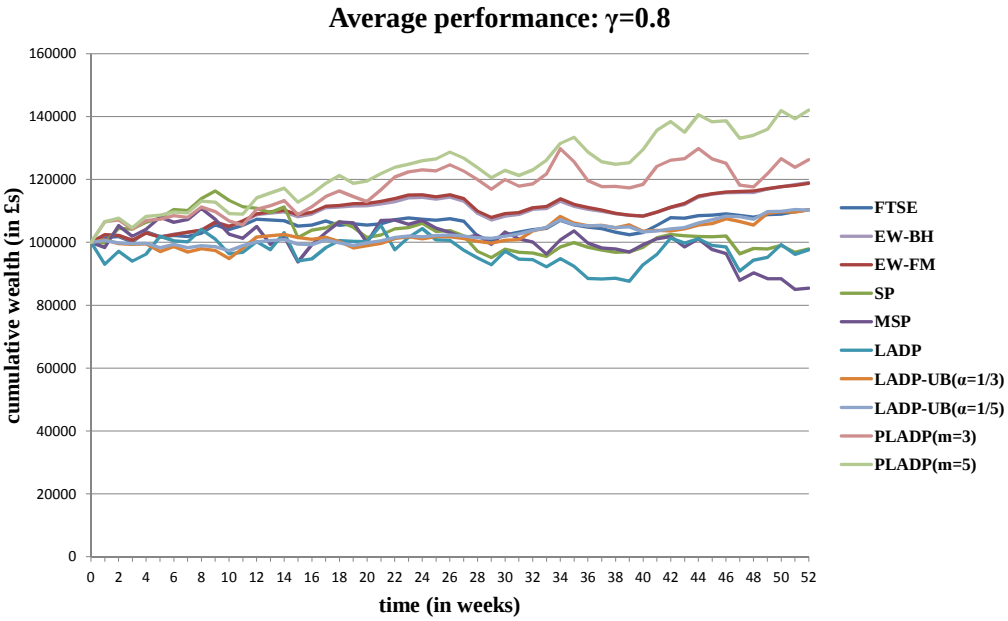


Figure 8.10: Average out-of-sample cumulative wealth against time for $\gamma = 0.8$

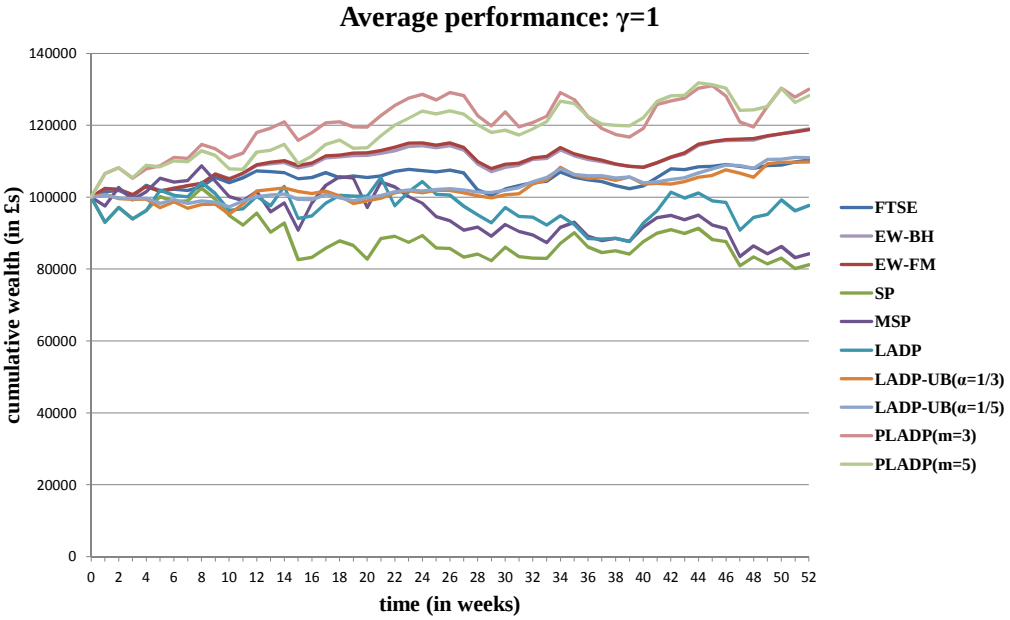


Figure 8.11: Average out-of-sample cumulative wealth against time for $\gamma = 1$

Average Performance of ADP Methods for Other Stepsizes

In our experiments, we used stepsize rule $\frac{b}{(b-1)+n_t^s}$ and the numerical results given above are for $b = 1$ for the simple LADP method and $b = 25$ for all other ADP methods. The reason for selecting these stepsizes, as explained earlier in section 8.2, is that they provide a good compromise between exploring a sufficiently high number of "states" and achieving convergence of the slopes in the value function approximations. Here, we provide average performance numerical results to examine the impact of stepsize rule $\frac{b}{(b-1)+n_t^s}$ on our results for other values of parameter b. In particular, we examine how ADP methods perform on average for $b = 1, 5, 10, 15, 20, 25, 30$ and 35. For any other $b > 35$, we have observed that the number of the new "states" explored is insignificant thus making the price we have to pay for convergence unnecessarily high. In the discussion that follows, we use the buy-and-hold equally-weighted strategy as a reference for our comparisons with the ADP methods since in the numerical results provided above, disregarding the ADP methods, the buy-and-hold equally-weighted strategy is the best performing among all other methods.

Table 8.11 summarizes the average terminal wealth values across the four datasets for the different values of the risk importance parameter γ. To make it easier for

the reader to follow up the numerical results provided in Table 8.11, in Figures 8.12-8.17 we provide bar charts, where the height of each bar gives us the average out-of-sample terminal wealth value for each method and each value of parameter b.

Observing Figures 8.12-8.17, we notice that for the above values of parameter b the results are very similar to the ones given above for the selected values $b = 1$ for the LADP method and $b = 25$ for the other ADP methods. In particular, we observe the following: Overall, the best performing method is PLADP, followed by the equally-weighted buy-and-hold strategy, then the LADP-UB method and finally the simple LADP method. In particular for the PLADP methods, for $\gamma = 0$ and 0.2 PLADP with $m = 3$ slopes performs better than PLADP with $m = 5$ slopes for 5 out of the 8 values of parameter b. On the contrary, for $\gamma = 0.4, 0.6$ and 0.8 PLADP with $m = 5$ slopes performs better than PLADP with $m = 3$ slopes respectively for 6, 5 and 6 out of the 8 values of parameter b. For $\gamma = 1$, the two methods seem to be equally good since they outperform each other for half of the values of parameter b. Further, both PLADP methods consistently beat the buy-and-hold equally-weighted strategy for all values of parameter b except for $b = 1$, where the buy-and-hold equally-weighted strategy has a slightly better performance, and $b = 5$ where all three methods seem to do equally good. For $b > 5$, the average performance of the two PLADP methods improves significantly and is far better than that of the fixed-mix equally-weighted method. The reason for this is that for very low values of parameter b our slope estimates converge faster. As a result of this, we visit less "states" and our slope estimates converge to "worse" values.

γ	b	EW-BH	LADP	LADP-UB		PLADP	
				$\alpha = 1/3$	$\alpha = 1/5$	$m = 3$	$m = 5$
0	1	118990.89	97677.58	124329.27	123452.92	115676.82	115060.35
	5		85161.48	129878.92	117490.06	128090.62	119149.41
	10		83700.95	124895.97	117983.96	147781.35	138615.84
	15		87366.16	115201.09	111912.06	138601.14	139220.05
	20		89907.79	112657.08	111837.09	150468.13	123450.00
	25		90020.17	111506.25	111059.76	129448.92	141396.12
	30		91492.12	113536.83	104583.86	151870.38	125036.98
	35		93918.46	112044.79	110433.66	117644.99	127290.65
0.2	1	118990.89	97677.58	125656.57	123432.49	117056.24	114625.46
	5		82046.22	129535.82	117588.84	136101.68	121579.68
	10		83419.76	124764.33	117598.54	139666.37	131253.66
	15		88987.50	116202.59	111816.45	132072.57	125216.45
	20		91144.26	112970.61	111977.47	130630.59	131331.30
	25		90020.17	110152.69	111004.17	134648.93	142759.34
	30		91305.12	114835.04	105093.90	131100.20	137638.19
	35		91979.05	113057.56	110546.57	123300.46	122042.57
0.4	1	118990.89	97677.58	125395.25	122840.94	116150.1	117617.03
	5		82120.23	127474.43	117493.61	114750.68	118573.97
	10		83844.35	124292.55	118772.07	149518.75	140356.30
	15		88619.42	114620.32	111939.92	150024.88	163949.59
	20		91144.26	112974.15	112411.10	128770.35	140549.06
	25		91303.38	110270.02	111423.87	125079.49	133919.02
	30		92425.47	114979.58	104955.19	129906.90	118109.13
	35		93188.74	112928.85	110818.12	133300.58	136357.03
0.6	1	118990.89	97677.58	126265.14	122922.44	114547.70	115007.16
	5		82120.23	127182.76	117839.47	124821.56	122104.38
	10		83932.10	124390.89	118421.69	152324.95	135376.70
	15		88541.34	114867.35	112080.19	133811.54	145197.91
	20		91233.91	112633.06	112157.41	151360.19	122047.96
	25		91393.69	110781.51	111090.99	128391.16	134367.69
	30		92433.69	114312.65	104654.74	126393.84	137792.02
	35		93273.26	112073.14	110046.72	128706.49	138486.76
0.8	1	118990.89	97677.58	126168.94	123499.66	114776.5	112861.97
	5		85475.23	129256.13	118155.02	118571.21	123120.25
	10		83932.10	124932.73	118422.68	154956.01	129919.49
	15		88800.76	116518.57	111960.75	134682.55	136186.98
	20		92018.98	112639.27	111708.79	125226.04	136675.16
	25		91587.02	110329.08	110299.59	126324.84	142098.74
	30		93356.40	114235.29	104634.69	125506.67	132401.24
	35		93485.68	111902.74	110030.08	113744.19	131756.82
1	1	118990.89	97677.58	125070.89	123842.56	114376.49	116159.09
	5		84038.03	128931.01	118021.1	122800.22	120195.96
	10		84696.90	125469.21	118141.91	133013.20	129371.67
	15		89736.41	117108.47	111835.02	142023.96	131575.58
	20		92977.68	111945.69	111593.2	127664.65	133721.34
	25		93136.80	109772.42	110987.24	130068.14	128291.21
	30		94315.62	112345.93	104634.69	125831.91	127059.91
	35		95091.91	110943.20	110290.35	126413.42	135914.21

Table 8.11: Average out-of-sample terminal wealths for different stepsize values

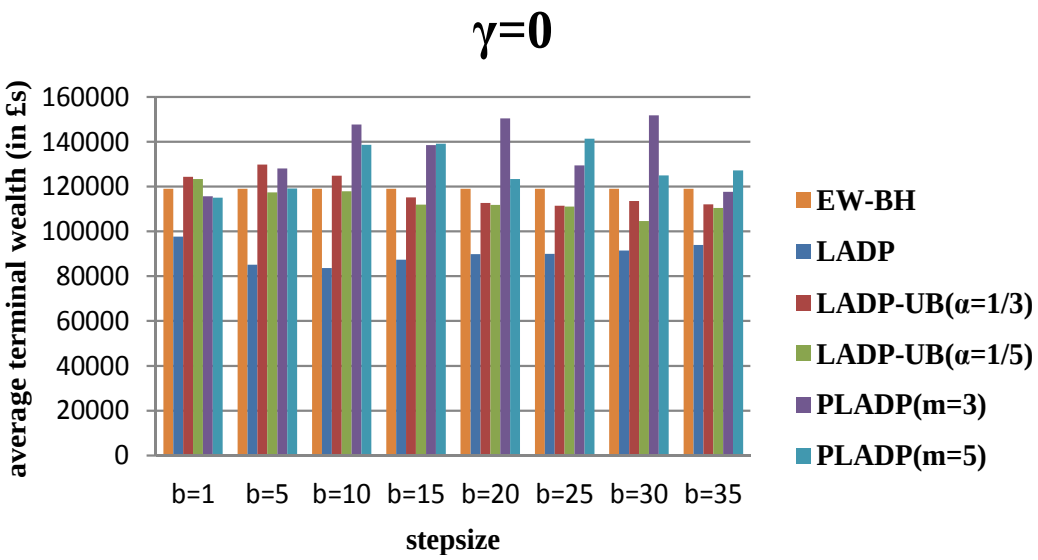


Figure 8.12: Average performance for different stepsizes: $\gamma = 0$

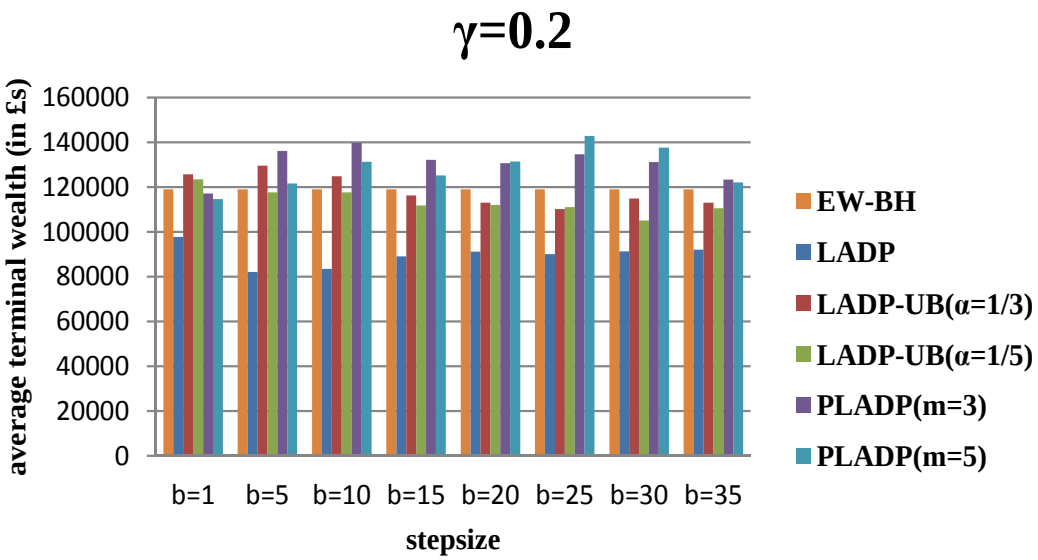


Figure 8.13: Average performance for different stepsizes: $\gamma = 0.2$

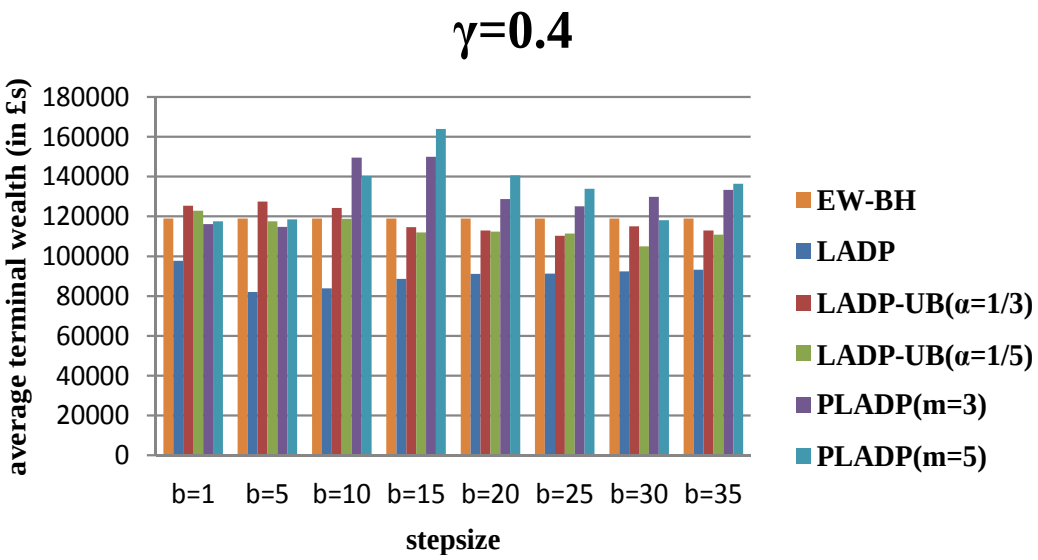


Figure 8.14: Average performance for different stepsizes: $\gamma = 0.4$

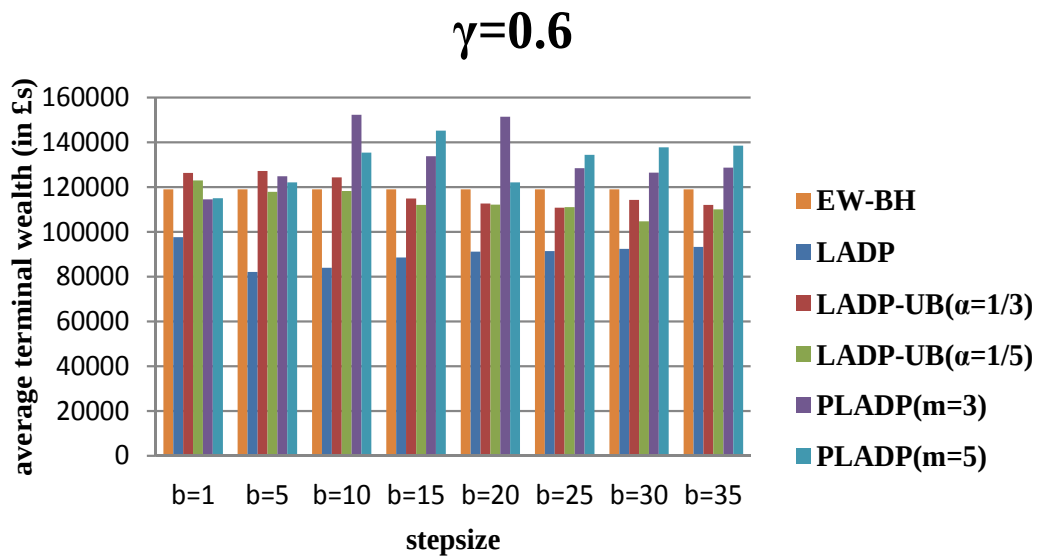


Figure 8.15: Average performance for different stepsizes: $\gamma = 0.6$

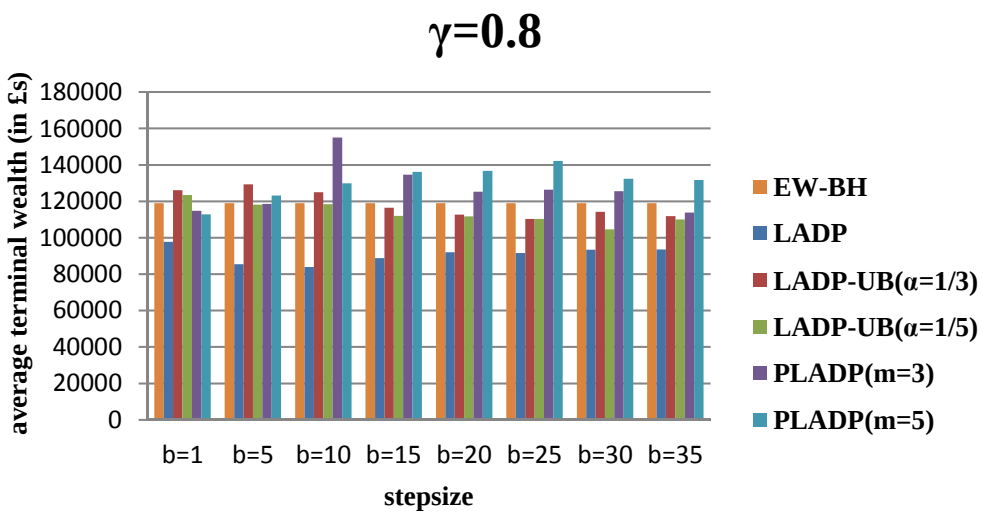


Figure 8.16: Average performance for different stepsizes: $\gamma = 0.8$

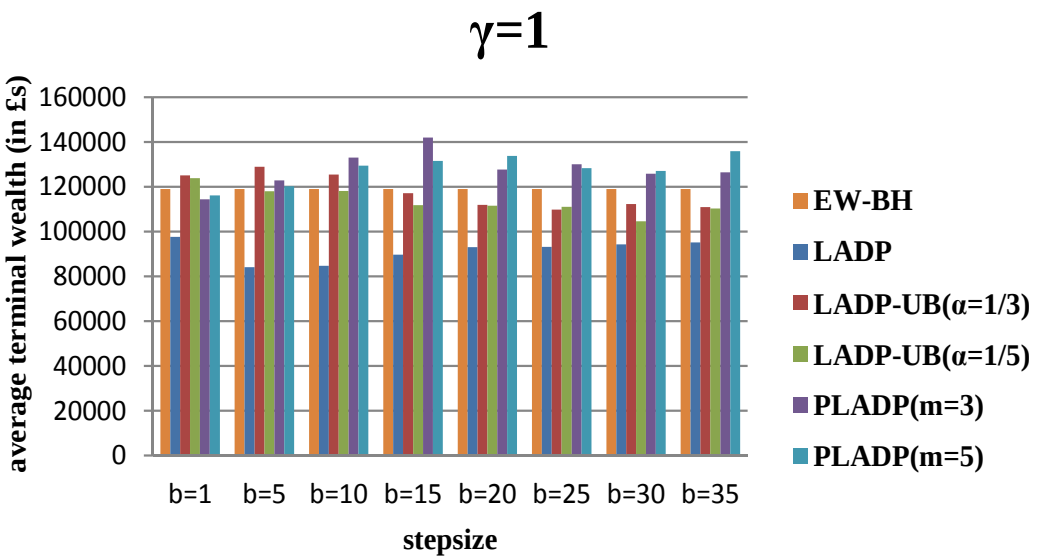


Figure 8.17: Average performance for different stepsizes: $\gamma = 1$

Computational Times

Table 8.12 summarizes the computational times (in seconds) to evaluate each method for each level of the risk importance parameter γ and for each dataset. Note that these times include the times to construct the trees in the multistage stochastic programming method, which are also given separately in Appendix D, as well as the times to perform training iterations in the ADP methods.

