

Portfolio Optimization with Transaction Costs

A Major Qualifying Project Report
Submitted to the Faculty of
WORCESTER POLYTECHNIC INSTITUTE
In partial fulfillment of the requirements for the
Degree of Bachelor of Science
by

Jessica M. Clark

Sean E. Mulready

Approved:

Professor Arthur Heinricher, Advisor

APRIL 26, 2007

Contents

1	Introduction	1
2	Background	3
2.1	Basic Portfolio Analysis	3
2.2	Portfolio Optimization	5
2.2.1	Risk versus Return	5
2.2.2	Efficient Portfolios	6
2.2.3	Fully Invested Constraint	7
3	Constraints	9
3.1	Booksize	9
3.2	Turnover and Transaction Costs	10
4	Implementation of Portfolio Optimization	12
4.1	Constraint Linearization	12
4.2	Proof of Validity of Linearization	13
4.3	Standard Constraint Formulation	15
4.4	Description of Algorithm	17
5	Results	19
5.1	Data Management	19
5.1.1	Source of Data	19
5.1.2	Definition of Time Periods	19
5.1.3	Verifying Validity of Optimization	20
5.2	Evolution of Portfolios	20
5.3	Portfolio Value Calculations	21
5.4	Comparing Value and Utility Plots	22
5.4.1	Efficient vs. Equally Weighted Plots	22
5.4.2	Booksize Plots	22
5.4.3	Turnover Plots	24
6	Explorations	29
6.1	Turnover Budgeting	29
6.2	Threshold Trading	30

7 Conclusion	34
8 Appendices	36
8.1 Appendix A: MatLab Code	36
8.2 Appendix B: PowerPoint Presentation	47

Abstract

Investors often update their portfolios at regular time intervals by trading stocks, but there are costs associated with these trades. This project seeks to limit these transaction costs by controlling the portfolio turnover (absolute change as a fraction of book size) between time periods. The result is a multi-period optimization problem with quadratic objective function and non-smooth constraints. The resulting portfolios outperformed benchmark portfolios in both expected utility and actual portfolio value.

Chapter 1

Introduction

Investors often update their stock portfolios at regular time intervals, buying and selling stocks to maximize expected return while also minimizing risk. One standard approach is minimize a utility function incorporating both risk and return, typically with a parameter to measure risk tolerance and additional constraints.[3] In traditional portfolio optimization, the sum of the fraction invested in each stock is constrained. If short-selling is allowed, then it is useful to instead constrain the *booksize*, the sum of the absolute values of the weights. This mathematical framework allows investors to select an optimal (or at least efficient) portfolio at any given time.

In practice, portfolios are held for long time periods and are adjusted, or re-optimized at specific intervals. While Markowitz [3] showed how to find the best portfolio at a given time, the basic formulation does not include the costs associated with *changing* the weights in the portfolio. Each transaction has an associated cost and these costs must be included in the decision process.

It is difficult to model these costs, because they can depend on many factors

including the size of the trade and the type of asset being traded. One way to limit transaction costs is by introducing a constraint on *turnover*, the sum of the absolute differences in weights between adjacent time periods. The goals of this project were to implement a portfolio optimization algorithm with both booksize and turnover constraints and to explore the effects of the turnover constraint on portfolio value and utility in a multi-period setting.

The objective function of the portfolio optimization problem is quadratic; however, the constraints on booksize and turnover are nonlinear and non-smooth. Since standard quadratic programming packages require linear constraints, the constraints were linearized by splitting the variables.

The optimization algorithm was tested on a set of four stocks over eleven time periods. Both the expected utility and actual performance of the resulting portfolios were compared to those of unconstrained and equally weighted portfolios. The optimal portfolios typically outperformed the equally weighted portfolios. Comparisons were also made of the performance of portfolios with different allowances of booksize and turnover.

With turnover constraints in place, the ideas of *turnover budgeting* and *threshold trading* were developed and tested. Turnover budgeting allows dynamic allocation of turnover across multiple time periods. Threshold trading limits trading subject to criteria on the expected change in utility. These and other extensions of the turnover constraint can be used to model and limit transaction costs in more realistic ways.