Looking at Table 8.12, we notice that the fastest methods, as expected, are the equally-weighted buy-and-hold and fixed-mix strategies, which take us respectively between only 0.0089 seconds and 0.0144 and between only 1.24 and 1.41 seconds to evaluate since they involve solving small linear programs that do not use the scenario returns. In all other methods, however, where we use the scenario returns, time increases significantly. In particular, we notice the following:

1. The second fastest method is the single-period method, where having the CVaR in our objective makes it more difficult for CPLEX to find the optimal solutions for the underlying linear programs and thus requires more computational time. In the Up-Up dataset, for example, removing CVaR from our objective for $\gamma = 1$ takes us only 1.20 seconds to evaluate the single-period method which is approximately 328 times faster than when we have only CVaR in the objective for $\gamma = 0$ which takes us 329.27 seconds.

2. The third fastest method is the simple LADP method where each instance takes us between 529 and 608 seconds.

3. The third fastest method is LADP-UB method, where, due to the additional upper bound constraints on the holdings of the risky assets, it takes us more time to solve than the simple LADP method. In particular, for this method each instance takes us between 649 and 727 seconds.

4. The fourth fastest method is the PLADP method, where computational time increases significantly with the number of slopes. Specifically, for the PLADP method with $m = 3$ slopes each instance takes us between 2467 and 2815 seconds to evaluate, which is approximately 4 times more as compared to an instance of the LADP-UB methods, and for the PLADP method with $m = 5$ slopes each instance takes us between 3631 and 3944 seconds to evaluate, which is approximately 6 times slower than an instance of the LADP-UB methods and approximately 1.5 times slower than an instance of the PLADP

method with $m = 3$ slopes. Note that we have attempted to implement the PLADP methods by including more slopes but for $m \geq 7$ slopes MATLAB would run out of memory.

5. Finally, the less efficient method with respect to computational time is multistage stochastic programming, where in each time period we need to spend time to first construct a scenario tree and then solve a linear program whose size becomes explosive since it combines all scenario returns. For this method, it takes us between 105000 and 111000 seconds to evaluate each instance for $\gamma > 0$ and between 260000 and 315000 seconds to evaluate each instance for $\gamma = 0$. Thus, having only CVaR in the objective makes it much more difficult for CPLEX to converge to an optimal solution. Comparing the times in this method with the times in the PLADP methods, note that the PLADP method with $m = 3$ slopes is approximately 105 times faster than multistage stochastic programming for $\gamma = 0$ and 35 times faster for $\gamma > 0$, and the PLADP method with 5 slopes is approximately 75 times faster than multistage stochastic programming for $\gamma = 0$ and 25 times faster for $\gamma > 0$. Thus, in the multistage stochastic programming method, except for the price we have to pay in our performance due to cutting a significant amount of the input information, we have to pay an additional price which is the time to solve the resulting big linear program.

Note that especially when it comes to comparing the computational times in multistage stochastic programming with the times in the PLADP method with $m = 5$ slopes, it took us cumulatively around 10 days! to evaluate all six instances of multistage stochastic programming in each dataset, while for the PLADP method with $m = 5$ methods we spent only around 6.5 hours per dataset. The reason for this big difference in the computational times is that in the multistage stochastic programming method we rebalance our portfolio every 1 week in order to extract a policy (recall that in this method we use only the first-stage decisions from the scenario trees) as opposed to the PLADP methods which by construction extract a full policy in only one (testing) iteration and as a result take only a few hours. The big discrepancy in the computational times between the two methods makes multistage stochastic programming not affordable and the PLADP method very attractive.

Dataset	γ	EW-BH	EW-FM	SP	MSP	LADP	LADP-UB		PLADP	
							$\alpha = 1/3$	$\alpha = 1/5$	$m = 3$	$m = 5$
Up-Up	0	0.0144	1.24	329.27	300040.43	570.43	714.31	726.98	2755.22	3889.47
	0.2			328.12	109240.82	576.20	715.05	724.86	2746.11	3883.03
	0.4			327.09	107296.69	592.83	715.38	722.72	2814.99	3870.48
	0.6			325.40	107115.05	593.84	714.76	721.70	2720.13	3907.17
	0.8			148.99	106861.49	592.83	716.63	723.61	2735.50	3915.73
	1			1.20	107494.54	571.99	697.56	705.45	2712.32	3943.50
Up-Down	0	0.0121	1.41	326.44	293946.96	547.95	687.98	673.50	2608.88	3751.96
	0.2			322.56	108375.22	546.53	687.95	673.60	2597.75	3861.20
	0.4			323.86	106957.10	546.71	688.62	675.16	2609.83	3767.20
	0.6			321.59	106663.26	545.96	688.54	673.50	2594.24	3777.09
	0.8			299.80	106833.31	546.31	689.17	674.86	2628.05	3772.32
	1			1.21	105491.34	529.96	671.60	656.22	2606.73	3735.47
Down-Up	0	0.0089	1.25	75.54	262853.07	602.71	682.95	674.78	2476.20	3643.94
	0.2			72.54	107791.36	607.17	680.09	673.55	2440.32	3650.84
	0.4			73.10	106671.80	605.21	680.48	674.67	2470.17	3700.60
	0.6			277.59	106016.57	607.09	682.84	674.59	2441.77	3631.10
	0.8			303.20	105856.35	601.07	683.97	676.04	2469.35	3700.75
	1			1.24	106010.86	583.07	665.69	658.19	2467.19	3665.19
Down-Down	0	0.0090	1.27	75.65	314051.46	550.07	676.16	666.62	2677.76	3790.63
	0.2			74.31	110532.99	549.92	672.82	666.67	2704.75	3869.35
	0.4			74.21	109040.47	549.45	673.14	666.73	2692.45	3858.90
	0.6			318.20	107247.24	549.69	674.82	669.08	2722.00	3870.51
	0.8			311.77	107137.44	550.77	673.79	667.43	2695.78	3883.42
	1			1.26	107360.12	534.15	656.22	649.93	2708.07	3844.62

Table 8.12: Computational times in seconds

8.3.3 Expected Terminal Wealth and CVaR of the Different Portfolio Policies

In this section, we report the expected terminal wealths and the CVaR values of the different portfolio policies of the best performers, which are the PLADP methods and the equally-weighted strategies (note that for the PLADP methods we have used the stepsize rule $\frac{25}{24+n_t^s}$).

Tables 8.13-8.14 and 8.15-8.16 summarize respectively the expected terminal wealths and the CVaR values of the PLADP methods and the equally-weighted strategies on the generated 4000 scenarios of returns.

Looking at the expected terminal wealth and the CVaR of the different portfolio policies, we notice that the PLADP methods lead to significantly higher expected terminal wealths but, due to the smaller degree of diversification, they incur higher losses in the 5% worst-case scenarios as compared to the equally-weighted strategies. Further, note that due to the higher transaction costs that we pay in the fixed-mix equally-weighted strategy, the respective expected terminal wealth and CVaR values are worse than the buy-and-hold strategy.

Dataset	γ	**EW-BH**	**EW-FM**	**PLADP**	
				$m = 3$	$m = 5$
Up-Up	0	119005.47	117161.17	132983.17	131262.03
	0.2			132349.23	130842.71
	0.4			132804.22	131961.14
	0.6			131695.56	130753.56
	0.8			132567.43	131405.03
	1			132689.37	129765.05
Up-Down	0	119823.17	115148.13	158495.65	165818.62
	0.2			155706.30	162665.25
	0.4			163145.04	159096.29
	0.6			164007.56	165025.98
	0.8			156068.73	159832.23
	1			159698.48	158932.90

Table 8.13: Up-Up, Up-Down: Expected Terminal Wealth of the Portfolio Policies of the PLADP methods and the equally-weighted strategies

Dataset	γ	EW-BH	EW-FM	PLADP	
				$m = 3$	$m = 5$
Down-Up	0	101221.81	99061.62	134990.65	135270.90
	0.2			141748.42	129411.99
	0.4			129029.84	129368.49
	0.6			135736.07	128660.85
	0.8			126261.31	130354.19
	1			130215.79	121872.74
Down-Down	0	112499.12	110113.66	119328.85	119838.63
	0.2			121271.35	120468.17
	0.4			123511.77	119412.43
	0.6			122165.97	118282.86
	0.8			119843.24	119613.20
	1			120408.37	119395.37

Table 8.14: Down-Up, Down-Down: Expected Terminal Wealth of the Portfolio Policies of the PLADP methods and the equally-weighted strategies

Dataset	γ	EW-BH	EW-FM	PLADP	
				$m = 3$	$m = 5$
Up-Up	0	-8859.14	-6616.67	41058.52	41480.06
	0.2			41597.09	43812.41
	0.4			40561.71	43637.90
	0.6			41043.99	43062.41
	0.8			41319.77	40784.81
	1			39928.16	44600.45
Up-Down	0	-3113.01	2613.12	73361.44	75822.65
	0.2			71262.05	75999.20
	0.4			70818.42	74714.54
	0.6			70362.70	75282.91
	0.8			72419.80	74853.72
	1			73376.45	74591.04

Table 8.15: Up-Up, Up-Down: CVaR of the Portfolio Policies of the PLADP methods and the equally-weighted strategies

Dataset	γ	EW-BH	EW-FM	PLADP	
				$m = 3$	$m = 5$
Down-Up	0	10928.62	15820.76	84661.59	83872.02
	0.2			84820.16	88178.69
	0.4			86405.69	85436.95
	0.6			83940.87	86748.97
	0.8			84614.91	87495.89
	1			86380.63	86991.23
Down-Down	0	-168.37	3045.65	57406.71	60794.52
	0.2			56807.80	60202.18
	0.4			58382.93	60285.34
	0.6			56010.31	61657.34
	0.8			58029.84	60284.24
	1			59366.05	60682.45

Table 8.16: Down-Up, Down-Down: CVaR of the Portfolio Policies of the PLADP methods and the equally-weighted strategies

8.3.4 Impact of Length of Planning Horizon

In this section, we study the impact of the length of the planning horizon and the number of periods considered. Recall that in the simulation results presented above we used in-sample data of 104 weeks to generate 4000 out-of-sample scenarios of returns, each scenario with length 52 weeks. The generated scenarios of returns were then used to extract the portfolio policies of the approximate dynamic programming methods and the other portfolio selection methods used as benchmarks, which were then evaluated and compared in terms of the achieved out-of-sample wealth at the end of the out-of-sample horizon.

Here, we examine the impact of the length of the planning horizon by considering smaller out-of-sample horizons and running "rolling-horizon" simulations (see for example [20], [21], [23], [79] and [86]). In particular, assuming that the in-sample data consist of the most recent 104 weeks, as above, we generate 4000 out-of-sample scenarios of returns and extract the associated portfolio policies which now consist of fewer time periods. After evaluating the extracted policies using the corresponding out-of-sample actual returns (see section 8.1 for a discussion on how we evaluate the extracted portfolio policies), we update the in-sample data to include the actual returns that have been revealed (note that these are the actual returns of the previous planning horizon), we generate a new set of 4000 scenarios of returns for the new planning horizon and we use them to extract the new associated portfolio policies, and so on.

In the "rolling-horizon" simulation results presented below, we considered plan-

ning horizons of 13 and 26 weeks and we rolled forward respectively 4 and 2 times (in this manner in the end we will have reached the 52^{nd} out-of-sample week) updating as we go the in-sample data respectively with the actual returns of the new 13 and 26 weeks that were revealed each time. Note that the impact of the length of the planning horizon is examined for two levels of the risk importance parameter γ, which are $\gamma = 0.2$ and 0.6, and the composition of the selected portfolios as well as the achieved wealths at the end of each out-of-sample horizon are recorded. As benchmarks to compare with the approximate dynamic programming methods we used the equally-weighted strategies and the market.

Table 8.17 summarizes the composition of the selected portfolios during the 52 out-of-sample weeks (see section 8.1 for an explanation of the reported measures), where we notice that for longer look-ahead horizons the PLADP methods invest in more risky assets and select more diversified portfolios. One possible explanation for this could be that in longer horizons the investor is faced with a longer uncertain future (i.e. there is more uncertainty ahead) when taking his decisions and is trying to compensate for future losses in risky assets by selecting more diversified portfolios. Further, as expected, for all planning horizons in the PLADP method with $m = 5$ slopes we have more diversified portfolios and we use more assets out-of-sample as compared to the PLADP method with $m = 3$ slopes.

13 weeks									
γ	**Dataset**	$m = 3$				$m = 5$			
		NA$_t$	**NA**	**Cash**	**HI***	**NA**$_t$	**NA**	**Cash**	**HI***
0.2	Up-Up	3-12	25	0	0.21-0.89	3-12	27	0	0.21-0.77
	Up-Down	2-8	23	0	0.25-0.95	2-9	19	0	0.29-0.83
	Down-Up	1-7	18	0	0.25-1	2-5	12	0	0.22-0.85
	Down-Down	2-9	26	0	0.23-0.90	2-9	24	0	0.21-0.76
0.6	Up-Up	4-11	25	0	0.21-0.79	4-13	25	0	0.24-0.75
	Up-Down	2-8	20	0	0.18-0.94	2-14	19	0	0.25-0.70
	Down-Up	1-6	13	0-1	0.29-1	2-6	15	0	0.25-0.78
	Down-Down	2-9	22	0	0.23-0.89	2-13	25	0	0.20-0.83
26 weeks									
γ	**Dataset**	$m = 3$				$m = 5$			
		NA$_t$	**NA**	**Cash**	**HI***	**NA**$_t$	**NA**	**Cash**	**HI***
0.2	Up-Up	3-19	29	0	0.22-0.89	2-19	32	0	0.22-0.99
	Up-Down	1-11	18	0	0.17-1	2-20	30	0	0.13-0.88
	Down-Up	2-5	17	0	0.25-0.98	2-7	24	0	0.23-0.95
	Down-Down	2-9	24	0	0.23-0.76	2-11	23	0	0.15-0.99
0.6	Up-Up	4-13	23	0	0.15-0.85	4-12	25	0	0.20-0.84
	Up-Down	2-11	22	0	0.21-0.99	3-26	34	0	0.14-0.68
	Down-Up	2-5	18	0	0.25-0.91	2-9	24	0	0.24-0.92
	Down-Down	1-10	24	0	0.26-1	1-12	27	0	0.20-1
52 weeks									
γ	**Dataset**	$m = 3$				$m = 5$			
		NA$_t$	**NA**	**Cash**	**HI***	**NA**$_t$	**NA**	**Cash**	**HI***
0.2	Up-Up	3-33	33	0	0.18-0.83	3-22	31	0	0.18-0.86
	Up-Down	3-18	34	0	0.11-0.80	3-34	41	0	0.13-0.80
	Down-Up	2-19	33	0-1	0.20-0.97	2-27	45	0	0.12-0.95
	Down-Down	4-16	33	0	0.19-0.64	3-34	51	0	0.15-0.66
0.6	Up-Up	3-25	38	0	0.18-0.84	4-29	34	0	0.21-0.83
	Up-Down	3-20	31	0	0.18-0.84	3-25	36	0	0.14-0.79
	Down-Up	2-21	39	0	0.21-0.88	3-30	45	0	0.14-0.85
	Down-Down	5-23	38	0	0.16-0.63	3-34	47	0	0.14-0.66

Table 8.17: Characteristics of selected portfolios for planning horizons of 13, 26 and 52 weeks

Tables 8.18 and 8.19 summarize the out-of-sample wealths at times $t = 0, 13, 26,$ 39 and 52 in the PLADP methods and the benchmarks for the considered planning horizons of 13, 26 and 52 weeks. In terms of the achieved out-of-sample wealths at the different points in time, note the following:

- In the Up-Up dataset, for a planning horizon of 13 weeks the PLADP methods outperform the benchmarks only at times $t = 39$ and 52, for a planning horizon of 26 weeks the PLADP methods outperform the benchmarks at most points in time and for a planning horizon of 52 weeks the PLADP methods perform better than the benchmarks only at times $t = 26$ and 39.

- In the Up-Down dataset, for a planning horizon of 13 weeks the PLADP

methods are doing worse than the benchmarks at most points in time (except for the market which is below the PLADP methods at most points in time except for $t = 52$), while for planning horizons of 26 (except for $\gamma = 0.2$ and $m = 3$) and 52 weeks the PLADP methods perform better than the benchmarks at most points in time.

- In the Down-Up dataset, for a planning horizon of 13 weeks the PLADP methods perform worse than the benchmarks at most points in time (except for the market which is below the PLADP methods at most points in time), while for planning horizons of 26 and 52 weeks the PLADP methods outperform the benchmarks at most points in time.

- In the Down-Down dataset, the higher degree of diversification in the selected portfolios of the 52-week planning horizon leads to a better performance than the benchmarks at most points in time, while for planning horizons of 13 and 26 weeks at most points in time we are doing worse than the benchmarks and experience significant losses.

Table 8.20 summarizes the number of instances out of the 8 (2 levels of risk $\times$ 4 datasets) each PLADP method outperforms each benchmark. Looking at Table 8.20, we notice that for a planning horizon of 13 weeks the PLADP methods outperform the benchmarks in less than half of the instances at most points in time, while for planning horizons of 26 and 52 weeks the PLADP methods perform better than the benchmarks in more than half of the instances at most points in time. Thus, the higher degree of diversification in the selected portfolios for planning horizons of 26 and 52 weeks seem to help PLADP methods perform way better than when we have a planning horizon of 13 weeks. Further, note that for all three planning horizons the PLADP method with $m = 5$ slopes performs better than the benchmarks in more instances than the PLADP method with $m = 3$ slopes.

Up-Up										
γ	t	FTSE	EW-BH	EW-FM	PLADP					
					13 weeks		26 weeks		52 weeks	
					$m=3$	$m=5$	$m=3$	$m=5$	$m=3$	$m=5$
0.2	0	100000	100000	100000	100000	100000	100000	100000	100000	100000
	13	114246.07	106926.92	106838.37	99036.54	106946.99	101276.82	114565.23	113699.94	109905.41
	26	124658.37	118865.93	119454.10	117345.40	118815.50	141095.80	141418.55	136085.69	130721.05
	39	119218.20	112938.58	112454.46	117274.26	155135.12	138400.38	134654.58	137425.36	132353.68
	52	**143601.38**	**130011.93**	**128821.15**	**131497.72**	**184506.41**	**148662.08**	**137873.16**	**127064.39**	**123768.05**
0.6	0	100000	100000	100000	100000	100000	100000	100000	100000	100000
	13	114246.07	106926.92	106838.37	107087.09	100822.31	113398.18	115798.05	112946.40	107743.47
	26	124658.37	118865.93	119454.10	106657.36	112377.22	136842.77	139524.48	127868.17	119399.11
	39	119218.20	112938.58	112454.46	131994.05	132451.74	143385.03	134853.67	119601.29	119469.45
	52	**143601.38**	**130011.93**	**128821.15**	**162230.31**	**172425.84**	**153939.56**	**169453.07**	**108087.36**	**120966.91**

Up-Down										
γ	t	FTSE	EW-BH	EW-FM	PLADP					
					13 weeks		26 weeks		52 weeks	
					$m=3$	$m=5$	$m=3$	$m=5$	$m=3$	$m=5$
0.2	0	100000	100000	100000	100000	100000	100000	100000	100000	100000
	13	93245.51	95626.97	97230.94	86511.90	94408.30	87880.93	100484.07	89149.51	98955.26
	26	93365.27	102683.15	106158.05	103151.43	96518.73	96515.21	118222.34	109196.57	121749.01
	39	90682.81	100015.59	101411.64	97849.98	96687.12	80643.46	137689.61	101772.25	126172.63
	52	**89788.17**	**105262.25**	**106652.30**	**85625.75**	**93929.96**	**73423.45**	**122513.18**	**105383.91**	**117407.22**
0.6	0	100000	100000	100000	100000	100000	100000	100000	100000	100000
	13	93245.51	95626.97	97230.94	99357.45	92861.08	107691.99	97698.01	92354.16	92024.46
	26	93365.27	102683.15	106158.05	111976.45	95158.11	116185.65	112112.47	105396.15	114958.50
	39	90682.81	100015.59	101411.64	98875.82	97302.85	124859.03	129231.94	102098.76	121192.57
	52	**89788.17**	**105262.25**	**106652.30**	**89679.17**	**84245.10**	**112812.51**	**128416.45**	**90676.71**	**100336.32**

Table 8.18: Up-Up, Up-Down: Out-of-sample wealths at times $t = 0, 13, 26, 39$ and 52 in the PLADP methods and the benchmarks for planning horizons of 13, 26 and 52 weeks

Down-Up										
γ	t	FTSE	EW-BH	EW-FM	PLADP					
					13 weeks		26 weeks		52 weeks	
					$m=3$	$m=5$	$m=3$	$m=5$	$m=3$	$m=5$
0.2	0	100000	100000	100000	100000	100000	100000	100000	100000	100000
	13	120171.71	128089.26	128300.01	132418.80	131377.71	139948.55	133498.64	144232.52	130441.10
	26	124915.60	139597.77	139258.95	146206.56	134247.99	156106.88	152202.45	152239.54	162100.15
	39	126885.91	139290.23	139906.47	137416.43	123328.43	140872.18	126733.16	146693.89	157796.04
	52	**132532.01**	**156160.11**	**155357.50**	**147920.13**	**136957.43**	**166599.80**	**165241.58**	**217338.56**	**222927.47**
0.6	0	100000	100000	100000	100000	100000	100000	100000	100000	100000
	13	120171.71	128089.26	128300.01	138190.28	126325.75	155174.38	137586.23	143441.80	150165.43
	26	124915.60	139597.77	139258.95	113249.18	110137.62	188415.34	156487.27	156065.54	164733.30
	39	126885.91	139290.23	139906.47	102666.39	103326.45	170715.12	134517.791	150408.40	162238.87
	52	**132532.01**	**156160.11**	**155357.50**	**112646.38**	**122581.25**	**207252.50**	**177727.77**	**200196.25**	**204696.62**

Down-Down										
γ	t	FTSE	EW-BH	EW-FM	PLADP					
					13 weeks		26 weeks		52 weeks	
					$m=3$	$m=5$	$m=3$	$m=5$	$m=3$	$m=5$
0.2	0	100000	100000	100000	100000	100000	100000	100000	100000	100000
	13	100651.67	106467.52	106540.28	102194.89	87560.47	98686.49	114728.11	113677.80	116877.67
	26	87146.85	96283.54	95498.90	77468.53	75159.77	91978.50	104218.78	97229.29	106351.93
	39	72783.38	82170.88	80746.25	65564.07	56880.83	83054.69	76170.55	78986.18	92609.47
	52	**75524.21**	**84529.28**	**84227.81**	**69905.35**	**57450.35**	**77257.27**	**78298.11**	**88808.88**	**106934.62**
0.6	0	100000	100000	100000	100000	100000	100000	100000	100000	100000
	13	100651.67	106467.52	106540.28	97369.33	108970.65	112036.80	105432.65	113539.60	115973.79
	26	87146.85	96283.54	95498.90	77837.12	96101.04	104983.61	92412.49	105161.09	104058.43
	39	72783.38	82170.88	80746.25	63186.35	68712.82	60960.86	72988.41	99849.20	94224.82
	52	**75524.21**	**84529.28**	**84227.81**	**62684.63**	**78611.74**	**62138.14**	**67986.29**	**114604.32**	**111470.92**

Table 8.19: Down-Up, Down-Down: Out-of-sample wealths at times $t = 0, 13, 26, 39$ and 52 in the PLADP methods and the benchmarks for planning horizons of 13, 26 and 52 weeks

# of instances = 8	t	PLADP					
		13 weeks		26 weeks		52 weeks	
		$m = 3$	$m = 5$	$m = 3$	$m = 5$	$m = 3$	$m = 5$
FTSE	13	4	4	4	8	4	5
	26	3	4	8	8	8	7
	39	4	4	6	7	8	8
	52	2	5	6	6	6	6
EW-BH	13	4	3	5	7	6	7
	26	3	0	6	7	8	8
	39	2	2	6	4	7	8
	52	2	2	5	6	5	5
EW-FM	13	4	3	5	7	6	7
	26	2	1	6	7	7	7
	39	2	2	6	4	7	8
	52	2	2	5	6	4	5

Table 8.20: In how many instances out of the 8 a column PLADP method outperforms a row method at times $t = 13, 26, 39$ and 52 and for planning horizons of 13, 26 and 52 weeks

For planning horizons of 13, 26 and 52 weeks, Tables 8.21 and 8.22 summarize respectively the average out-of-sample wealths at times $t = 13, 26, 39$ and 52 across the four datasets and the number of instances out of the 2 (2 levels of risk) a column PLADP method outperforms a row method. Looking at Table 8.21, we notice that in the PLADP method with $m = 5$ slopes the average terminal wealths are higher than in the PLADP method with $m = 3$ slopes at most points in time. Further, note that as compared to the benchmarks (see Table 8.22), the average performance improves for longer planning horizons. In particular, for a planning horizon of 13 weeks the PLADP methods underperform the benchmarks in at least half of the instances at most points in time. For a planning horizon of 26 weeks, the PLADP method with $m = 3$ slopes performs better than the benchmarks in all instances at most points in time, while the PLADP method with $m = 5$ slopes outperforms the benchmarks in all instances at all points in time. Finally, for a planning horizon of 52 weeks the average performance of the PLADP methods is better than the average performance of the benchmarks everywhere.