Chapter 2

Background

2.1 Basic Portfolio Analysis

There are two main problems in portfolio analysis: *estimation* and *optimization*. The estimation problem is basically the problem of forecasting investment return. In a sense this, is a statistical or econometric problem and it is the most difficult of the two problems. If a manager has very good forecasts for stock behavior, investment decisions are very easy.

The focus of this project is almost entirely on the optimization problem. Given a set of data and some basic statistics computed from that data, what is the optimal portfolio choice? How can the investor balance the goal of high return against the challenge of large risks? This view, the idea that investors balance return and risk, was the insight of Harry Markowitz and his view is the foundation for Mean–Variance analysis in portfolio management.

This project considers the situation where an investor is choosing among a set of n stocks in which to invest. This set of stocks can be represented as an $n \times 1$ vector. Let this vector be denoted as \vec{x} . Then, x_i is the number of shares

invested in asset i .

Notice that the entries in \vec{x} can be positive, 0, or negative. If $x_i > 0$, then the investor owns a positive number of shares in asset i . If $x_i < 0$, then the investor has a short position in asset i . This means that the investor has borrowed shares of stock from someone else, sold them, and kept the proceeds from the sale. At some point in the future, the investor will have to buy the shares of stock back and give them to the original owner. This is called closing the short position. Clearly, short selling is done with the hope that the stock will drop in price. If $x_i = 0$, the investor owns no shares of stock i .

To further illustrate the idea of short selling, consider investor A and investor B. Assume that Microsoft stock is worth \$30 per share today. Investor A thinks that Microsoft will go down in the next week, so he borrows 10 shares of Microsoft stock from investor B and sells them immediately, collecting \$300. If Microsoft stock drops to \$20 per share next week, investor A can buy the 10 shares back for \$200 and close his short position with B by giving the shares back. Investor A would make a profit of \$100 from this transaction. However, if Microsoft stock goes up, A will lose money. For instance, if Microsoft is worth \$60 next week, it will cost A \$600 to buy the stocks back and close the position, earning him a net loss of \$300. For this reason, short selling is riskier than only buying stocks because the possible amount of loss is unbounded. Stock prices cannot drop below \$0 per share, but they can go up almost indefinitely.

2.2 Portfolio Optimization

2.2.1 Risk versus Return

Investors want to choose their portfolio to minimize risk while simultaneously obtaining the maximum amount of return. These two objectives can sometimes oppose each other. For instance, consider the portfolio where all of the investor's money is invested in the stock that he thinks will perform the best. Although this portfolio has a high amount of expected return, it also is very risky, because if that one stock drops, the investor does not have any other assets that could make up for it. On the other hand, if an investor invests equally in all possible stocks, he is not taking advantage of any stocks that will probably perform better than others.

Economists have long studied the ways that investors attempt to balance risk versus return, but it was Harry Markowitz [3] who first *defined* risk as variance in portfolio return.

$$\text{Risk} = \vec{x}^T \Omega \vec{x} = \sum_{i,j} x_i \sigma_{ij} x_j \quad (2.1)$$

$$\text{Return} = \vec{\mu}^T \vec{x} = \sum_i \mu_i x_i \quad (2.2)$$

where Ω is the $(n \times n)$ covariance matrix for the assets and $\vec{\mu}$ is the $((n \times 1)$ vector of returns. These are estimated from historical data for stock prices.

2.2.2 Efficient Portfolios

There is not a single portfolio which is “optimal” for all investors. Different investors have different levels of risk tolerance and are willing to carry more risk for the chance of obtaining more return. It is, however, possible to identify portfolios which an investor should consider.

An *efficient portfolio* has the lowest risk for a given return, as well as the highest return for that amount of risk [3].

Efficient portfolios are found by solving the optimization problem

$$\text{Minimize } \vec{x}^T \Omega \vec{x} - \frac{1}{\lambda} \vec{\mu}^T \vec{x} \quad (2.3)$$

where λ is a parameter representing the investor’s risk tolerance. If λ is large, then $\frac{1}{\lambda}$ will be close to zero, meaning that the investor does not have much risk tolerance (most of the emphasis in the optimization problem is placed on risk). Conversely, if λ is small, then $\frac{1}{\lambda}$ will be large, placing more emphasis on return.

The *efficient frontier* is the curve made up of all efficient portfolios. It is calculated by running the above optimization over many values of λ , and visualized by plotting all of the efficient frontiers in risk-return space, as in figure 2.1 [3].