Average performance											
γ	t	FTSE	EW-BH	EW-FM	PLADP						
					13 weeks		26 weeks		52 weeks		
					$m=3$	$m=5$	$m=3$	$m=5$	$m=3$	$m=5$
0.2	0	100000	100000	100000	100000	100000	100000	100000	100000	100000
	13	107078.74	109277.67	109727.40	105040.53	105073.37	106948.20	115819.01	115189.94	114044.86
	26	107521.52	114357.60	115092.50	111042.98	106185.50	121424.10	129015.53	123687.77	130230.54
	39	102392.57	108603.82	108629.71	104526.18	108007.87	110742.68	118811.97	116286.92	127232.96
	52	**110361.44**	**118990.89**	**118764.69**	**108737.24**	**118211.04**	**116485.65**	**125981.51**	**134648.93**	**142759.34**
0.6	0	100000	100000	100000	100000	100000	100000	100000	100000	100000
	13	107078.74	109277.67	109727.40	110501.04	107244.95	122075.34	114128.73	115570.49	116476.79
	26	107521.52	114357.60	115092.50	102430.03	103443.50	136606.85	125134.18	123622.74	125787.33
	39	102392.57	108603.82	108629.71	99180.65	100448.46	124980.01	117897.95	117989.41	124281.43
	52	**110361.44**	**118990.89**	**118764.69**	**106810.12**	**114465.98**	**134035.68**	**135895.90**	**128391.16**	**134367.69**

Table 8.21: Average out-of-sample wealths at times $t = 0, 13, 26, 39$ and 52 in the PLADP methods and the benchmarks for planning horizons of 13, 26 and 52 weeks

# of instances = 2	t	PLADP					
		13 weeks		26 weeks		52 weeks	
		$m=3$	$m=5$	$m=3$	$m=5$	$m=3$	$m=5$
FTSE	13	1	1	1	2	2	2
	26	1	0	2	2	2	2
	39	1	1	2	2	2	2
	52	0	2	2	2	2	2
EW-BH, EW-FM	13	1	0	1	2	2	2
	26	0	0	2	2	2	2
	39	0	0	2	2	2	2
	52	0	0	1	2	2	2

Table 8.22: Average performance: In how many instances out of the 2 a column PLADP method outperforms a row method at times $t = 13, 26, 39$ and 52 and for planning horizons of 13, 26 and 52 weeks

Finally, Figures 8.18-8.19 show how the average cumulative wealth changes with time for each method for the selected levels of risk and one can clearly see that the PLADP methods work better on average in longer horizons.

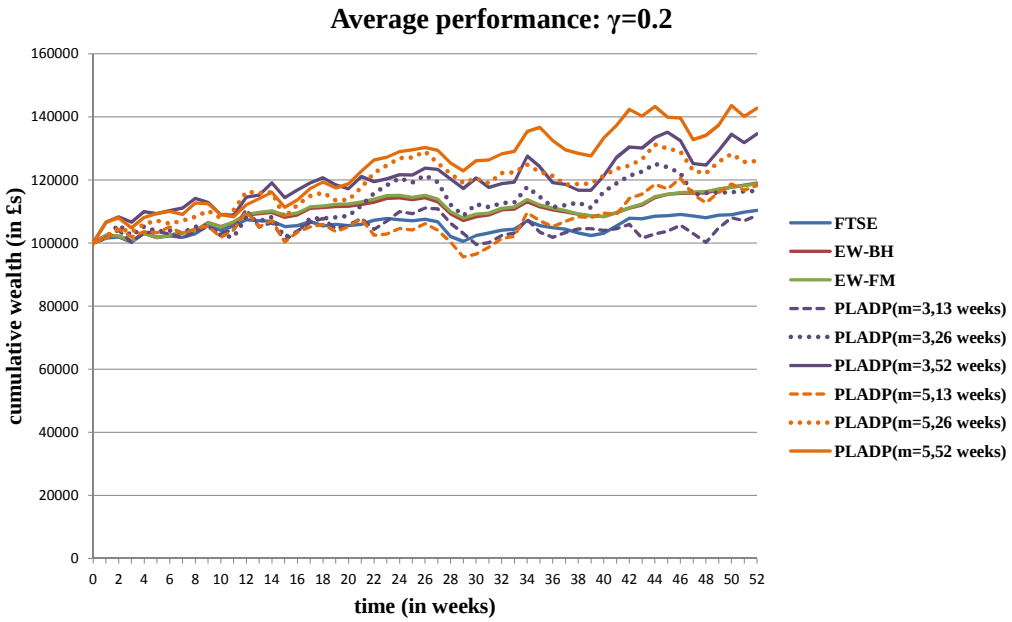


Figure 8.18: Average out-of-sample cumulative wealth for $\gamma = 0.2$ and planning horizons of $13, 26$ and 52 weeks

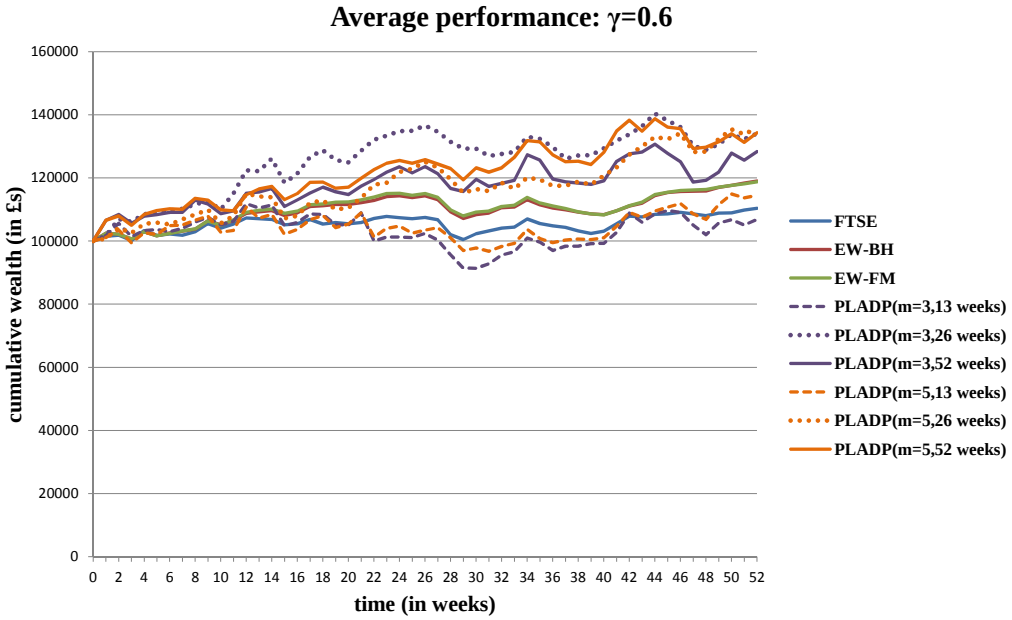


Figure 8.19: Average out-of-sample cumulative wealth for $\gamma = 0.6$ and planning horizons of $13, 26$ and 52 weeks

8.3.5 Impact of Transaction Costs

In this section, we examine the impact of transaction costs in the composition of the selected portfolios and the performance of the best performers, which are the PLADP methods and the equally-weighted strategies, by increasing the value of the transaction costs parameter θ from 0.2% to 0.5%.

Tables 8.23-8.24 summarize the out-of-sample terminal wealths and the total transaction costs paid in the equally-weighted strategies for $\theta = 0.2\%$ and 0.5%, where we notice that in the presence of higher transaction costs the terminal wealth values deteriorate in both strategies. Further, note that, although the discrepancies in the terminal wealth values in the two strategies are negligible, the equally-weighted buy-and-hold strategy is doing slightly better than the fixed-mix one in all datasets except for the Up-Down dataset, where the buy-and-hold strategy is slightly below the fixed-mix one.

Table 8.25 summarizes the percentage differences of the out-of-sample terminal wealth values of the equally-weighted buy-and-hold strategy from the fixed-mix one for $\theta = 0.2\%$ and 0.5%, where we notice that in the presence of higher transaction costs, that is after increasing the transaction costs parameter θ from 0.2%

to 0.5%, the relative performance of the equally-weighted buy-and-hold strategy as compared to the fixed-mix one improves further.

Strategy	**Up-Up**			
	$\boldsymbol{\theta = 0.2\%}$		$\boldsymbol{\theta = 0.5\%}$	
	Terminal wealth	**Transaction costs**	**Terminal wealth**	**Transaction costs**
EW-BH	130011.93	192.70	129621.25	492.61
EW-FM	128821.15	495.32	127898.67	1264.56
Strategy	**Up-Down**			
	$\boldsymbol{\theta = 0.2\%}$		$\boldsymbol{\theta = 0.5\%}$	
	Terminal wealth	**Transaction costs**	**Terminal wealth**	**Transaction costs**
EW-BH	105262.25	192.70	104945.94	492.61
EW-FM	106652.30	622.00	105619.58	1586.31

Table 8.23: Up-Up and Up-Down datasets: Out-of-sample terminal wealths and total transaction costs paid in the equally-weighted strategies for $\theta = 0.2\%$ and 0.5%

Strategy	**Down-Up**			
	$\boldsymbol{\theta = 0.2\%}$		$\boldsymbol{\theta = 0.5\%}$	
	Terminal wealth	**Transaction costs**	**Terminal wealth**	**Transaction costs**
EW-BH	156160.11	192.70	155690.86	492.61
EW-FM	155357.50	527.40	154274.59	1346.27
Strategy	**Down-Down**			
	$\boldsymbol{\theta = 0.2\%}$		$\boldsymbol{\theta = 0.5\%}$	
	Terminal wealth	**Transaction costs**	**Terminal wealth**	**Transaction costs**
EW-BH	84259.28	192.70	84275.27	492.61
EW-FM	84227.81	484.58	83563.58	1237.05

Table 8.24: Down-Up and Down-Down datasets: Out-of-sample terminal wealths and total transaction costs paid in the equally-weighted strategies for $\theta = 0.2\%$ and 0.5%

Transaction costs	**Up-Up**	**Up-Down**	**Down-Up**	**Down-Down**
$\boldsymbol{\theta = 0.2\%}$	0.92%	-1.30 %	0.52 %	0.36 %
$\boldsymbol{\theta = 0.5\%}$	1.35 %	-0.64 %	0.92 %	0.85%

Table 8.25: Percentage differences of the terminal wealths in the fixed-mix equally-weighted strategy from the buy-and-hold one for $\theta = 0.2\%$ and 0.5%

To examine the impact of the transaction costs in the PLADP methods (note that as above we use stepsize rule $\frac{25}{24+n_t^s}$), we have considered two levels of the risk importance parameter γ, which are $\gamma = 0.2$ and 0.6, and we have recorded the composition of the selected portfolios as well as the terminal wealth values when transaction costs parameter θ changes from 0.2% to 0.5%.

Table 8.26 summarizes the composition of the selected portfolios, the terminal wealth values as well as the total transaction costs paid in the PLADP methods for

the selected levels of the risk importance parameter γ. Further, Table 8.27 summarizes the number of instances out of the 8 (2 levels of risk $\times$ 4 datasets) the PLADP methods outperform the benchmarks (note that we use as benchmarks the equally-weighted strategies and the market). Regarding the composition of the selected portfolios, we notice that for $\theta = 0.5\%$ the investor is trying to trade off the higher trading costs by trading less frequently (this involves holding the same portfolio for longer periods and using less risky assets in some of the instances) throughout the out-of-sample horizon and selecting more diversified portfolios in most of the instances (this involves paying higher transaction costs) as compared to $\theta = 0.2\%$. In terms of the actual performance, although we have to pay a higher price due to transaction costs which is reflected in the significantly lower terminal wealth values in most instances, the PLADP methods are still doing better than the equally-weighted strategies and the market in at least half of the 8 instances.

Further, in terms of the average performance, Table 8.28 summarizes the average terminal wealths in the PLADP methods and the benchmarks for two levels of the transaction costs parameter θ, where we notice that, although the average terminal wealths in the PLADP methods deteriorate in the presence of higher transaction costs, they are still doing better than the benchmarks.

Dataset	θ	PLADP	$\gamma = 0.2$						$\gamma = 0.6$					
			NA_t	NA	Cash	$HI^\star$	Terminal wealth	Transaction costs	NA_t	NA	Cash	$HI^\star$	Terminal wealth	Transaction costs
Up-Up	0.2%	$m = 3$	3-33	33	0	0.18-0.83	127064.39	12528.85	3-25	38	0	0.18-0.84	108087.36	12905.78
		$m = 5$	3-22	31	0	0.18-0.86	123768.05	13109.65	4-29	34	0	0.21-0.83	120966.91	13158.43
	0.5%	$m = 3$	3-31	37	0-1	0.19-0.83	94356.12	26619.80	4-28	32	0	0.16-0.83	91924.87	27394.73
		$m = 5$	4-35	39	0	0.19-0.84	108522.66	27307.86	4-27	37	0	0.21-0.83	105025.61	27579.61
Up-Down	0.2%	$m = 3$	3-18	34	0	0.11-0.80	105383.91	11176.69	3-20	31	0	0.18-0.84	90676.71	10744.14
		$m = 5$	3-34	41	0	0.13-0.80	117407.22	11043.59	3-25	36	0	0.14-0.79	100336.32	10196.94
	0.5%	$m = 3$	3-41	48	0-1	0.15-0.83	83688.42	21468.44	3-31	42	0	0.16-0.78	91974.43	21206.90
		$m = 5$	3-27	36	0	0.09-0.76	82202.09	20394.11	3-31	35	0	0.10-0.76	84814.29	20562.29
Down-Up	0.2%	$m = 3$	2-19	33	0-1	0.20-0.97	217338.56	12581.43	2-21	39	0	0.21-0.88	200196.25	16739.76
		$m = 5$	2-27	45	0	0.12-0.95	222927.47	14160.19	3-30	45	0	0.14-0.85	204696.62	17067.14
	0.5%	$m = 3$	3-42	53	0-1	0.16-0.96	247804.09	29325.55	3-38	48	0-1	0.15-0.91	212990.68	26598.42
		$m = 5$	3-51	56	0	0.11-0.91	210906.02	20608.12	3-43	55	0-1	0.09-0.86	205052.04	25801.18
Down-Down	0.2%	$m = 3$	4-16	33	0	0.19-0.64	88808.88	10755.74	5-23	38	0	0.16-0.63	114604.32	11134.66
		$m = 5$	3-34	51	0	0.15-0.66	106934.62	11168.03	3-34	47	0	0.14-0.66	111470.92	11599.92
	0.5%	$m = 3$	4-49	50	0-1	0.14-0.63	84940.79	21524.68	4-46	51	0-1	0.16-0.63	87502.55	21111.98
		$m = 5$	4-42	49	0	0.18-0.65	89613.66	22844.69	3-36	49	0	0.14-0.66	85126.46	23362.49

Table 8.26: Characteristics of selected portfolios, terminal wealth values and total transaction costs paid in PLADP methods for $\theta = 0.2\%, 0.5\%$

# of instances = 8 Strategy	PLADP			
	$m = 3$		$m = 5$	
	$\theta = 0.2\%$	$\theta = 0.5\%$	$\theta = 0.2\%$	$\theta = 0.5\%$
FTSE	6	5	6	4
EW-BH	5	4	5	4
EW-FM	4	4	5	4

Table 8.27: In how many instances out of the 8 the PLADP methods outperform the equally-weighted strategies and the market

γ	FTSE	EW-BH	EW-FM	PLADP			
				$m = 3$		$m = 5$	
				$\theta = 0.2\%$	$\theta = 0.5\%$	$\theta = 0.2\%$	$\theta = 0.5\%$
0.2	110361.44	118990.89	118764.69	134648.93	127697.36	142759.34	122811.11
0.6				128391.16	121098.13	134367.69	120004.60

Table 8.28: Average out-of-sample terminal wealths for $\theta = 0.2\%$ and 0.5%

Finally, Figures 8.20-8.21 show how the average cumulative wealth changes with time for the selected levels of risk and for $\theta = 0.2\%$ and 0.5%, where one can clearly see that the impact of increasing transaction costs parameter θ is more significant in the PLADP method with $m = 5$ slopes than in the PLADP method with $m = 3$ slopes. The reason for this is that, as explained above, when we have more slopes the PLADP method trades more frequently in the sense that it selects more diversified portfolios which results in us paying higher trading costs.

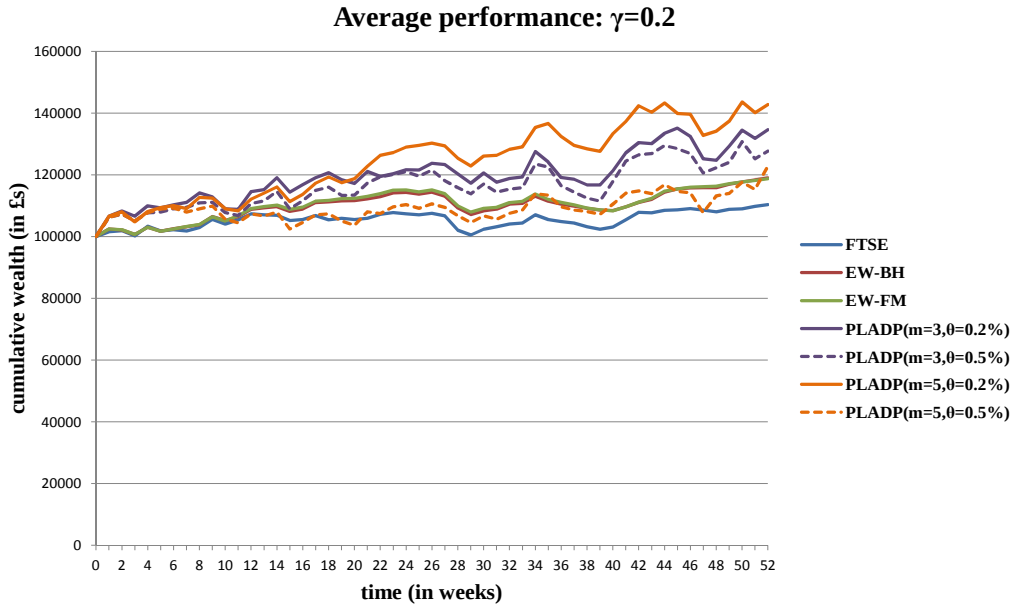


Figure 8.20: Average out-of-sample cumulative wealth for $\gamma = 0.2$ and $\theta = 0.2\%$ and 0.5%

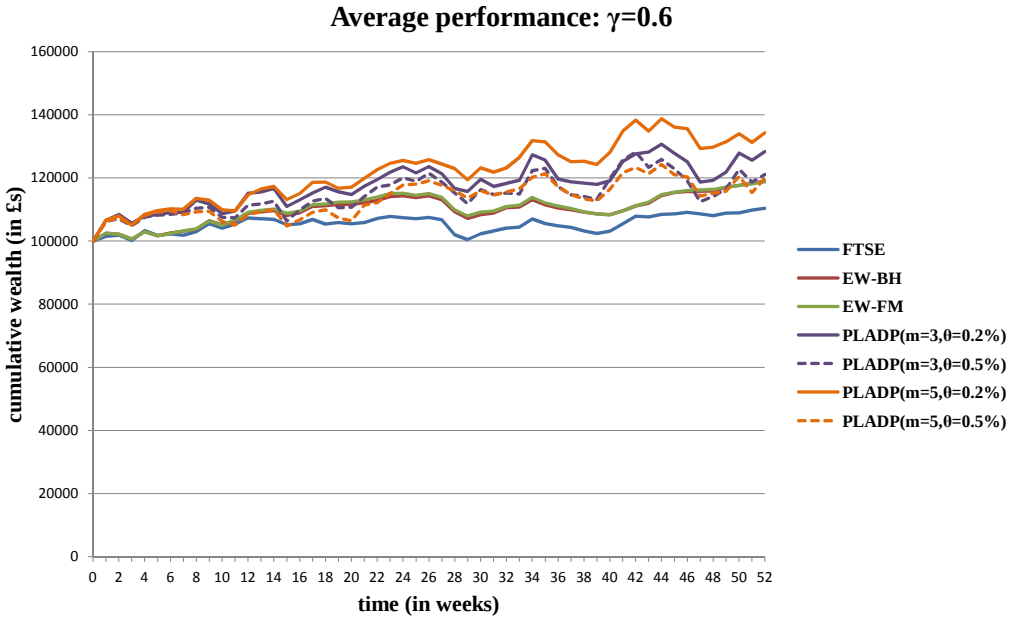


Figure 8.21: Average out-of-sample cumulative wealth for $\gamma = 0.6$ and $\theta = 0.2\%$ and 0.5%

Chapter 9

Conclusions and Future Research

In this chapter, considering our simulation results from chapter 8, we provide our concluding remarks and discuss directions for future research.

9.1 Conclusions

In the literature, ADP methods have many applications and they have proved to work considerably well. To our knowledge, however, these methods have not been tested in the portfolio selection problem and in this thesis we attempted to fill exactly this gap by constructing ADP algorithms for the multi-period portfolio selection problem, where we used CVaR as a measure of risk and we assumed that transactions costs are proportional to the trading amounts.

To evaluate the extracted policies from our ADP algorithms, we performed extensive simulations where we compared them out-of-sample against other portfolio selection methods, both myopic and multi-period, as well as against the market portfolio. In particular, for our comparisons we formulated a multi-period portfolio model as a multistage stochastic program and solved it by approximating the stochastic process of random returns with a scenario tree. Further, we formulated and solved the equally-weighted and a single-period portfolio models.

In our simulations, we concentrated on how our ADP algorithms perform not only when the market conditions are stable but also when they change. For this reason, we applied and compared all methods using four datasets from the FTSE 100 Index that had the same and opposite market trends in the in-sample and out-of-sample data.

Observing the numerical results from our simulations in chapter 8, we feel that

we can draw the following conclusions:

1. Due to the assumed linear functional approximations in the LADP methods, where in each asset every additional unit of wealth is valued the same, these ignore risk and exhibit a similar behavior to the multistage stochastic programming method without risk. As far as the simple LADP method is concerned, this can have a serious impact on our portfolio value and can lead to catastrophic losses in highly volatile market conditions. And this is where the importance of imposing restrictions on how much we can have in each asset lies. Our simulation results suggest that, despite ignoring risk, we can prevent high losses and improve our performance by adding constraints that allow for more diversified portfolios as in the LADP-UB methods.

2. Due to the assumed piecewise linear functional approximations in the PLADP methods, which track better the utility function of a risk-averse investor, we have natural upper bounds on how much we can hold in each asset. These allow us to hold more diversified portfolios by jumping from one asset to another if we think that increasing our wealth in an asset has less value than start using another one. In our simulations, there is evidence that we can benefit from using more slopes in the approximate value functions in terms of our performance, but we have to pay a price in computational time which increases significantly even for a few more slopes. For our data, the PLADP methods outperform significantly all other methods including the perfectly diversified equally-weighted method which seems to be the next best performer and which was the most difficult to beat.

3. When it comes to comparing multistage stochastic programming with dynamic programming, the two methods are similar in the sense that from the multistage stochastic programming formulation we can derive the optimality equations of the dynamic programming formulation of the same stochastic problem. What these two methods differ in are the solution approaches. On the one hand, to solve a multistage stochastic program we first need to reduce the input distribution of the underlying stochastic process and represent it as a scenario tree, and then on the generated tree we write the deterministic equivalent problem and solve it optimally using optimization software. On the other hand, for dynamic programming, if we can't use traditional dynamic programming algorithms to solve it due to the explosive state space, we resort

to approximate dynamic programming methods. Unlike multistage stochastic programming where we reduce the input information and solve a big problem optimally, in approximate dynamic programming we use all the generated scenario paths and our problem decomposes in time in smaller problems but is solved suboptimally. Considering this trade-off between optimality and the amount of information that we use in each of the two methods, our simulation results suggest that the price we have to pay in terms of performance and computational time in multistage stochastic programming is too high and thus it is not worth it going for it just for the sake of "optimality".

4. Diversification is very important in achieving our long-term financial goals as it can reduce the risk of experiencing high losses due to the uncertainty in the returns of the assets (this type of risk is also known as the "specific risk" or the "unsystematic risk" or the "diversifiable risk") and boost our performance. This is particularly true for the stock markets which are very volatile. Thus, the simple LADP method which uses only one asset per time period is more sensitive to big changes in the returns of the assets, while on the contrary the single-period, the LADP-UB, the PLADP and the equally-weighted methods that use more assets per time period exhibit a more stable out-of-sample behavior. Note that although multistage stochastic programming gives us diversified portfolios it performs very badly in our data since to use it we had to cut a significant amount of the input information.

5. Multi-period portfolio methods, except for multistage stochastic programming where to use it we had to reduce considerably the input information, perform better than the myopic single-period method. This is in line with many simulation studies which suggest that looking ahead in the future we are much better off in the decisions that we take now.

6. To our knowledge, in the literature piecewise linear approximate dynamic programming has been applied to "easy" problems in the sense that they have very few dimensions and discrete action/state/outcome space. This study proposes a simple update rule that keeps control over the number of the slopes in the piecewise linear value functions and we believe that this is of great general value as it opens the way to find "good" approximate solutions to other large-scale problems with multiple dimensions and continuous action/state/outcome space.

Overall, we believe that dynamic programming provides a good formulation for the portfolio selection problem and piecewise linear approximate dynamic programming is a good compromise.

9.2 Future Research

In this study, our main contribution has been to formulate the portfolio selection problem as a dynamic program and solve it using approximate dynamic programming methods. To evaluate these methods, we compared them in terms of their out-of-sample performance and computational effort against a number of other portfolio selection methods, myopic and multiperiod.

In the multistage stochastic programming method, when we wrote the deterministic equivalent problem in section 5.1.2 we explained that the way information is represented in multistage stochastic programs creates a staircase structure which allows us to decompose the original problem in a group of smaller problems, one for each node. In the literature, for multistage stochastic programs with staircase structures there exist decomposition algorithms that solve them optimally and they have the advantage of using much more information as opposed to the big linear deterministic problem. Among these algorithms, the most famous one is the *Nested Benders Decomposition Algorithm* (see for example [32] and [44]) which involves doing forward and backward passes on the scenario tree and solving the smaller subproblems of the nodes adding *feasibility* and *optimality cuts* if necessary. The advantage of using decomposition algorithms is that instead of solving one big problem which many times requires a significant reduction in the amount of the input information and takes a long time to solve, we solve a group of much smaller problems and this allows us to use more information. In our problem, we have not been able to implement Benders decomposition due to the CVaR constraints sharing a common variable, the one that gives us VaR, which does not allow us to decompose them on the nodes. To our knowledge, this problem has been solved using Benders decompositions only for two-stage stochastic programs and, as explained in section 5.1.1, it has been implemented in as the CVaRMin solver. The catch in this algorithm is that CVaR is decomposed into two stages and the common variable of VaR is treated as a first-stage decision. Given that CVaR is a very popular measure of risk not only in financial applications but also in other fields (see for example [27] for an application in electricity portfolio management), we believe that something similar can be done for multistage problems. We are currently working on

a modified version of Nested Benders Decomposition, where within the multistage framework we solve a two-stage problem that iteratively estimates CVaR in the end and the initial results for small-size trees look very promising.

Recent advances in the field of stochastic programming suggest, instead of using the traditional stochastic programming methods which are based on approximating the distribution of the uncertain parameters with scenario trees and are very time consuming, the use of a new class of tractable approximation techniques which preserve the input information but impose a linear structure on the recourse decisions of the multistage stochastic programs (see for example [49]). It would be interesting to apply these methods in our problem and compare with approximate dynamic programming methods.

Further, since we had to use an approximation for CVaR in the approximate dynamic programming methods, we feel that it would be very interesting to consider other types of utility functions in the objective such as for example a power utility function. This would allow us to indirectly include risk in the optimal dynamic programming formulation and thus would make the approximate dynamic programming methods more tractable.

Finally, it would be very interesting to see how the piecewise linear approximate dynamic programming methods presented in this study work in other large-scale problems with continuous action/state/outcome space.

Appendices

Appendix A

OGARCH in Scenario Generation

Let $\mathbf{R}_t$ be the $1 \times N$ time series vector of rates of returns of N securities at time t. According to the MGARCH method, the returns of the securities evolve as follows:

$$\mathbf{R}_t = \mu_t + \epsilon_t \tag{A.1}$$

where μ_t is the conditional mean vector and ϵ_t is the $1 \times N$ vector of residuals which are computed by:

$$\epsilon_t = \nu_t \boldsymbol{H}_t^{\frac{1}{2}} \tag{A.2}$$

In (A.2), $\boldsymbol{H}_t^{\frac{1}{2}}$ is the lower triangular matrix obtained by the cholesky factorization of the $N \times N$ covariance matrix $\boldsymbol{H}_t$ and ν_t is a $1 \times N$ vector of Gaussian white noise. Every variable in vector ν_t is an independent, identically distributed sequence that follows a normal distribution with zero mean and unit variance.

To implement the OGARCH method, we assume a constant conditional mean which we compute by taking averages of the historical rates of returns and compute the time varying covariance matrix $\boldsymbol{H}_t$ by applying principal component analysis to the normalized matrix of the historical returns.

Suppose that after applying the principal component analysis we extract m components, where $m \leq N$. If V is the matrix with elements $U_{ij}\sigma_i$, where U_j is the eigenvector of the j^{th} component and σ_i is the standard deviation of the i^{th} asset, and Λ_t is the diagonal matrix of the conditional variances of the components, then according to [1] the time varying conditional matrix $\boldsymbol{H}_t$ can be computed as follows:

$$\boldsymbol{H}_t = V \Lambda_t V^T \tag{A.3}$$

The time varying conditional variance of every component is modeled as a univariate GARCH(1,1) process (see [11]) as follows:

$$\left.\begin{aligned} \sigma_t^2 &= c + \phi\sigma_{t-1}^2 + \chi\epsilon_{t-1}^2 \\ \epsilon_t &= \sigma_t\xi_t \end{aligned}\right\} \tag{A.4}$$

where σ_t^2 is the variance of the component at time t, σ_{t-1}^2 is the lagged variance at time $t-1$, ϵ_{t-1}^2 is the lagged squared residual at time $t-1$, ξ_t is an independent, identically distributed sequence that follows a normal distribution with zero mean and unit variance, and parameters c, ϕ and χ are parameters estimated by *maximum likelihood*.

Note that in (A.3) the time varying covariance matrix $\boldsymbol{H}_t$ must be strictly positive definite, otherwise we might end up having a zero or negative portfolio variance. However, matrix $\boldsymbol{H}_t$ may not be strictly positive definite if the number of components is less than N. To avoid this, in this study we extract all N components.

To use the OGARCH method in scenario generation, we forecast the variance of each component using (A.4), where a random number generator is used to compute the residuals. Then, using the forecasted variances of the components we form the diagonal matrix Λ_t and compute the covariance matrix $\boldsymbol{H}_t$ using (A.3). After computing the covariances in $\boldsymbol{H}_t$, we use a random number generator and sample N values from the standard normal distribution to compute the residuals of (A.2). Finally, we generate a realization of vector $\mathbf{R}_t$ using (A.1). In the above manner, we can recursively generate as many realizations as we want, where each realization depends on the realized variances of the components of the previous realization. A collection of T realizations form one scenario path with length T periods and the above process is repeated many times until we obtain the desired number of scenario paths.

Appendix B

Scenario Reduction and Scenario Tree Construction

Let Ξ and Z be continuous distributions of two N-dimensional stochastic processes on a time horizon of t periods. Suppose distributions Ξ and Z are approximated respectively by S scenarios $\xi^i = \{\xi^i_\tau\}^t_{\tau=1}$ with probabilities p_i, such that $\sum_{i=1}^S p_i = 1$, and $\tilde{S}$ scenarios $\zeta^j = \{\zeta^j_\tau\}^t_{\tau=1}$ with probabilities q_j, such that $\sum_{j=1}^{\tilde{S}} q_j = 1$.

The distance between the two distributions Ξ and Z, which we denote with $D_K(\Xi, Z)$, in the Kantorovich sense is the optimal value of a linear transportation problem, where every scenario ξ^i can be understood as a "supply" node with supply p_i and every scenario ζ^j can be understood as a "demand" node with demand q_j. If ρ_{ij} is the probability mass "transferred" from node i to node j, then the problem of finding the distance between the two distributions Ξ and Z involves determining the minimum cost solution to satisfy supply and demand and is expressed as follows:

$$\left.\begin{aligned} D_K(\Xi, Z) = \min_{\rho_{ij}} \quad & \sum_{i=1}^{S}\sum_{j=1}^{\tilde{S}} \rho_{ij} c_t(\xi^i, \zeta^j) \\ \text{s.t.} \quad & \sum_{j=1}^{\tilde{S}} \rho_{ij} = p_i, \ \ \forall i \\ & \sum_{i=1}^{S} \rho_{ij} = q_j, \ \ \forall j \\ & \rho_{ij} \geq 0, \ \ \forall i, j \end{aligned}\right\} \tag{B.1}$$

where $c_t(\xi^i, \zeta^j) = \sum_{\tau=1}^t |\xi^i_\tau - \zeta^j_\tau|$ with $|.|$ denoting some norm on $\mathbb{R}^N$. The optimal value of problem (B.1) can be understood as the minimal cost we have to

pay in order to transform one distribution into the other.

Now suppose Z is the *reduced distribution* of Ξ, i.e. the set of scenarios in Z is a subset of the set of scenarios in Ξ. If D denotes some index set that contains the deleted scenarios, then from Theorem 2 in [26] the minimal distance between the two distributions can be computed explicitly using the following formula:

$$D_K(\Xi, Z) = \sum_{i \in D} p_i \min_{j \notin D} c_t(\xi^i, \xi^j), \tag{B.2}$$

and the probabilities of the preserved scenarios are given by the following rule:

$$q_j = p_j + \sum_{i \in D(j)} p_i \tag{B.3}$$

where $D(j) = \{i \in D : j = j(i)\}$ with $j(i) \in \arg\min_{j \notin D} c_t(\xi^i, \zeta^j)\ \forall i \in D$ and is known as the *optimal redistribution rule*. That is, the probability of every preserved scenario in the reduced distribution Z is updated with the probabilities of those deleted scenarios from distribution Ξ that are closest to it with respect to distance c_t.

Considering the above, to construct a scenario tree with T stages from a group of scenario paths, in every stage t for $t = 2, ..., T$ we apply the following *maximal reduction strategy*: We determine the optimal index set D, such that the Kantorovich distance (B.2) between the initial and the reduced distribution does not exceed a certain level of accuracy ϵ_t, i.e. $\sum_{i \in D} p_i \min_{j \notin D} c_t(\xi_i, \xi_j) \leq \epsilon_t$. The probabilities of the preserved scenarios in the reduced distribution are given by (B.3).

In the literature, there exist algorithms that apply the above reduction rule to construct scenario trees from an input of scenario paths in either a forward or a backward fashion in polynomial times (see for example the algorithms in [43]). The key idea in these algorithms is that at each stage t we crop from the scenario paths the information related to later stages and merge those current-stage nodes for which their respective t-stage scenario paths have minimal distance (B.2) updating at the same time the probability of the new node according to (B.3) and connecting with arcs the new node with all the children nodes of the merged nodes. In this manner, branching is created and a scenario tree comes out from the original distribution of scenario paths. To control the amount of information maintained in the reduced distribution, a *relative tolerance* indicator is defined as follows:

$$\epsilon_{\text{rel}} = \frac{\epsilon}{\epsilon_{\text{max}}} \tag{B.4}$$

where ϵ_{max} is the best possible Kantorovich distance between the distribution of the initial set of scenarios and the distribution of one of the scenarios with unit mass (i.e. with probability 1), ϵ is an upper bound on the cumulative relative tolerance $\sum_{t=2}^{T} \epsilon_t$, i.e. we have $\sum_{t=2}^{T} \epsilon_t \leq \epsilon$, and for some $q \in (0, 1)$ is split among individual tolerances ϵ_t according to the following recursive exponential rule:

$$\epsilon_t = \begin{cases} \epsilon(1-q), & t = T \\ q\epsilon_{t+1}, & t = T-1, \ldots, 2 \end{cases} \tag{B.5}$$

Numerical experience in [43] shows that a value closer to 1 would be more desirable as it leads to a higher number of leaf nodes (i.e. scenarios) and branching points.

The relative tolerance indicator can be understood as the proportion of the information we want to delete from the original distribution and takes values in $[0, 1]$. If $\epsilon_{\text{rel}} = 0$ the tree will not be reduced, while if $\epsilon_{\text{rel}} = 1$ the reduction will be maximal and only one scenario will be maintained.

Appendix C

Derivation of Gradients for the LADP Methods

Here, we derive the gradients for the LADP methods by substituting the optimal solutions of the corresponding subproblems of ADP in their objectives.

Gradients for LADP

After solving problem (6.6), we will have one of the following outcomes:

1. **Case 1**: Set Buy-assets is empty. The objective of (6.6) is given by:

$$\begin{aligned}\tilde{V}_{t-1}\left(\boldsymbol{h}_{t-1}^{+}\right) &= \sum_{i=1}^{N} k_{it}x_{it} + \sum_{i=1}^{N} l_{it}y_{it} + \sum_{i=0}^{N} u_{it}R_{it}h_{i(t-1)}^{+} \\ &= \sum_{i\in\mathcal{I}} l_{it}R_{it}h_{i(t-1)}^{+} + \sum_{i=0}^{N} u_{it}R_{it}h_{i(t-1)}^{+},\end{aligned}$$

   and the gradients are as follows:

   (a) For every asset $i \in \mathcal{I}$:

$$\Delta\tilde{V}_{i(t-1)} = \left(l_{it} + u_{it}\right)R_{it} = \left(1-\theta\right)u_{0t}R_{it}$$

   (b) For every other asset i including cash:

$$\Delta\tilde{V}_{i(t-1)} = u_{it}R_{it}$$

Note that any additional wealth in Sell-assets is converted to cash using transformation coefficient $1-\theta$ of Table 6.2, while for all other assets including cash any additional wealth remains with the asset.

2. **Case 2**: Set Buy-assets is non-empty, i.e. there exists an asset $j^\star$. The objective of (6.6) is given by:

$$\begin{aligned}\tilde{V}_{t-1}\left(\boldsymbol{h}^+_{t-1}\right) &= \sum_{i=1}^{N} k_{it}x_{it} + \sum_{i=1}^{N} l_{it}y_{it} + \sum_{i=0}^{N} u_{it}R_{it}h^+_{i(t-1)} \\ &= k_{j^\star t}\left(\frac{1}{1+\theta}R_{0t}h^+_{0(t-1)} + \frac{1-\theta}{1+\theta}\sum_{i\in\mathcal{I}\cup\mathcal{J}} R_{it}h^+_{i(t-1)}\right) \\ &+ \sum_{i\in\mathcal{I}\cup\mathcal{J}} l_{it}R_{it}h^+_{i(t-1)} + \sum_{i=0}^{N} u_{it}R_{it}h^+_{i(t-1)},\end{aligned}$$

and the gradients are as follows:

(a) For every asset $i \in \mathcal{I}\cup\mathcal{J}$:

$$\Delta\tilde{V}_{i(t-1)} = \left(\frac{1-\theta}{1+\theta}k_{j^\star t} + l_{it} + u_{it}\right)R_{it} = \frac{1-\theta}{1+\theta}u_{j^\star t}R_{it}$$

(b) For cash:

$$\Delta\tilde{V}_{0(t-1)} = \left(\frac{1}{1+\theta}k_{j^\star t} + u_{0t}\right)R_{0t} = \frac{1}{1+\theta}u_{j^\star t}R_{0t}$$

(c) For every other asset i:

$$\Delta\tilde{V}_{i(t-1)} = u_{it}R_{it}$$

Note that any additional wealth in Sell-assets and Sell-to-buy-assets is diverted to asset $j^\star$ using transformation coefficient $\frac{1-\theta}{1+\theta}$ of Table 6.2, while for all other assets any additional wealth remains with the asset.

Note that the gradients of the assets of sets $\mathcal{I}$ and $\mathcal{J}$ are also used to compute the observed slopes of those Sell-assets and Sell-to-buy-assets that previously had zero holdings and thus were not used in the solution of problem (6.6).

Gradients for LADP-UB

After solving problem (6.18) with Algorithm 6.2 in section 6.3, we provided a sketch of the optimal solution, which is given by (6.26) and (6.27) for Case 1, (6.28) and (6.29) for Sub-case 2.1 and (6.30) and (6.31) for Sub-case 2.2. Note that in the gradients that we compute below we further divide Case 1 into two new Sub-cases, one where buying did not occur because there were no Buy-assets with holdings less than w_0 (Sub-case 1.1) and one where buying did not occur because we had no resources (Sub-case 1.2). Substituting (6.26) and (6.27) in the objective of (6.18) gives us the gradients only for Sub-case 1.1. We will see later on that the gradients of the assets change if we start using a Buy-asset $j^\star$ after an incremental increase in our resources. Further, we divide Sub-case 2.1 into two new Sub-cases, one where we filled up all Buy-assets with cash resources and when we stopped buying we had no Sell-to-buy-assets (Sub-case 2.1.1) and one where we filled up all Buy-assets with cash resources and when we stopped buying we had Sell-to-buy-assets (Sub-case 2.1.2). Substituting (6.28) and (6.29) in the objective of (6.18) gives us the gradients only for Sub-case 2.1.2. We will see later on that the gradients of the assets change if we did not have a Sell-to-buy-asset d when we stopped buying.

Therefore, after solving problem (6.18) with Algorithm 6.2 and using $h_{it} = R_{it}h^+_{i(t-1)}$ we will have one of the following cases:

1. **Case 1**: No buying occurred, i.e. $\mathcal{C} = \emptyset$, because:

   (a) **Sub-case 1.1**: there were no Buy-assets with holdings less than w_0. Substituting (6.26) and (6.27) in the objective of (6.18), we get:

$$\begin{aligned}\tilde{V}_{t-1}\left(\boldsymbol{h}^+_{t-1}\right) &= \sum_{i=1}^{N} k_{it}x_{it} + \sum_{i=1}^{N} l_{it}y_{it} + \sum_{i=0}^{N} u_{it}R_{it}h^+_{i(t-1)} \\ &= \sum_{i\in\mathcal{I}} l_{it}R_{it}h^+_{i(t-1)} + \sum_{i\in\bar{\mathcal{F}}} l_{it}(R_{it}h^+_{i(t-1)} - w_0) \\ &+ \sum_{i=0}^{N} u_{it}R_{it}h^+_{i(t-1)},\end{aligned}$$

   and the gradients are as follows:

   i. For every asset $i \in \mathcal{I} \cup \bar{\mathcal{F}}$:

$$\Delta\tilde{V}_{i(t-1)} = (l_{it} + u_{it})\, R_{it} = (1-\theta) u_{0t} R_{it}$$

ii. For every other asset i including cash:

$$\Delta\tilde{V}_{i(t-1)} = u_{it} R_{it}$$

Note that any additional wealth in Sell-assets and Forced-to-sell-assets is converted to cash using transformation coefficient $1-\theta$ of Table 6.1. For all other assets, including cash, any additional wealth remains with the asset.