The efficient frontier can be interpreted as the collection of all portfolios that have the highest risk for a given amount of return, as well as the highest return for that amount of risk. In figure 2.1, notice that portfolio a is not on the efficient frontier because one can achieve a higher amount of return for the same amount of risk with portfolio b, and a lower amount of risk for the same amount

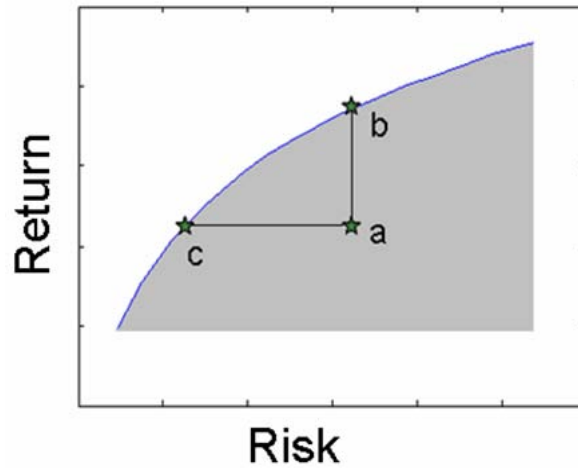


Figure 2.1: Sample Efficient Frontier

of return with portfolio c. Also notice that in order to get a large amount of return, one must be willing to take on a lot of risk, and to get a very small amount of risk, one must be able to accept low expected return.

2.2.3 Fully Invested Constraint

A constraint that is commonly placed on the classical portfolio optimization problem is called the *Fully Invested* constraint, which requires that the amount of money invested in the portfolio is equal to some number F . This can be viewed as a requirement that the investor buys only the number of stocks he wants to, while not exceeding that number of stocks. With the fully invested constraint added, the optimization problem becomes:

$$\text{Minimize } \vec{x}^T \Omega \vec{x} - \frac{1}{\lambda} \vec{\mu}^T \vec{x} \quad (2.4)$$

$$s.t. \quad \sum_{i=1}^n x_i = F \quad (2.5)$$

The fully invested constraint is typically used for problems where short selling is not allowed. As is shown in the next section, the fully invested constraint is not very meaningful when applied to portfolios where short selling is allowed.

Chapter 3

Constraints

3.1 Booksize

In classic portfolio optimization, there typically exists a fully invested constraint, which sets the sum of the portfolio weights equal to a constant. As our model allows for short-selling, $x_i < 0$, this constraint is not very meaningful. Instead, this approach limits the sum of the absolute value of portfolio weights, or the booksize.

$$\text{Booksize} = \sum_{i=1}^n |x_i| \leq M \quad (3.1)$$

Here's a simple two-stock example to illustrate the importance of booksize when short-selling is allowed:

At time 0, there are two stocks, each priced at \$10 per share. Say someone buys ten shares of Stock 1 and sells-short ten shares of Stock 2. To purchase the ten shares of Stock 1, the person withdrew \$100 from a bank account. However, when selling-short in Stock 2, he received that \$100 back. Therefore, his net change in wealth is 0. The position vector becomes $\vec{x} = \begin{bmatrix} 10 \\ -10 \end{bmatrix}$. Notice the

sum of the weights equals 0. Also the booksize is 20.

At time 1, things go bad for the investor. Stock 1 drops in price to \$5 per share while Stock 2 increases to \$15 per share. The investor decides to sell his shares in Stock 1 and close his position in Stock 2. By selling Stock 1, the investor gains \$50. However, to close his position in Stock 2, the investor must withdraw \$150. The investor now has a net change in wealth of -\$100. This loss could not have been predicted by examining only the net weight in shares, which equalled 0. This shows the importance of booksize with regards to the amount of money at risk for loss.

3.2 Turnover and Transaction Costs

There are three types of transaction costs associated with every trade: commission, the bid/ask spread, and market impact. Commission is the amount a broker charges for performing a trade, which can be a set price or depend on the size of the trade. The bid/ask spread is the difference in price at which one can buy a share of stock and then immediately sell it. The market impact is the cost associated with trading several shares of a stock, i.e. changing the price of the stock. There is also a fourth type of transaction cost called opportunity cost which measures the amount lost for not trading a particular stock. [2]

An investor may want to limit the amount that the portfolio changes over time, because of the transaction costs associated with changing the portfolio (buying and selling assets). The turnover constraint is introduced to accomplish this.