(b) **Sub-case 1.2**: there were no resources available. Note that in this case there exists a Buy-asset $j^{\star}$, which is the last Buy-asset c of set $\mathcal{C}$ and which we may use after increasing the holdings of the assets that were not previously in the optimal solution because they had zero holdings. Thus, using Buy-asset c in the objective of (6.18) by setting $x_{ct} = \frac{1}{1+\theta} R_{0t} h^{+}_{0(t-1)} + \frac{1-\theta}{1+\theta} \sum_{i \in \mathcal{I} \cup \bar{\mathcal{F}} \cup \mathcal{D}} R_{it} h^{+}_{i(t-1)}$ along with (6.26) and (6.27), we get:

$$\begin{aligned}
\tilde{V}_{t-1}\left(\boldsymbol{h}^{+}_{t-1}\right) &= \sum_{i=1}^{N} k_{it} x_{it} + \sum_{i=1}^{N} l_{it} y_{it} + \sum_{i=0}^{N} u_{it} R_{it} h^{+}_{i(t-1)} \\
&= k_{ct} \left(\frac{1}{1+\theta} R_{0t} h^{+}_{0(t-1)} + \frac{1-\theta}{1+\theta} \sum_{i \in \mathcal{I} \cup \bar{\mathcal{F}} \cup \mathcal{D}} R_{it} h^{+}_{i(t-1)} \right) \\
&+ \sum_{i \in \mathcal{I} \cup \mathcal{D}} l_{it} R_{it} h^{+}_{i(t-1)} + \sum_{i \in \bar{\mathcal{F}}} \left(l_{it} R_{it} h^{+}_{i(t-1)} - w_0 \right) \\
&+ \sum_{i=0}^{N} u_{it} R_{it} h^{+}_{i(t-1)}
\end{aligned}$$

and the gradients are as follows:

i. For every asset $i \in \mathcal{I} \cup \bar{\mathcal{F}} \cup \mathcal{D}$:

$$\Delta\tilde{V}_{i(t-1)} = \left(\frac{1-\theta}{1+\theta} k_{ct} + l_{it} + u_{it} \right) R_{it} = \frac{1-\theta}{1+\theta} u_{ct} R_{it}$$

ii. For cash:

$$\Delta\tilde{V}_{0(t-1)} = \left(\frac{1}{1+\theta} k_{ct} + u_{0t} \right) R_{0t} = \frac{1}{1+\theta} u_{ct} R_{0t}$$

iii. For every other asset i:

$$\Delta\tilde{V}_{i(t-1)} = u_{it}R_{it}$$

Note that any additional wealth in cash and other assets that we liquidate (i.e. Sell-assets, Forced-to-sell-assets and Sell-to-buy-assets) is diverted to asset $j^\star$ using respectively transformation coefficients $\frac{1}{1+\theta}$ and $\frac{1-\theta}{1+\theta}$ of Table 6.1, while in all other assets any additional wealth remains with the asset.

2. **Case 2**: Buying occurred, i.e. $\mathcal{C} \neq \emptyset$, but we stopped buying because:

   (a) **Sub-case 2.1**: we filled up all Buy-assets in set $\mathcal{C}$ to level w_0 with cash resources and when we stopped buying:

      i. **Sub-case 2.1.1**: we had no Sell-to-buy-assets. Substituting (6.28) and (6.29) in the objective of (6.18) with $\mathcal{D} = \emptyset$, we get:

$$\begin{aligned}\tilde{V}_{t-1}\left(\boldsymbol{h}^+_{t-1}\right) &= \sum_{i=1}^{N} k_{it}x_{it} + \sum_{i=1}^{N} l_{it}y_{it} + \sum_{i=0}^{N} u_{it}R_{it}h^+_{i(t-1)} \\ &= \sum_{i\in\mathcal{C}} k_{it}\left(w_0 - R_{it}h^+_{i(t-1)}\right) + \sum_{i\in\mathcal{I}} l_{it}R_{it}h^+_{i(t-1)} \\ &+ \sum_{i\in\bar{\mathcal{F}}} l_{it}(R_{it}h^+_{i(t-1)} - w_0) + \sum_{i=0}^{N} u_{it}R_{it}h^+_{i(t-1)}\end{aligned}$$

      and the gradients are as follows:

      A. For every asset $i \in \mathcal{I} \cup \bar{\mathcal{F}}$:

$$\Delta\tilde{V}_{i(t-1)} = (l_{it} + u_{it})\,R_{it} = (1-\theta)u_{0t}R_{it}$$

      B. For every asset $i \in \mathcal{C}$:

$$\Delta\tilde{V}_{i(t-1)} = (-k_{it} + u_{it})\,R_{it} = (1+\theta)u_{0t}R_{it}$$

      C. For every other asset i including cash:

$$\Delta\tilde{V}_{i(t-1)} = u_{it}R_{it}$$

Note that any additional wealth in Sell-assets or Forced-to-sell-assets is converted to cash using transformation coefficient $1-\theta$ of Table 6.1, while any additional wealth in one of the Buy-assets of set $\mathcal{C}$ decreases the amount of cash used for buying through projection coefficient $1+\theta$ of Table 6.2. For all other assets, including cash, any additional wealth remains with the asset.

ii. **Sub-case 2.1.2**: we had Sell-to-buy-assets. Note that in this case we would continue selling the last Sell-to-buy-asset, which is asset d, if there existed another Buy-asset $j^{\star}$. Substituting (6.28) and (6.29) in the objective of (6.18), we get:

$$\begin{aligned}
\tilde{V}_{t-1}\left(\boldsymbol{h}^{+}_{t-1}\right) &= \sum_{i=1}^{N} k_{it}x_{it} + \sum_{i=1}^{N} l_{it}y_{it} + \sum_{i=0}^{N} u_{it}R_{it}h^{+}_{i(t-1)} \\
&= \sum_{i\in\mathcal{C}} k_{it}\left(w_0 - R_{it}h^{+}_{i(t-1)}\right) + \sum_{i\in(\mathcal{I}\cup\mathcal{D})\setminus\{d\}} l_{it}R_{it}h^{+}_{i(t-1)} \\
&+ l_{dt}\Big[\frac{1+\theta}{1-\theta}\sum_{i\in\mathcal{C}}\left(w_0 - R_{it}h^{+}_{i(t-1)}\right) - \frac{1}{1-\theta}R_{0t}h^{+}_{0(t-1)} \\
&- \sum_{i\in(\mathcal{I}\cup\mathcal{D})\setminus\{d\}} R_{it}h^{+}_{i(t-1)} - \sum_{i\in\bar{\mathcal{F}}}(R_{it}h^{+}_{i(t-1)} - w_0)\Big] \\
&+ \sum_{i\in\bar{\mathcal{F}}} l_{it}(R_{it}h^{+}_{i(t-1)} - w_0) + \sum_{i=0}^{N} u_{it}R_{it}h^{+}_{i(t-1)}
\end{aligned}$$

and the gradients are as follows:

A. For every asset $i \in \left(\mathcal{I}\cup\bar{\mathcal{F}}\cup\mathcal{D}\right)\setminus\{d\}$:

$$\Delta\tilde{V}_{i(t-1)} = \left(l_{it} - l_{dt} + u_{it}\right)R_{it} = u_{dt}R_{it}$$

B. For every asset $i \in \mathcal{C}$:

$$\Delta\tilde{V}_{i(t-1)} = \left(-k_{it} - \frac{1+\theta}{1-\theta}l_{dt} + u_{it}\right)R_{it} = \frac{1+\theta}{1-\theta}u_{dt}R_{it}$$

C. For cash:

$$\Delta\tilde{V}_{0(t-1)} = \left(-\frac{1}{1-\theta}l_{dt} + u_{0t}\right)R_{0t} = \frac{1}{1-\theta}u_{dt}R_{0t}$$

D. For every other asset i including asset d:

$$\Delta\tilde{V}_{i(t-1)} = u_{it}R_{it}$$

Note that any additional wealth in Sell-assets, Forced-to-sell-assets and Sell-to-buy-assets, except for asset d, decrease the wealth of asset d sold by the same amount since these assets will be sold before asset d. Further, any additional wealth in one of the Buy-assets of set $\mathcal{C}$ or any additional wealth in cash decreases the amount of the last Sell-to-buy-asset d sold through projection coefficients $\frac{1+\theta}{1-\theta}$ and $\frac{1}{1-\theta}$ of Table 6.2 respectively. In all other assets, including asset d, any additional wealth remains with the asset.

(b) **Sub-case 2.2**: we run out of resources. Note that in this case we would continue buying the last Buy-asset c of set $\mathcal{C}$ if we had more resources. Substituting (6.30) and (6.31) in the objective of (6.18), we get:

$$\begin{aligned}
\tilde{V}_{t-1}\left(\boldsymbol{h}^{+}_{t-1}\right) &= \sum_{i=1}^{N} k_{it}x_{it} + \sum_{i=1}^{N} l_{it}y_{it} + \sum_{i=0}^{N} u_{it}R_{it}h^{+}_{i(t-1)} \\
&= \sum_{i\in\mathcal{C}\setminus\{c\}} k_{it}\left(w_0 - R_{it}h^{+}_{i(t-1)}\right) + k_{ct}\Big[\frac{1}{1+\theta}R_{0t}h^{+}_{0(t-1)} \\
&+ \frac{1-\theta}{1+\theta}\sum_{i\in\mathcal{I}\cup\mathcal{D}} R_{it}h^{+}_{i(t-1)} + \frac{1-\theta}{1+\theta}\sum_{i\in\bar{\mathcal{F}}}\left(R_{it}h^{+}_{i(t-1)} - w_0\right) \\
&- \sum_{i\in\mathcal{C}\setminus\{c\}}\left(w_0 - R_{it}h^{+}_{i(t-1)}\right)\Big] + \sum_{i\in\mathcal{I}\cup\mathcal{D}} l_{it}R_{it}h^{+}_{i(t-1)} \\
&+ \sum_{i\in\bar{\mathcal{F}}} l_{it}\left(R_{it}h^{+}_{i(t-1)} - w_0\right) + \sum_{i=0}^{N} u_{it}R_{it}h^{+}_{i(t-1)}
\end{aligned}$$

and the gradients are as follows:

i. For every asset $i \in \mathcal{I}\cup\bar{\mathcal{F}}\cup\mathcal{D}$:

$$\Delta\tilde{V}_{i(t-1)} = \left(\frac{1-\theta}{1+\theta}k_{ct} + l_{it} + u_{it}\right)R_{it} = \frac{1-\theta}{1+\theta}u_{ct}R_{it}$$

ii. For every asset $i \in \mathcal{C}\setminus\{c\}$:

$$\Delta\tilde{V}_{i(t-1)} = \left(-k_{it} + k_{ct} + u_{it}\right)R_{it} = u_{ct}R_{it}$$

iii. For cash:

$$\Delta\tilde{V}_{0(t-1)} = \left(\frac{1}{1+\theta}k_{ct} + u_{0t}\right) R_{0t} = \frac{1}{1+\theta}u_{ct}R_{0t}$$

iv. For every other asset i including asset c:

$$\Delta\tilde{V}_{i(t-1)} = u_{it}R_{it}$$

Note that any additional wealth in cash and other assets that we liquidate (i.e. Sell-assets, Forced-to-sell-assets, or Sell-to-buy-assets) is diverted to the last Buy-asset c of set $\mathcal{C}$ using respectively transformation coefficients $\frac{1}{1+\theta}$ and $\frac{1-\theta}{1+\theta}$ of Table 6.1, while in all other assets any additional wealth remains with the asset.

Appendix D

Experimental Results: Scenario Reduction Parameters

Solver Scenred2 of GAMS requires the creation of an input file that contains the initial scenario paths as well as a number of scalar parameters that control both scenario reduction and scenario tree construction (a user manual for solver Scenred2 can be downloaded from `http://www.gams.com/solvers/`).

For our simulations, we constructed scenario trees in a backward fashion, i.e. starting each time from the end and moving backward reducing the information as we go, using a recursive exponential rule with $q = 0.85$ (see Appendix B for a definition of the exponential rule) and relative tolerances (this is the amount of information deleted from the initial distribution and is denoted with ϵ_{rel}) as summarized in Tables D.1-D.4, where:

- "# stages" denotes the number of stages,
- "# scenarios" denotes the number of scenarios,
- "# nodes" denotes the number of nodes,
- "ϵ_{rel}" denotes the relative tolerance, and
- "reduction time (in sec)" denotes the reduction time in seconds.

Note that the relative tolerances were selected in such a way so that on the one hand the reduced distribution contains as many nodes as possible, but on the other hand the deterministic equivalent of the associated multistage stochastic program remains computationally tractable.

# stages	ϵ_{rel}	**Initial distribution**		**Reduced distribution**		**reduction time (in sec)**
		# scenarios	**# nodes**	**# scenarios**	**# nodes**	
53	0.90	4000	208001	3339	20406	1469
52	0.89	4000	204001	3346	20406	1469
51	0.88	4000	200001	3353	21364	1444
50	0.88	4000	196001	3354	21149	1494
49	0.88	4000	192001	3354	20915	1581
48	0.88	4000	188001	3353	20677	1409
47	0.87	4000	184001	3361	20886	1287
46	0.87	4000	180001	3360	20535	1206
45	0.87	4000	176001	3359	20333	1170
44	0.86	4000	172001	3367	20625	1261
43	0.86	4000	168001	3367	20482	1140
42	0.85	4000	164001	3374	20564	1164
41	0.84	4000	160001	3381	21287	916
40	0.82	4000	156001	3394	22284	898
39	0.82	4000	152001	3393	22006	849
38	0.82	4000	148001	3392	21593	861
37	0.82	4000	144001	3391	21387	731
36	0.82	4000	140001	3389	20864	708
35	0.82	4000	136001	3390	20438	660
34	0.81	4000	132001	3394	20556	672
33	0.80	4000	128001	3400	20736	746
32	0.79	4000	124001	3405	20889	683
31	0.78	4000	120001	3411	20699	653
30	0.77	4000	116001	3418	21116	651
29	0.77	4000	112001	3418	20544	629
28	0.76	4000	108001	3425	21051	629
27	0.76	4000	104001	3421	20270	552
26	0.75	4000	100001	3426	20097	536
25	0.73	4000	96001	3441	20686	509
24	0.71	4000	92001	3452	20990	444
23	0.69	4000	88001	3463	21000	430
22	0.67	4000	84001	3475	21380	406
21	0.65	4000	80001	3486	21470	400
20	0.64	4000	76001	3490	21090	371
19	0.63	4000	72001	3492	20764	349
18	0.62	4000	68001	3493	20286	344
17	0.60	4000	64001	3504	20083	332
16	0.56	4000	60001	3528	20687	311
15	0.51	4000	56001	3561	21638	284
14	0.46	4000	52001	3593	22332	266
13	0.42	4000	48001	3619	22536	215
12	0.41	4000	44001	3615	21095	194
11	0.37	4000	40001	3637	20716	199
10	0.32	4000	36001	3669	20571	184
9	0.26	4000	32001	3713	20464	164
8	0.18	4000	28001	3786	20867	155
7	0.08	4000	24001	3895	21184	124
6	0	4000	20001	4000	20001	0
5	0	4000	16001	4000	16001	0
4	0	4000	12001	4000	12001	0
3	0	4000	8001	4000	8001	0
2	0	4000	4001	4000	4001	0

Table D.1: Scenario reduction parameters in the Up-Up dataset

# stages	ϵ_{rel}	**Initial distribution**		**Reduced distribution**		**reduction time (in sec)**
		# scenarios	**# nodes**	**# scenarios**	**# nodes**	
53	0.90	4000	208001	3339	20881	1767
52	0.89	4000	204001	3345	21043	1684
51	0.88	4000	200001	3353	21445	1653
50	0.88	4000	196001	3353	20999	1690
49	0.88	4000	192001	3353	20888	2163
48	0.88	4000	188001	3352	20746	1565
47	0.87	4000	184001	3359	20779	1525
46	0.87	4000	180001	3359	20460	1500
45	0.87	4000	176001	3358	20621	1445
44	0.86	4000	172001	3365	20766	1702
43	0.86	4000	168001	3363	20425	1379
42	0.85	4000	164001	3371	20659	1337
41	0.84	4000	160001	3380	21169	1076
40	0.82	4000	156001	3394	22241	1022
39	0.82	4000	152001	3393	21704	1017
38	0.82	4000	148001	3392	21528	1014
37	0.82	4000	144001	3392	21229	862
36	0.82	4000	140001	3391	20733	827
35	0.82	4000	136001	3389	20233	791
34	0.81	4000	132001	3395	20420	812
33	0.80	4000	128001	3401	20809	985
32	0.79	4000	124001	3408	20865	844
31	0.78	4000	120001	3415	20859	814
30	0.77	4000	116001	3423	21157	788
29	0.77	4000	112001	3420	20612	761
28	0.76	4000	108001	3425	20261	731
27	0.75	4000	104001	3430	20332	694
26	0.74	4000	100001	3435	20283	667
25	0.72	4000	96001	3449	20587	623
24	0.70	4000	92001	3461	21246	606
23	0.68	4000	88001	3474	21312	565
22	0.66	4000	84001	3486	21708	548
21	0.64	4000	80001	3501	22185	518
20	0.63	4000	76001	3505	21864	492
19	0.63	4000	72001	3497	20826	462
18	0.62	4000	68001	3500	20355	440
17	0.60	4000	64001	3506	20103	393
16	0.56	4000	60001	3530	20851	357
15	0.51	4000	56001	3564	21721	338
14	0.46	4000	52001	3596	22462	290
13	0.44	4000	48001	3605	21903	261
12	0.43	4000	44001	3600	21456	251
11	0.39	4000	40001	3624	20225	213
10	0.32	4000	36001	3674	20778	190
9	0.26	4000	32001	3717	20621	176
8	0.18	4000	28001	3787	20881	162
7	0.08	4000	24001	3896	21212	125
6	0	4000	20001	4000	20001	0
5	0	4000	16001	4000	16001	0
4	0	4000	12001	4000	12001	0
3	0	4000	8001	4000	8001	0
2	0	4000	4001	4000	4001	0

Table D.2: Scenario reduction parameters in the Up-Down dataset

# stages	ϵ_{rel}	**Initial distribution**		**Reduced distribution**		**reduction time (in sec)**
		# scenarios	**# nodes**	**# scenarios**	**# nodes**	
53	0.88	4000	208001	3351	20585	1762
52	0.87	4000	204001	3358	20789	1660
51	0.86	4000	200001	3366	21227	1640
50	0.86	4000	196001	3366	20973	1692
49	0.86	4000	192001	3365	21176	2178
48	0.86	4000	188001	3364	20948	1570
47	0.85	4000	184001	3370	21254	1517
46	0.85	4000	180001	3368	20988	1489
45	0.85	4000	176001	3367	20684	1435
44	0.84	4000	172001	3375	21397	1427
43	0.84	4000	168001	3374	21068	1065
42	0.83	4000	164001	3381	21476	1031
41	0.82	4000	160001	3388	21681	1056
40	0.82	4000	156001	3389	21778	1025
39	0.82	4000	152001	3387	21216	1018
38	0.82	4000	148001	3387	20937	991
37	0.82	4000	144001	3386	20938	865
36	0.81	4000	140001	3392	21052	816
35	0.81	4000	136001	3392	20960	784
34	0.79	4000	132001	3405	21499	798
33	0.78	4000	128001	3412	21537	880
32	0.77	4000	124001	3419	21659	886
31	0.76	4000	120001	3426	22005	815
30	0.76	4000	116001	3429	21944	783
29	0.76	4000	112001	3427	21465	759
28	0.75	4000	108001	3433	21112	743
27	0.74	4000	104001	3437	21143	698
26	0.73	4000	100001	3442	21244	667
25	0.71	4000	96001	3454	21810	622
24	0.70	4000	92001	3458	21538	610
23	0.70	4000	88001	3456	21083	569
22	0.69	4000	84001	3459	20588	547
21	0.67	4000	80001	3471	20900	516
20	0.66	4000	76001	3476	20685	499
19	0.64	4000	72001	3484	20441	461
18	0.61	4000	68001	3502	20793	429
17	0.59	4000	64001	3510	20572	392
16	0.55	4000	60001	3534	21104	355
15	0.50	4000	56001	3567	21960	322
14	0.47	4000	52001	3583	21810	286
13	0.45	4000	48001	3589	21076	264
12	0.42	4000	44001	3603	20632	229
11	0.38	4000	40001	3626	20289	213
10	0.31	4000	36001	3679	20932	191
9	0.25	4000	32001	3722	20780	178
8	0.17	4000	28001	3795	21119	161
7	0.07	4000	24001	3906	21460	124
6	0	4000	20001	4000	20001	0
5	0	4000	16001	4000	16001	0
4	0	4000	12001	4000	12001	0
3	0	4000	8001	4000	8001	0
2	0	4000	4001	4000	4001	0

Table D.3: Scenario reduction parameters in the Down-Up dataset

# stages	ϵ_{rel}	**Initial distribution**		**Reduced distribution**		**reduction time (in sec)**
		# scenarios	**# nodes**	**# scenarios**	**# nodes**	
53	0.90	4000	208001	3339	21297	1762
52	0.89	4000	204001	3346	21407	1664
51	0.88	4000	200001	3353	21287	1650
50	0.88	4000	196001	3353	21065	1897
49	0.88	4000	192001	3353	21287	1810
48	0.88	4000	188001	3353	20920	1581
47	0.87	4000	184001	3360	21521	1529
46	0.87	4000	180001	3359	21167	1472
45	0.87	4000	176001	3358	20772	1793
44	0.86	4000	172001	3365	21118	1489
43	0.86	4000	168001	3365	20843	1377
42	0.85	4000	164001	3371	20979	1344
41	0.84	4000	160001	3378	21339	1074
40	0.82	4000	156001	3392	22177	1023
39	0.82	4000	152001	3392	22158	1054
38	0.82	4000	148001	3391	21573	1007
37	0.82	4000	144001	3390	21497	875
36	0.82	4000	140001	3390	20861	841
35	0.82	4000	136001	3389	20675	797
34	0.81	4000	132001	3395	20787	952
33	0.80	4000	128001	3401	20890	1042
32	0.79	4000	124001	3408	21194	927
31	0.78	4000	120001	3415	21206	941
30	0.77	4000	116001	3420	21077	936
29	0.77	4000	112001	3419	20731	914
28	0.76	4000	108001	3426	20882	729
27	0.75	4000	104001	3431	20702	697
26	0.74	4000	100001	3436	20572	670
25	0.72	4000	96001	3450	21011	624
24	0.70	4000	92001	3462	21413	612
23	0.68	4000	88001	3475	21639	572
22	0.66	4000	84001	3487	21885	545
21	0.64	4000	80001	3499	22135	516
20	0.64	4000	76001	3494	21368	495
19	0.64	4000	72001	3489	20507	462
18	0.63	4000	68001	3492	20042	429
17	0.60	4000	64001	3509	20451	393
16	0.56	4000	60001	3533	20960	358
15	0.51	4000	56001	3565	21848	327
14	0.48	4000	52001	3581	21713	288
13	0.46	4000	48001	3586	20914	265
12	0.42	4000	44001	3610	20893	239
11	0.38	4000	40001	3632	20525	221
10	0.31	4000	36001	3686	21219	192
9	0.25	4000	32001	3730	21150	176
8	0.17	4000	28001	3801	21378	162
7	0.07	4000	24001	3896	21237	127
6	0	4000	20001	4000	20001	0
5	0	4000	16001	4000	16001	0
4	0	4000	12001	4000	12001	0
3	0	4000	8001	4000	8001	0
2	0	4000	4001	4000	4001	0

Table D.4: Scenario reduction parameters in the Down-Down dataset

Appendix E

Experimental Results: Cumulative Wealth Plots

Up-Up Plots

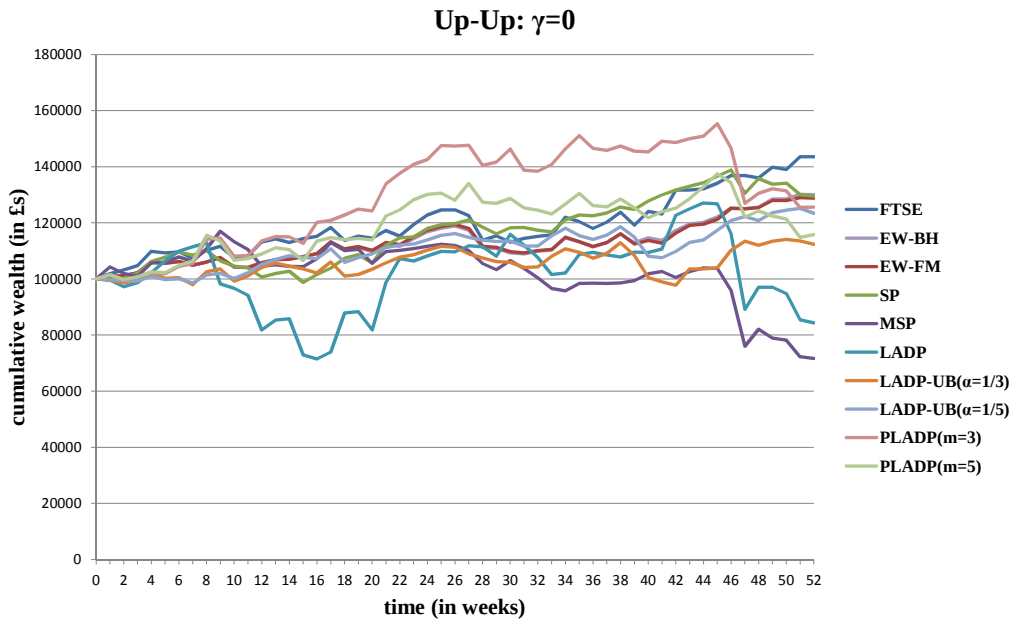


Figure E.1: Out-of-sample cumulative wealth against time: Up-Up, $\gamma = 0$

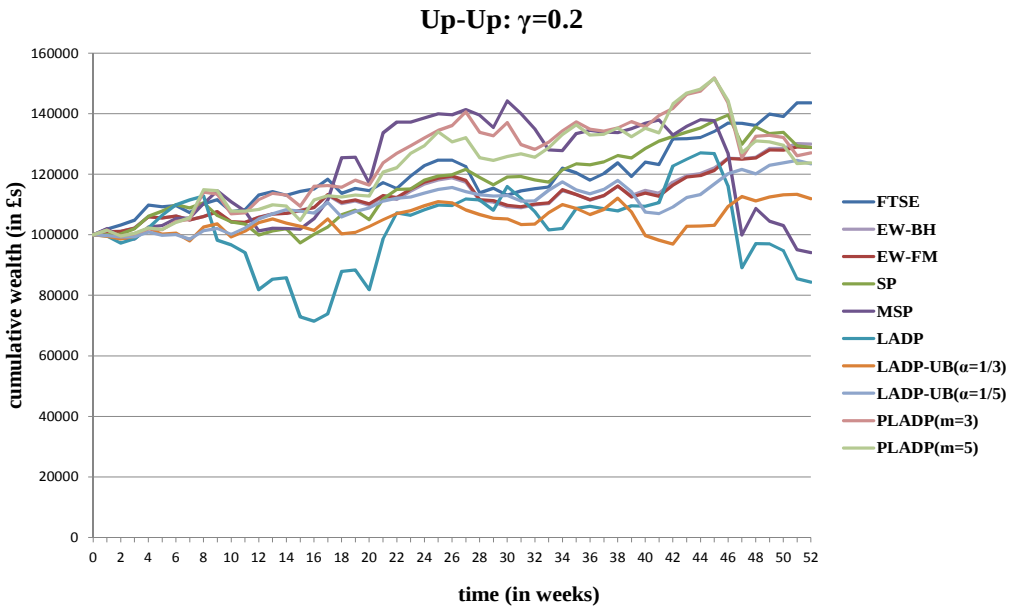


Figure E.2: Out-of-sample cumulative wealth against time: Up-Up, $\gamma = 0.2$

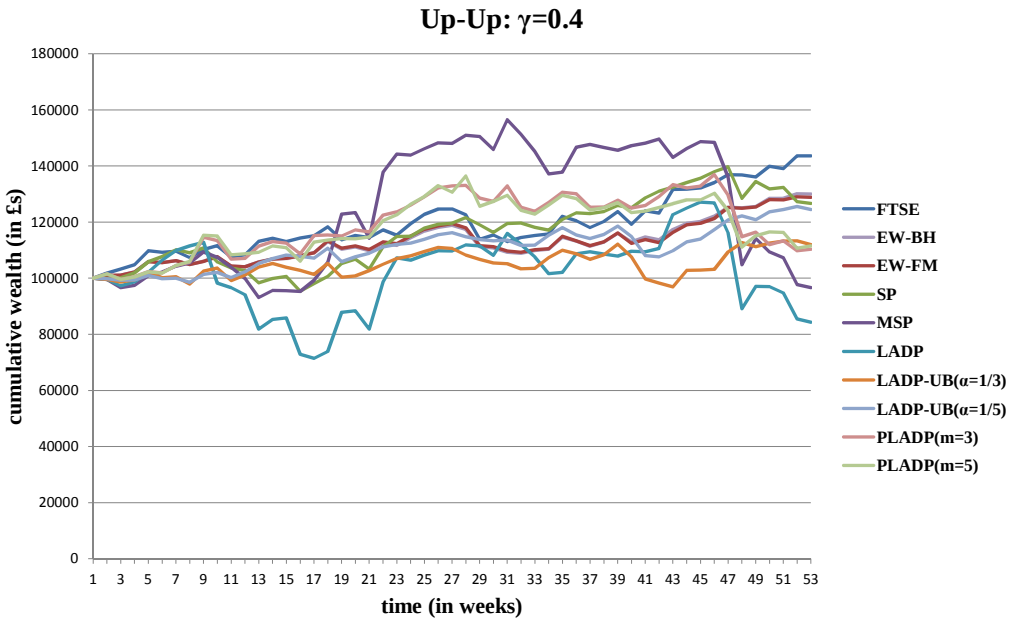


Figure E.3: Out-of-sample cumulative wealth against time: Up-Up, $\gamma = 0.4$

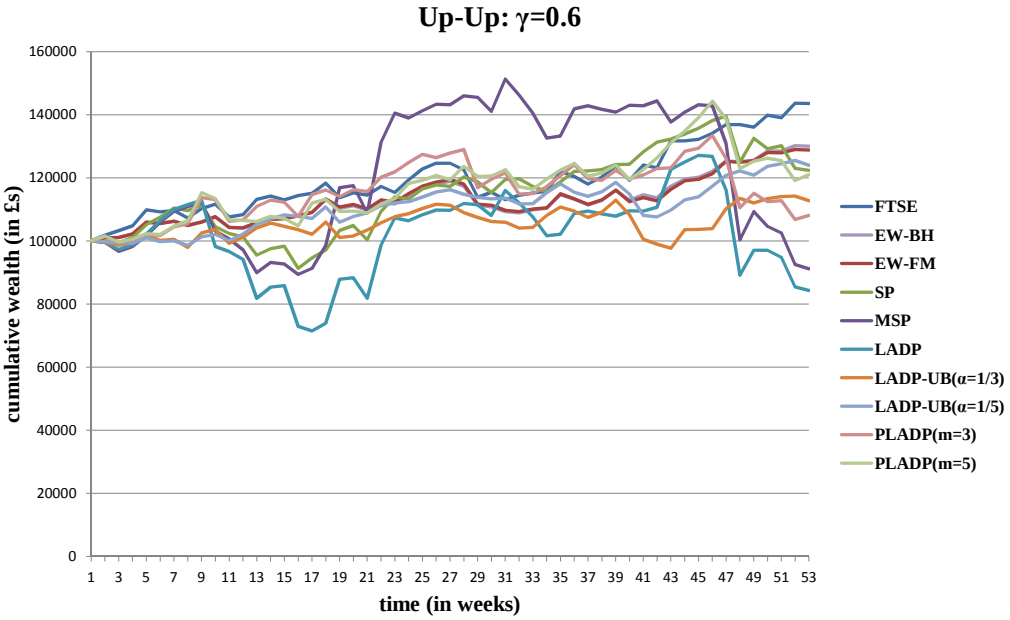


Figure E.4: Out-of-sample cumulative wealth against time: Up-Up, $\gamma = 0.6$

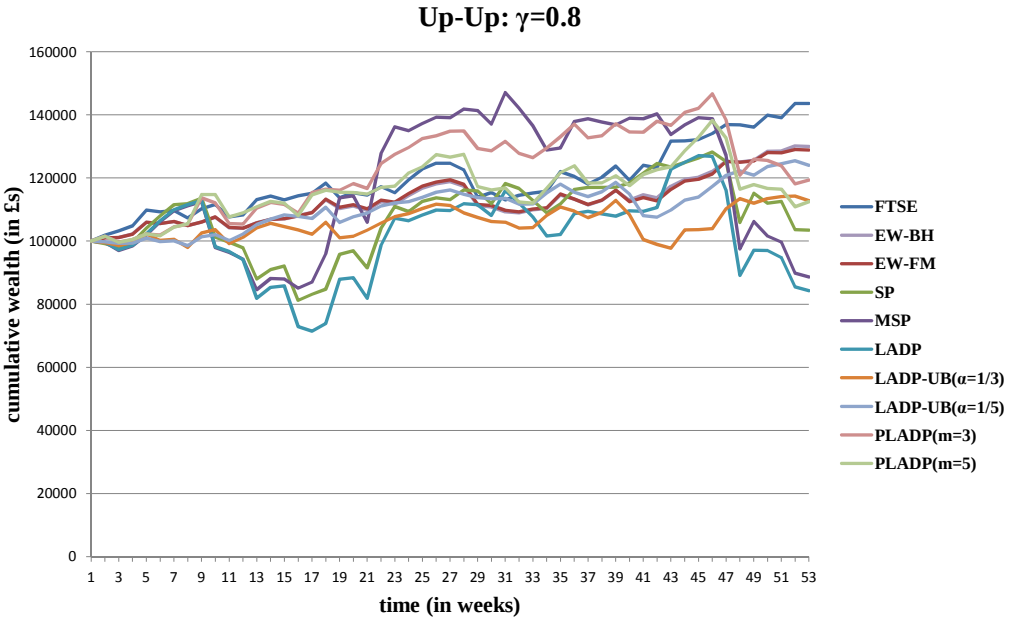


Figure E.5: Out-of-sample cumulative wealth against time: Up-Up, $\gamma = 0.8$

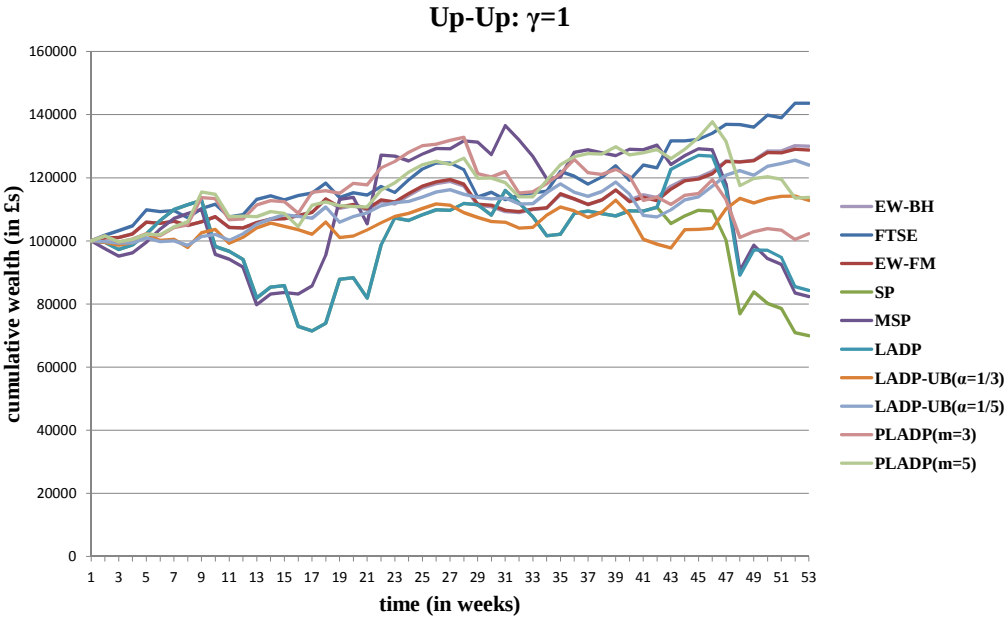


Figure E.6: Out-of-sample cumulative wealth against time: Up-Up, $\gamma = 1$

Up-Down Plots

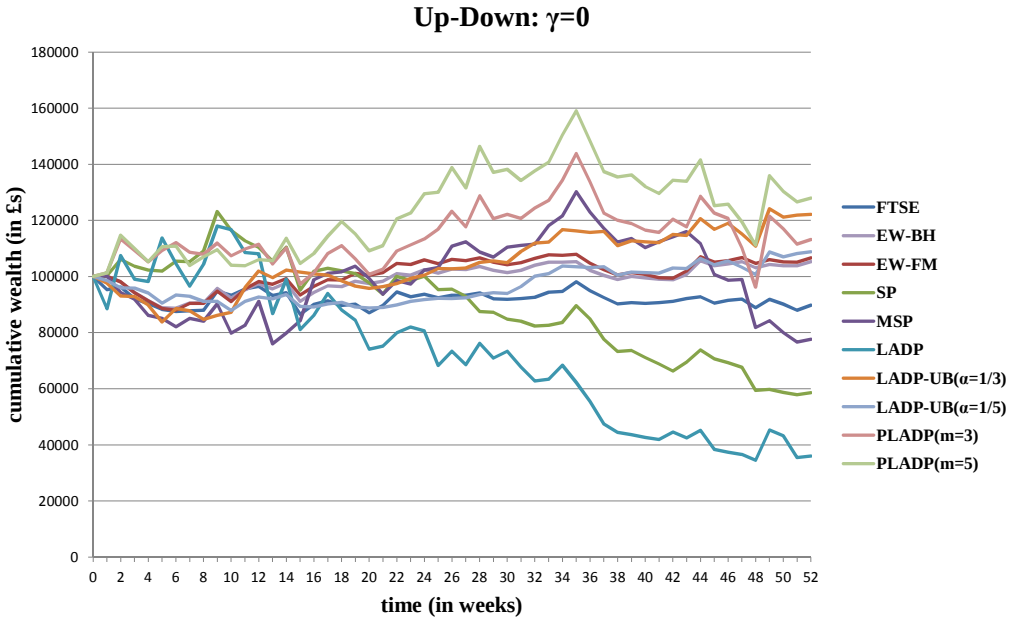


Figure E.7: Out-of-sample cumulative wealth against time: Up-Down, $\gamma = 0$

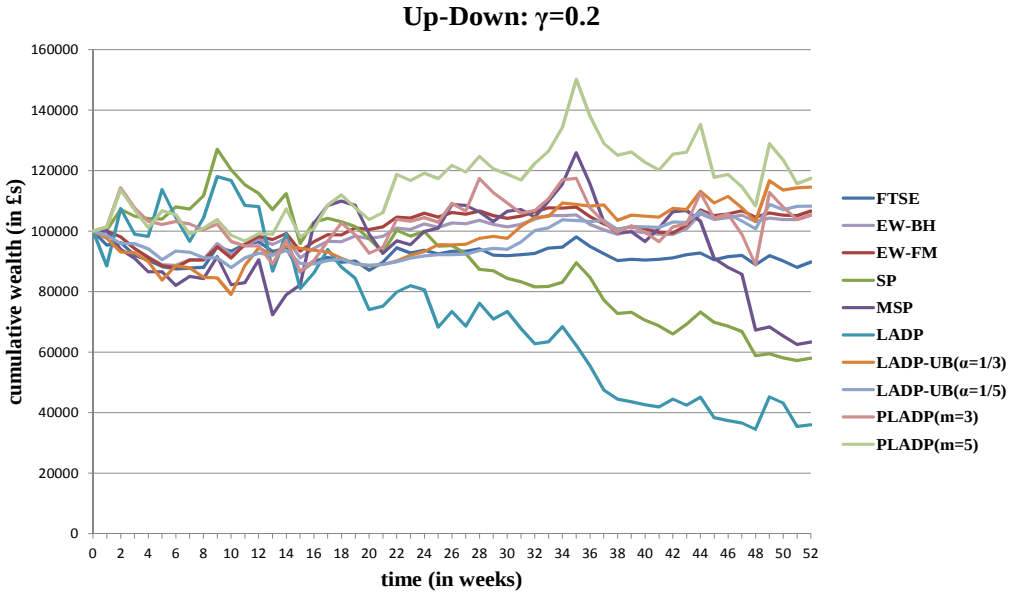


Figure E.8: Out-of-sample cumulative wealth against time: Up-Down, $\gamma = 0.2$

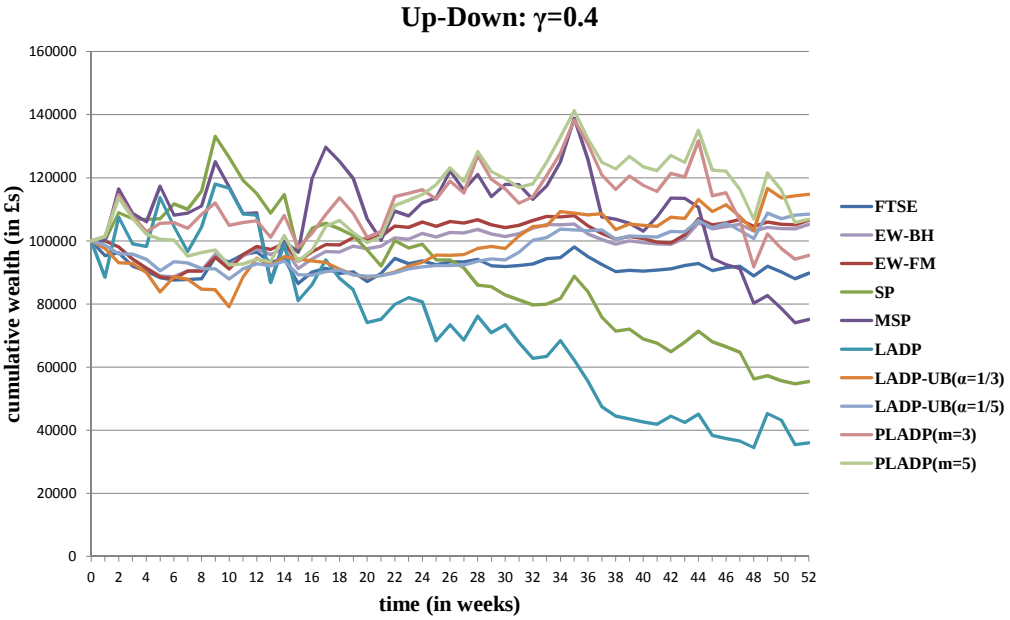


Figure E.9: Out-of-sample cumulative wealth against time: Up-Down, $\gamma = 0.4$

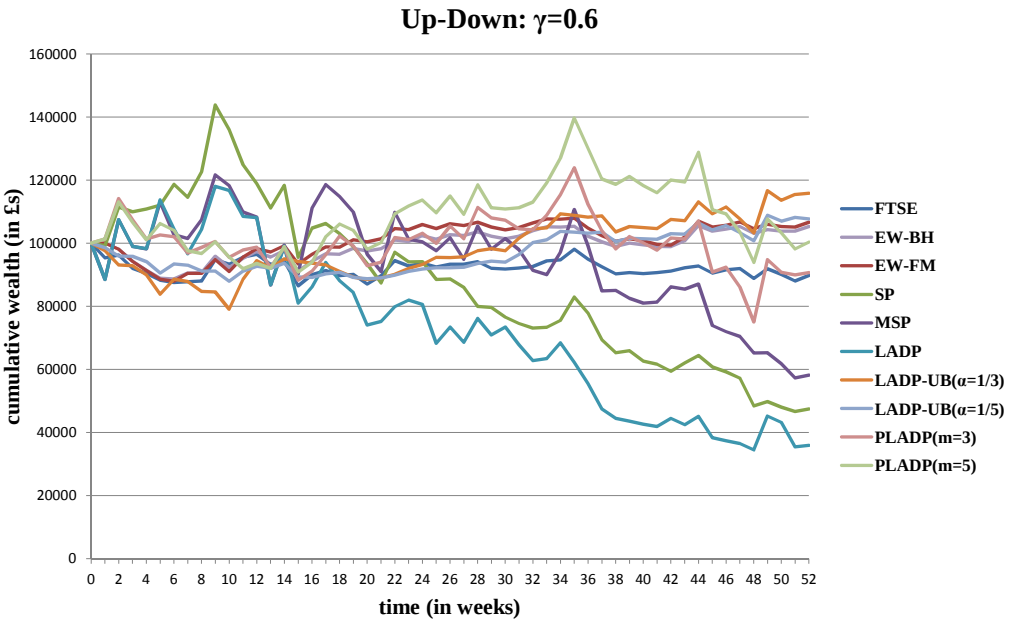


Figure E.10: Out-of-sample cumulative wealth against time: Up-Down, $\gamma = 0.6$

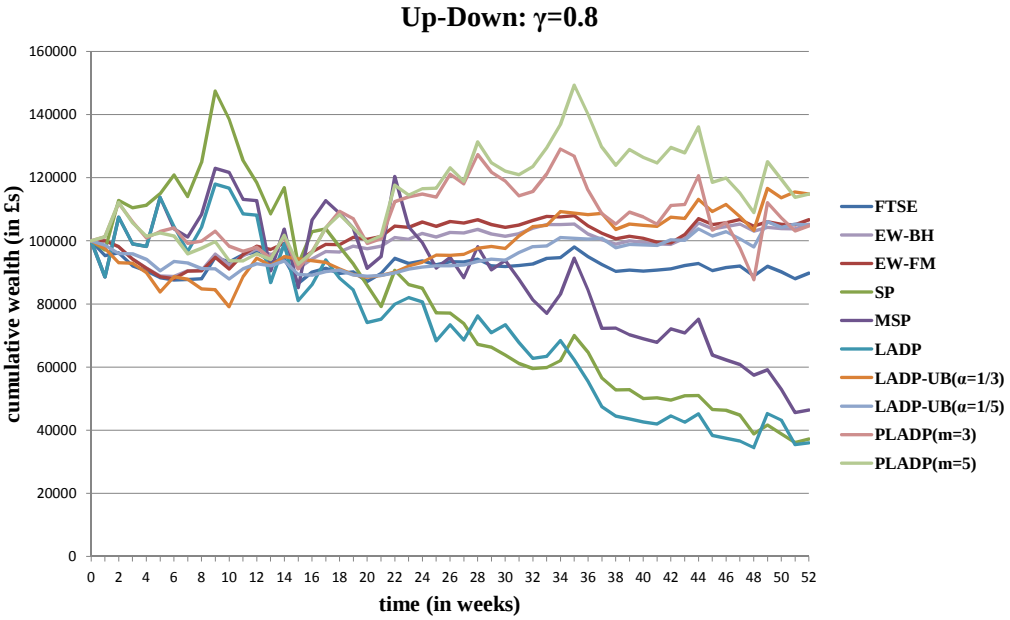


Figure E.11: Out-of-sample cumulative wealth against time: Up-Down, $\gamma = 0.8$

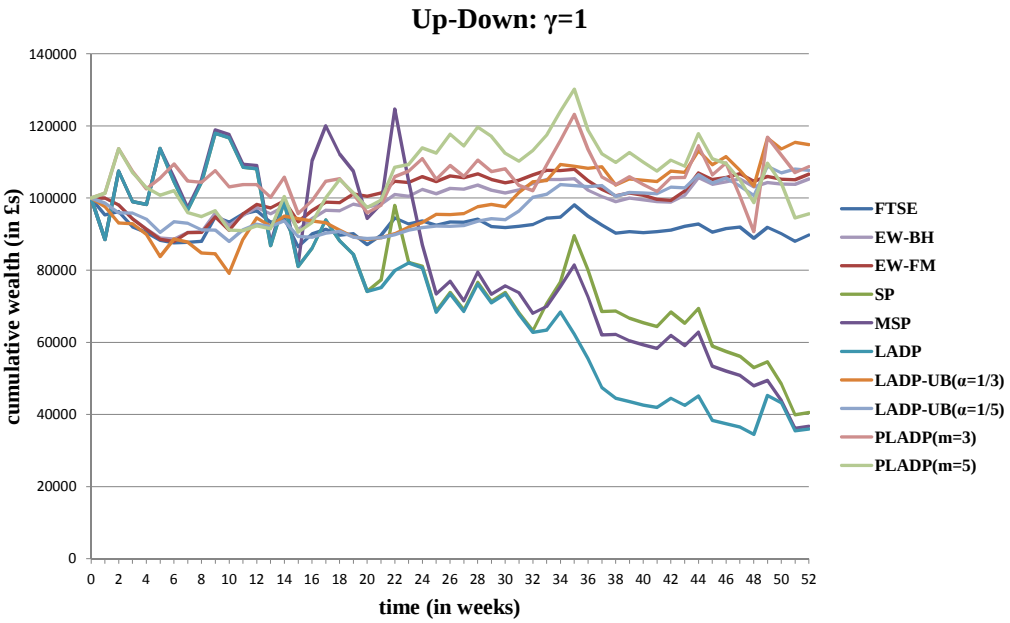


Figure E.12: Out-of-sample cumulative wealth against time: Up-Down, $\gamma = 1$

Down-Up Plots

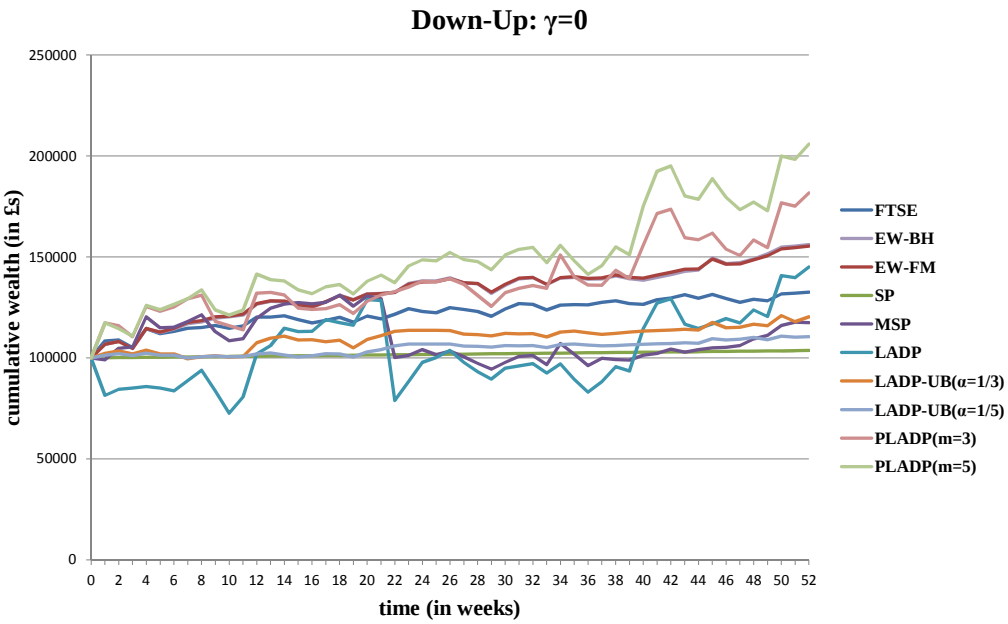


Figure E.13: Out-of-sample cumulative wealth against time: Down-Up, $\gamma = 0$

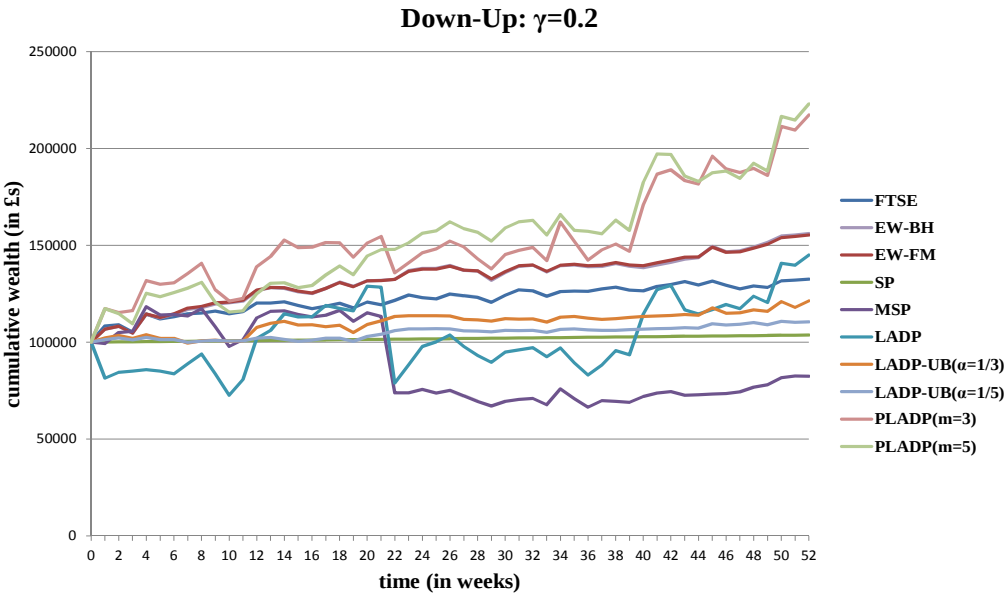


Figure E.14: Out-of-sample cumulative wealth against time: Down-Up, $\gamma = 0.2$

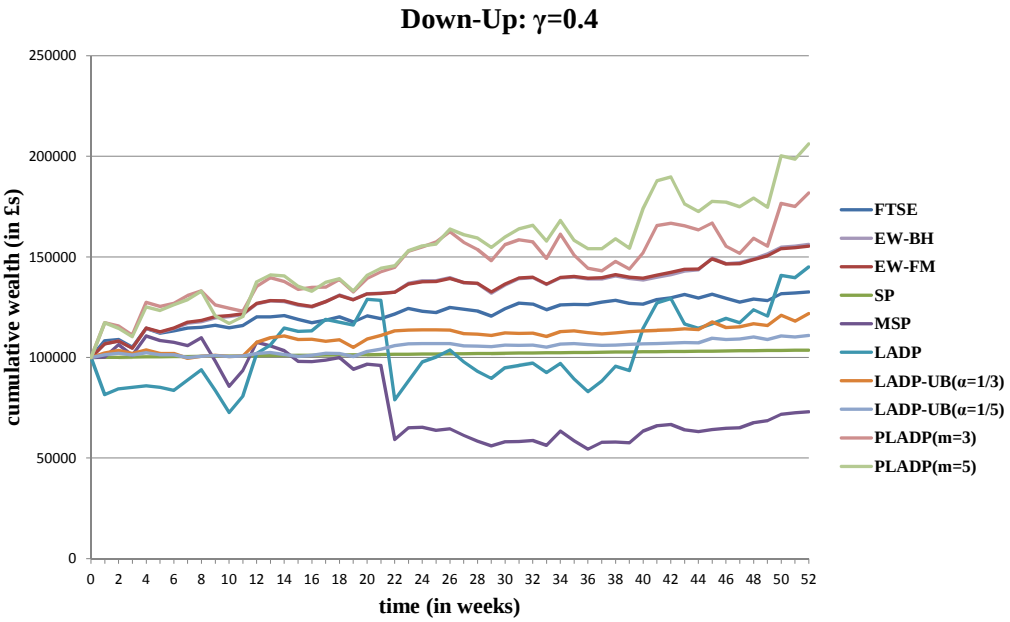


Figure E.15: Out-of-sample cumulative wealth against time: Down-Up, $\gamma = 0.4$

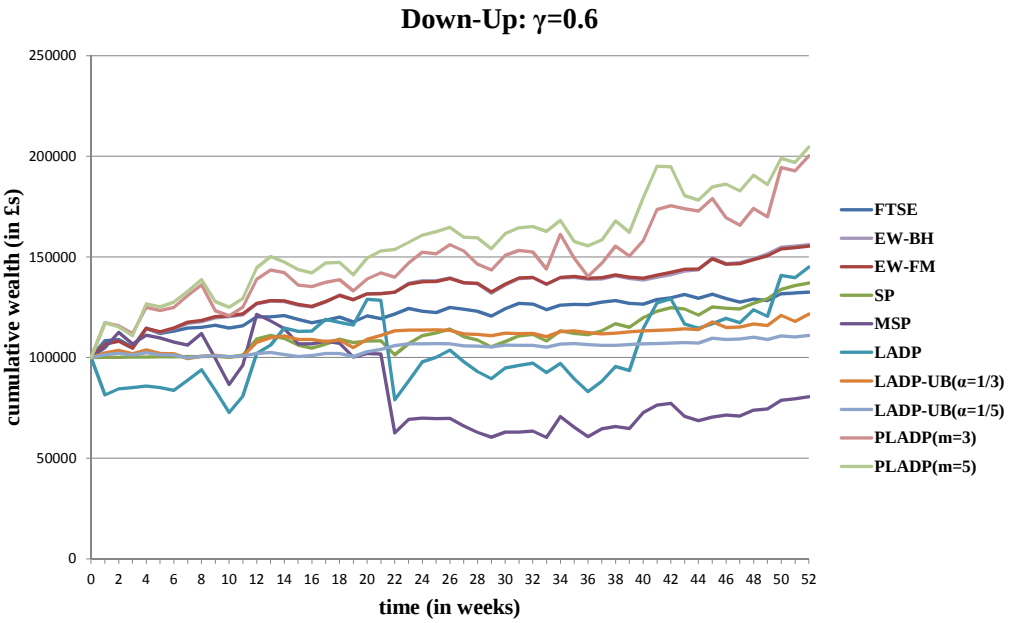


Figure E.16: Out-of-sample cumulative wealth against time: Down-Up, $\gamma = 0.6$

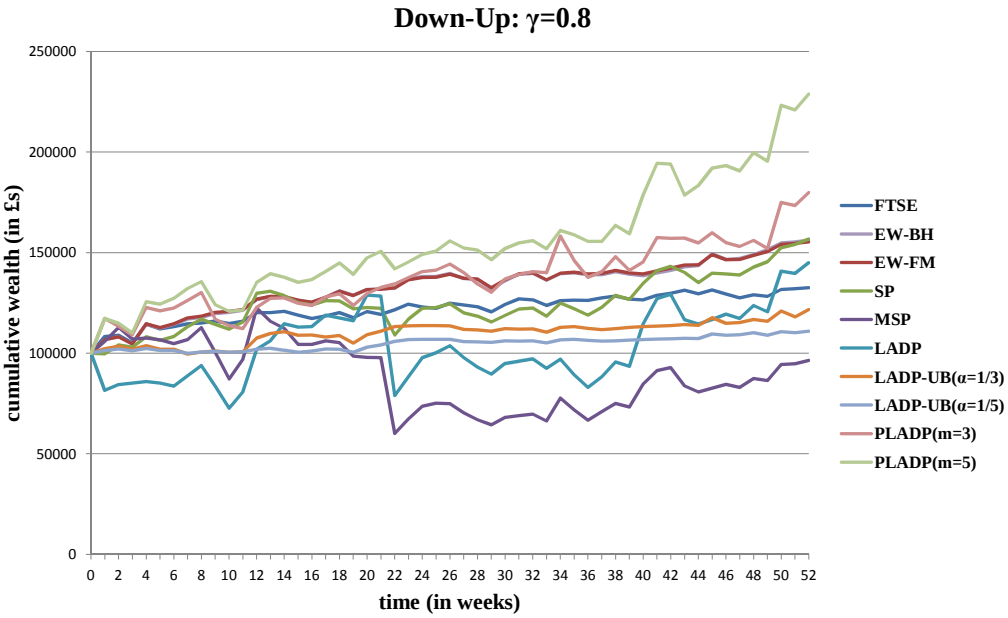


Figure E.17: Out-of-sample cumulative wealth against time: Down-Up, $\gamma = 0.8$

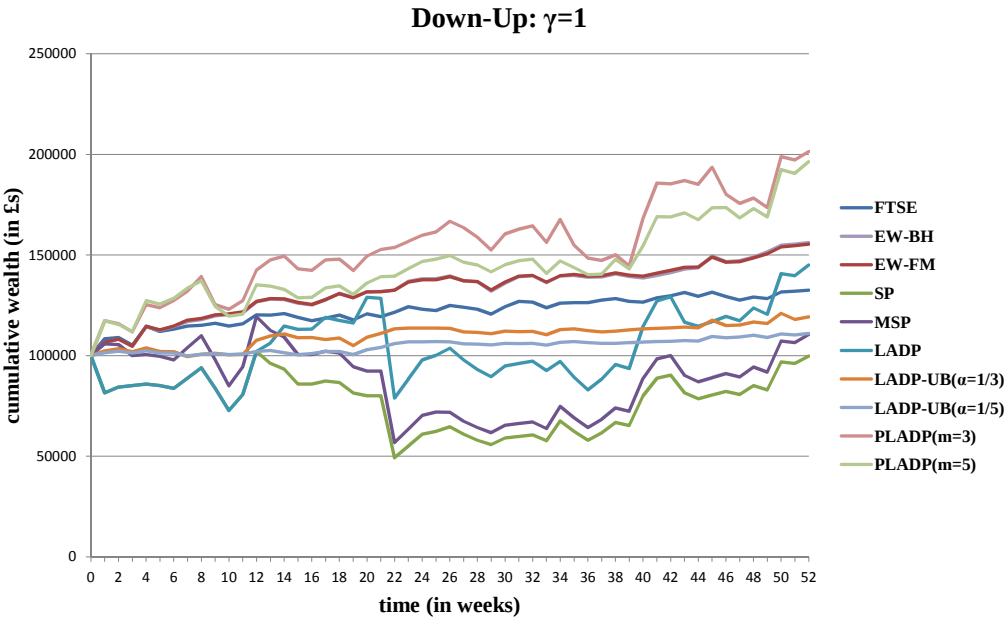


Figure E.18: Out-of-sample cumulative wealth against time: Down-Up, $\gamma = 1$

Down-Down Plots

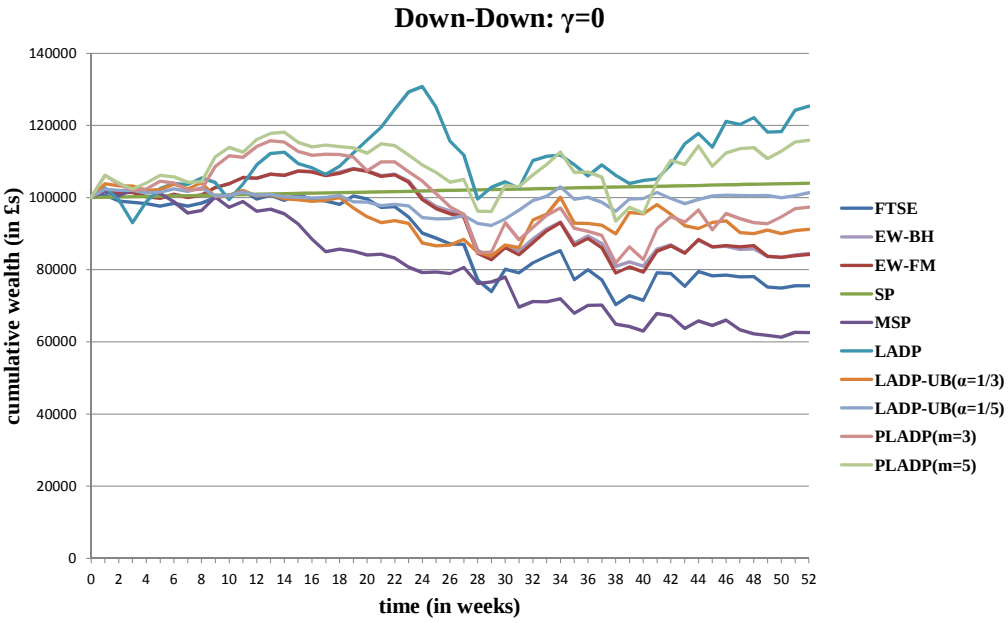


Figure E.19: Out-of-sample cumulative wealth against time: Down-Down, $\gamma = 0$

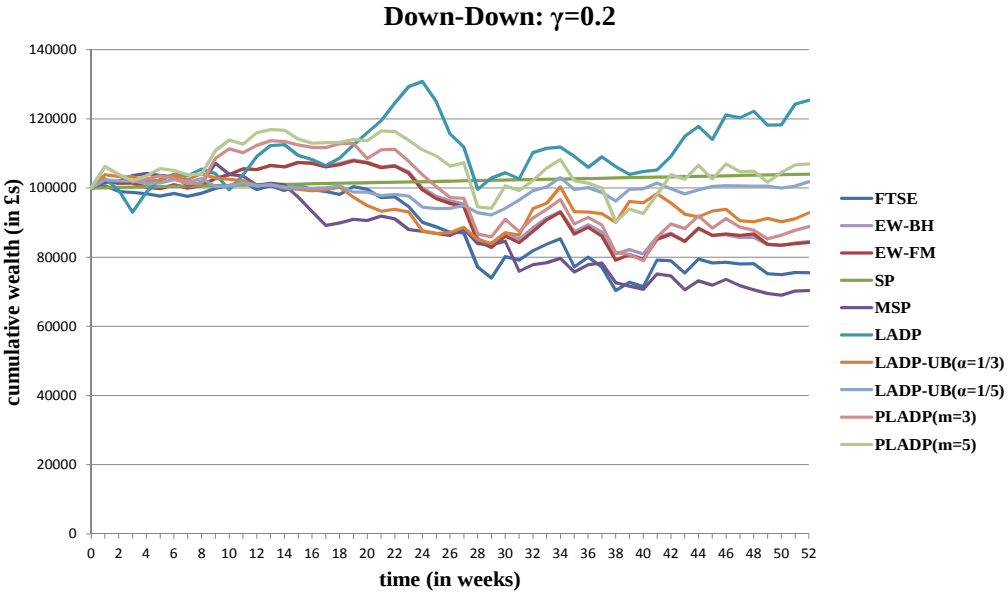


Figure E.20: Out-of-sample cumulative wealth against time: Down-Down, $\gamma = 0.2$

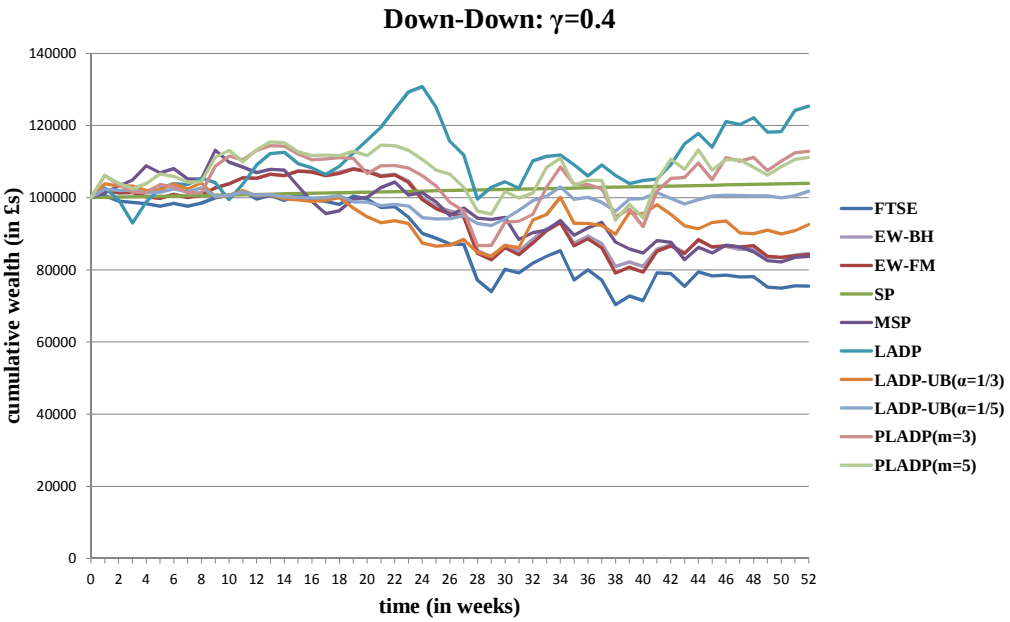


Figure E.21: Out-of-sample cumulative wealth against time: Down-Down, $\gamma = 0.4$

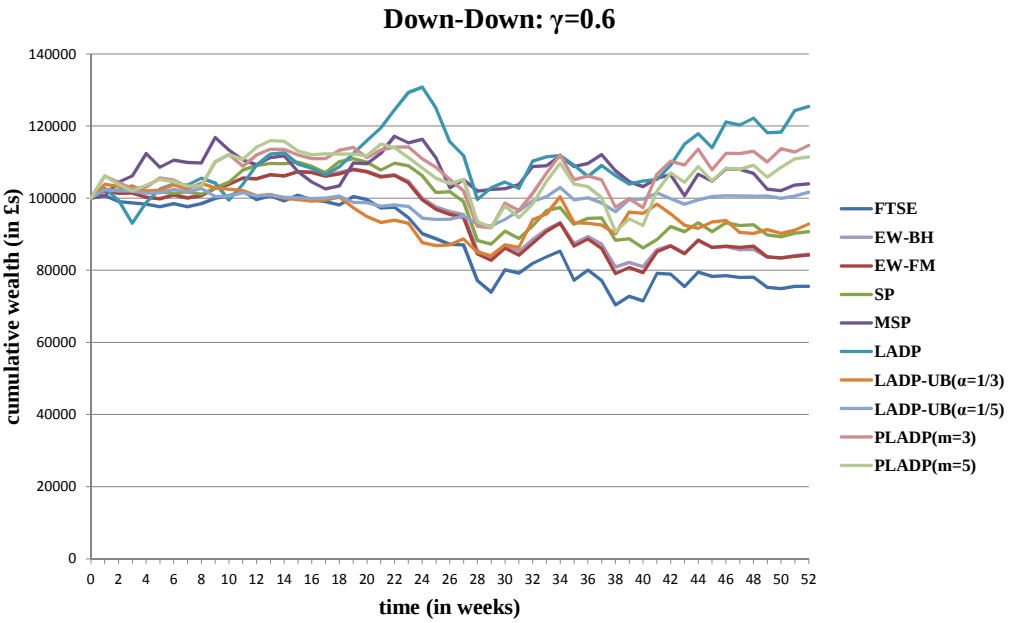


Figure E.22: Out-of-sample cumulative wealth against time: Down-Down, $\gamma = 0.6$

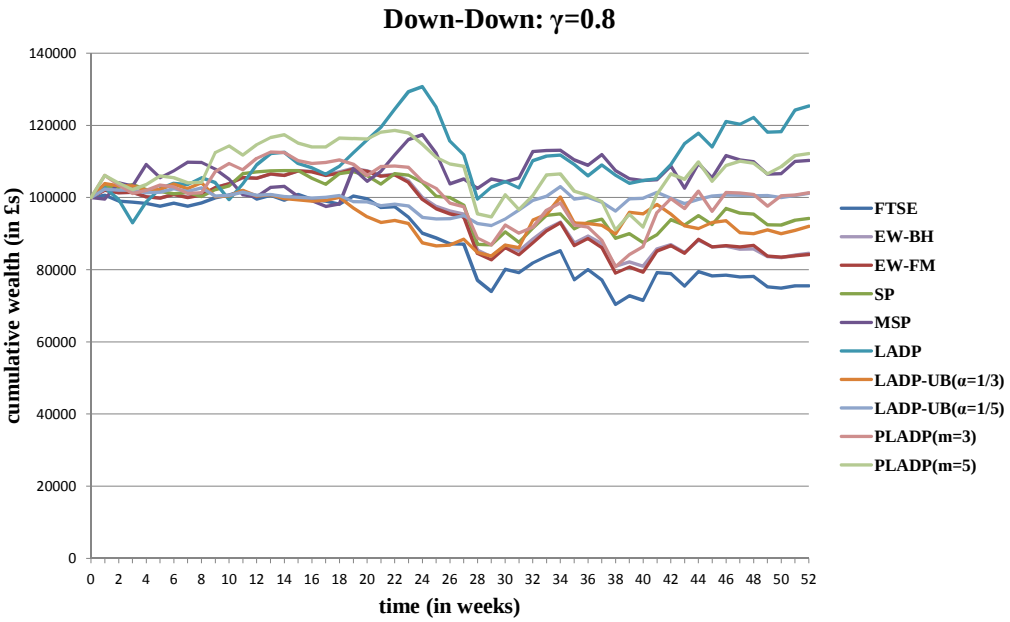


Figure E.23: Out-of-sample cumulative wealth against time: Down-Down, $\gamma = 0.8$

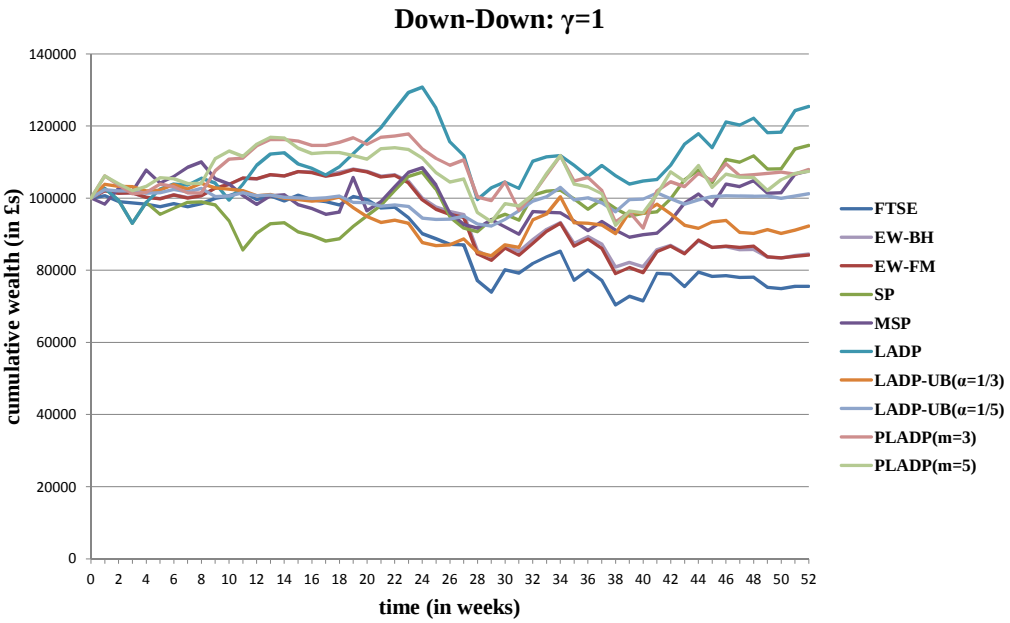


Figure E.24: Out-of-sample cumulative wealth against time: Down-Down, $\gamma = 1$

Bibliography

[1] Carol Alexander. *Mastering Risk Volume II*. Financial Times Prentice Hall, 2001.

[2] F. Andersson, H. Mausser, D. Rosen, and Uryasev S. Credit risk optimization with conditional value-at-risk criterion. *Mathematical Programming*, 89(2):273–291, 2001.

[3] P. Artzner, F. Delbaen, J.M. Eber, and D. Heath. Coherent measures of risk. *Mathematrical Finance*, 9(3):203–228, 1999.

[4] L. Bauwens, S. Laurent, and J. V. K. Rombouts. Mutivariate GARCH models: a survey. *Journal of Applied Econometrics*, 21(1):79–109, 2006.

[5] J. F. Benders. Partitioning procedures for solving mixed-variables programming problems. *Numerische Mathematik*, 4(1):238–252, 1962.

[6] M. Bertocchi, V. Moriggia, and J. Dupačŏvá. Horizon and stages in applications of stochastic programming in finance. *Annals of Operations Research*, 142(1):63–78, 2006.

[7] Dimitri P. Bertsekas. *Dynamic Programming: Deterministic and Stochastic Models*. Prentice-Hall, 1987.

[8] Dimitri P. Bertsekas and John N. Tsitsiklis. *Neuro-dynamic programming*. Athena Scientific, 1996.

[9] Patrick Billingsley. *Probability and measure: Third Edition*. John Wiley & Sons, 1995.

[10] J. R. Birge and F. V. Louveaux. A multicut algorithm for two-stage stochastic linear programs. *European Journal of Operational Research*, 34:384–392, 1988.

[11] T. Bollerslev. Generalized Autoregressive Conditional Heteroskedasticity. *Econometrica*, 31(3):307–327, 1986.

[12] H. Byström. Orthogonal GARCH and covariance matrix forecasting in a stress scenario: The Nordic stock markets during the Asian financial crisis 19971998. *European Journal of Finance*, 10(1):44–67, 2004.

[13] A. Charnes and W.W. Cooper. Chance-constrained programming. *Management science*, 6(1):73–79, 1959.

[14] G. Constantinides. Multiperiod consumption and investment behavior with convex transaction costs. *Management Science*, 25(11):1127–1137, 1979.

[15] G.B. Dantzig. Linear programming under uncertainty. *Management science*, 1(3/4):197–206, 1955.

[16] G.B. Dantzig and G. Infanger. Multi-stage stochastic linear programs for portfolio optimization. *Annals of Operations Research*, 45(1):59–76, 1993.

[17] George B. Dantzig and Mukund N. Thapa. *Linear Programming I: Introduction*. Springer, 1997.

[18] D.P. de Farias and B. Van Roy. The linear programming approach to approximate dynamic programming. *Operations Research*, 51(6):850–865, 2003.

[19] D.P. de Farias and B. Van Roy. A cost-shaping linear program for average-cost approximate dynamic programming with performance guarantees. *Mathematics of Operations Research*, 31(3):597–620, 2006.

[20] B. Defourny, D. Ernst, and L. Wehenkel. Scenario trees and policy selection for multistage stochastic programming using machine learning. *INFORMS Journal on Computing*, pages 1–14, 2012.

[21] V. DeMiguel, L. Garlappi, and F.J. Nogales. A Generalized Approach to Portfolio Optimization: Improving Performance by Constraining Portfolio Norms. *Management Science*, 55(5):798–812, 2009.

[22] V. DeMiguel, L. Garlappi, and R. Uppal. Optimal versus Naive Diversification: How Inefficient is the 1/N Portfolio Strategy? *Review of Financial Studies*, 22(5):1915–1953, 2009.

[23] V. DeMiguel and F.J. Nogales. Portfolio Selection with Robust Estimation. *Operations Research*, 57(3):560–577, 2009.

[24] N.D. Domenica, G. Mitra, P. Valente, and G. Birbilis. Stochastic programming and scenario generation within a simulation framework: An information systems perspective. *Decision Support Systems*, 42(4):2197–2218, 2007.

[25] J. Dupačŏvá, G. Consigli, and S.W. Wallace. Scenarios for multistage stochastic programs. *Annals of Operations Research*, 100(1-4):25–53, 2000.

[26] J. Dupačŏvá, N. Gröwe-Kuska, and W. Römisch. Scenario reduction in stochastic programming. *Mathematical Programming*, 95(3):493–511, 2003.

[27] A. Eichorn and W. Römisch. *Mean-risk optimization models for electricity portfolio management*. Proceedings of the 9th Int. Conf. on Probabilistic Methods Applied to Power Systems, 2006.

[28] F. R. Engle. Autoregressive Conditional Heteroskedasticity with Estimates of the Variance of United Kingdom Inflation. *Econometrica: Journal of Econometric Society*, 50(4):987–1007, 1982.

[29] R. Engle. Dynamic Conditional Correlation - A Simple Class of Multivariate GARCH Models. *Journal of Business and Economic Statistics*, 20(3):339–350, 2002.

[30] E.F. Fama. Multiperiod consumption-investment decisions. *The American Economic Review*, 60(1):163–174, 1970.

[31] S.-E. Fleten, K. Hoyland, and S.W. Wallace. The performance of stochastic dynamic and fixed mix portfolio models. *European Journal of Operational Research*, 140(1):37–49, 2002.

[32] H. I. Gassmann. MSLIP: A computer code for the multistage stochastic linear programming problem. *Mathematical Programming*, 47(1-3):407–423, 1990.

[33] A.P. George and W.B. Powell. Adaptive stepsizes for recursive estimation with applications in approximate dynamic programming. *Machine Learning*, 65(1):167–198, 2006.

[34] G.A. Godfrey and W.B. Powell. An Adaptive Dynamic Programming Algorithm for Dynamic Fleet Management, I: Single Period Travel Times. *Transportation Science*, 36(1):21–39, 2002.

[35] G.A. Godfrey and W.B. Powell. An Adaptive Dynamic Programming Algorithm for Dynamic Fleet Management, II: Multiperiod Travel Times. *Transportation Science*, 36(1):40–54, 2002.

[36] Gregory A. Godfrey and Warren B. Powell. An adaptive approximation method for stochastic, dynamic programs, with applications to inventory and distribution problems. Technical report, Princeton University, 1997.

[37] N. Gröwe-Kuska, H. Heitsch, and W. Römisch. Scenario reduction and scenario tree construction for power management problems. IEEE Bologna Power Tech Conference Proceedings, 2003.

[38] G. Guastaroba, R. Mansini, and G.M. Speranza. Models and Simulations for Portfolio Rebalancing. *Computational Economics*, 33(3):237–262, 2009.

[39] G. Guastaroba, R. Mansini, and G.M. Speranza. On the effectiveness of scenario generation techniques in single-period portfolio optimization. *European Journal of Operational Research*, 192(2):500–511, 2009.

[40] G. Guastaroba, G. Mitra, and G.M. Speranza. Investigating the effectiveness of robust portfolio optimization techniques. *Journal of Asset Management*, 12(4):260–280, 2011.

[41] N. Gulpinar, B. Rustem, and R. Settergren. Simulation and optimization approaches to scenario tree generation. *Journal of Economic Dynamics and Control*, 28(7):1291–1315, 2004.

[42] N.H. Hakansson. Optimal investment and consumption strategies under risk for a class of utility functions. *Econometrica: Journal of the Econometric Society*, 38(5):587–607, 1970.

[43] H. Heitsch and W. Römisch. Scenario tree modeling for multistage stochastic programs. *Mathematical Programming*, 118(2):371–406, 2009.

[44] Peter Kall and Janos Mayer. *Stochastic Linear Programming*. Springer's International Series, 2005.

[45] M. Kaut and S.W. Wallace. Evaluation of scenario-generation methods for stochastic programming. *Pacific Journal of Optimization*, 3(2):257–271, 2007.

[46] H. Kellerer, U. Pferschy, and D. Pisinger. *Knapsack Problems*. Springer, 2004.

[47] H. Konno and H. Yamazaki. Mean-absolute deviation portfolio optimization model and its application to Tokyo stock market. *Management Science*, 37(5):519–531, 1991.

[48] P. Krokhmal, J. Palmquist, and S. Uryasev. Portfolio Optimization with Conditional Value-at-Risk Objective and Constraints. *Journal of Risk*, 4:11–27, 2002.

[49] D. Kuhn, W. Wiesemann, and A. Georghiou. Primal and dual linear decision rules in stochastic and robust optimization. *Mathematical Programming*, 130(1):177–209, 2011.

[50] S. Kunnumkal and H. Topaloglu. Stochastic approximation algorithms and max-norm "projections". Technical report, School of Operations Research and Industrial Engineering in Cornell University, 2005.

[51] A. Künzi-Bay and J. Mayer. Computational aspects of minimizing conditional value-at-risk. *Computational Management Science*, 3(1):3–27, 2006.

[52] David G. Luenberger. *Investment Science*. Oxford University Press, 1998.

[53] R. Mansini, W. Ogryczak, and G.M. Speranza. LP solvable models for portfolio optimization: a classification and computational comparison. *IMA Journal of Management Mathematics*, 14(3):187–220, 2003.

[54] H. Markowitz. Portfolio selection. *Journal of Finance*, 7(1):77–91, 1952.

[55] Harry Markowitz. *Portfolio Selection: Efficient Diversification of Investments*. John Wiley & Sons, 1959.

[56] Robert C. Merton. *Continuous-time finance*. Basil Blackwell, 1990.

[57] J. Mossin. Optimal Multiperiod Portfolio Policies. *The Journal of Business*, 41(2):215–229, 1968.

[58] J. M. Mulvey, W. R. Pauling, and R. E. Madey. Advantages of multiperiod portfolio models. *The Journal of Portfolio Management*, 29(2):35–45, 2003.

[59] J.M. Mulvey and Vladimirou H. Stochastic network programming for financial planning problems. *Management science*, 38(11):1642–1664, 1992.

[60] J. Nascimento and W.B. Powell. Dynamic Programming Models and Algorithms for the Mutual Fund Cash Balance Problem. *Management Science*, 56(5):801–815, 2010.

[61] K.P. Papadaki and V. Friderikos. Approximate dynamic programming for link scheduling in wireless mesh networks. *Computers and Operations Research*, 35(12):3848–3859, 2007.

[62] K.P. Papadaki and W.B. Powell. An adaptive dynamic programming algorithm for a stochastic multiproduct batch dispatch problem. *Naval Research Logistics*, 50(7):742–769, 2003.

[63] A.F. Perold. Large-scale portfolio optimization. *Management Science*, 30(10):1143–1160, 1984.

[64] A.F. Perold and W.F. Sharpe. Dynamic Strategies for Asset Allocation. *Financial Analysts Journal*, 44(1):16–27, 1988.

[65] Warren B. Powell. *Approximate Dynamic Programming: Solving the Curses of Dimensionality*. John Wiley & Sons, 2008.

[66] Warren B. Powell and Benjamin Van Roy. Approximate dynamic programming for high dimensional resource allocation problems. In *Handbook of Learning and Approximate Dynamic Programming*. IEEE Press, 2004.

[67] Warren B. Powell and Huseyin Topaloglu. *Approximate dynamic programming for large-scale resource allocation problems*. Tutorials in Operations Research, INFORMS, 2005.

[68] W.B. Powell, A. Ruszczynski, and H. Topaloglu. Learning algorithms for separable approximations of discrete stochastic optimization problems. *Mathematics of Operations Research*, 29(4):814–836, 2004.

[69] Martin L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, 1994.

[70] R. T. Rockafellar and S. Uryasev. Optimization of conditional value-at-risk. *Journal of Risk*, 2(3):21–41, 2000.

[71] P. A. Samuelson. Lifetime portfolio selection by dynamic stochastic programming. *The Review of Economics and Statistics*, 51(3):239–246, 1969.

[72] Alexander Shapiro, Darinka Dentcheva, and Andrzej Ruszczynski. *Lectures on Stochastic Programming: modeling and theory*. MPS-SIAM Series on Optimization, 2009.

[73] W.F. Sharpe. A simplified model for portfolio analysis. *Management Science*, 9(2):277–293, 1963.

[74] W.F. Sharpe. Capital Asset Prices: A Linear Programming Algorithm for Mutual Fund Portfolio Selection. *Management Science*, 13(7):499–510, 1964.

[75] W.F. Sharpe. Capital asset prices: A theory of market equilibrium under conditions of risk. *The Journal of Finance*, 19(3):425–442, 1964.

[76] W.F. Sharpe. A linear programming approximation for the general portfolio analysis problem. *Journal of Financial and Quantitative Analysis*, 6(5):1263–1275, 1971.

[77] Richard S. Sutton and Andrew G. Barto. *Reinforcement learning: An Introduction*. The MIT press, 1998.

[78] J. Tobin. Liquidity preference as behavior towards risk. *The Review of Economic Studies*, 2(25):65–86, 1958.

[79] N. Topaloglou, H. Vladimirou, and S.A. Zenios. A dynamic stochastic programming model for international portfolio management. *European Journal of Operational Research*, 185(3):1501–1524, 2008.

[80] H. Topaloglu and S. Kunnumkal. Approximate dynamic programming methods for an inventory allocation problem under uncertainty. *Naval Research Logistics*, 53(8):822–841, 2006.

[81] H. Topaloglu and W.B. Powell. An algorithm for approximating piecewise linear concave functions from sample gradients. *Operations Research Letters*, 31(1):66–76, 2003.

[82] Stan Uryasev. *Probabilistic Constrained Optimization: Methodology and Applications*. Kluwer Academic Publishers, 2000.

[83] F. Woerheide and D. Persson. An index of portfolio diversification. *Financial Services Review*, 2(2):73–85, 1993.

[84] S. Yitzhaki. Stochastic dominance, mean variance, and gini's mean difference. *The American Economic Review*, 72(1):178–185, 1982.

[85] L. Y. Yu, X. D. Ji, and S. Y. Wang. Stochastic programming models in financial optimization: A survey. *Advanced Modeling and Optimization*, 5(1):1–26, 2003.

[86] S.A. Zenios, M.R. Holmer, R. McKendall, and C. Vassiadou-Zeniou. Dynamic models for fixed-income portfolio management under uncertainty. *Journal of Economic Dynamics and Control*, 22(10):1517–1541, 1998.