The *turnover* is the sum of the absolute values of the difference between each position at time t and time $t + 1$.

$$\text{Turnover}(t) = \sum_{i=1}^n |x_i(t) - x_i(t-1)| \quad (3.2)$$

where $x_i(t)$ refers to the position in asset i at time t and $x_i(t-1)$ refers to the position in asset i at time $t-1$.

The turnover is constrained by requiring that it is less than or equal to some constant, typically between 0 and 1, multiplied by the booksize of the portfolio at time $t-1$.

$$\sum_{i=1}^n |x_i(t) - x_i(t-1)| \leq L \sum_{i=1}^n |x_i(t-1)| \quad (3.3)$$

The optimization problem is then:

$$\begin{aligned} &\text{Minimize} \quad \vec{x}^T \Omega \vec{x} - \frac{1}{\lambda} \vec{\mu}^T \vec{x} \\ &s.t. \quad \sum_{i=1}^n |x_i| \leq M \\ &\quad \sum_{i=1}^n |x_i - x_i^{old}| \leq L \sum_{i=1}^n |x_i^{old}| \end{aligned} \quad (3.4)$$

Chapter 4

Implementation of Portfolio Optimization

4.1 Constraint Linearization

Notice that the optimization has a quadratic objective function, so it would be efficient and effective to use a standard quadratic programming tool (such as MatLab's `quadprog`); however, standard quadratic programming tools require linear constraints. The booksize and turnover constraints are nonlinear because the sums of absolute values are being calculated. The constraints were therefore linearized by splitting each x_i into two new variables, b_i^+ and b_i^- for the booksize constraint, and y_i^+ and y_i^- for the turnover constraint. Introducing split variables transforms the problem into the following standard quadratic programming problem:

$$\text{Minimize } \vec{x}^T \Omega \vec{x} - \frac{1}{\lambda} \vec{\mu}^T \vec{x}$$

$$\begin{aligned}
s.t. \quad & x_i = b_i^+ - b_i^- \\
& b_i^+ \geq 0 \\
& b_i^- \geq 0 \\
& \sum_{i=1}^n b_i^+ + b_i^- \leq M \\
& x_i = y_i^+ - y_i^- \\
& y_i^+ \geq 0 \\
& y_i^- \geq 0 \\
& \sum_{i=1}^n y_i^+ + y_i^- \leq L \sum_{i=1}^n |x_i^{old}|
\end{aligned}$$

An interpretation of the booksize split variables (b_i^+ and b_i^-) is that

$$b_i^+ = \begin{cases} x_i + c, & \text{if } x_i \geq 0 \\ d, & \text{if } x_i < 0 \end{cases}$$

$$b_i^- = \begin{cases} c, & \text{if } x_i \geq 0 \\ -x_i + d, & \text{if } x_i < 0 \end{cases}$$

where c and d are constants. This interpretation says that the split variables are the positive and negative components of each x_i , respectively, shifted by a constant. The interpretation is the same for the turnover constraint's split variables.

4.2 Proof of Validity of Linearization

Let $\vec{x} = (x_1, \dots, x_n)$, $\vec{b}^+ = (b_1^+, \dots, b_n^+)$, $\vec{b}^- = (b_1^-, \dots, b_n^-)$.

Define two sets. Let

$$A := \left\{ \begin{array}{l} (\vec{x}, \vec{b}^+, \vec{b}^-) s.t. \\ x_i = b_i^+ - b_i^-, \text{ (for each } i) \\ b_i^+ \geq 0, \\ b_i^- \geq 0, \\ \sum_{i=1}^n b_i^+ + b_i^- \leq M \end{array} \right\} \quad (4.1)$$

Let

$$B := \{\vec{x} \text{ s.t. } \sum_{i=1}^n |x_i| \leq M\} \quad (4.2)$$

We want to show that minimizing utility for A is the same as minimizing utility for B. To do this, we show that $\forall \vec{x} \in B, \exists (b^+, b^-)$ such that $(\vec{x}, b^+, b^-) \in A$, and that $\forall (\vec{x}, b^+, b^-) \in A, \vec{x} \in B$.

Let $\vec{x} \in B$ be given. Then $\sum_{i=1}^n |x_i| \leq M$. We need to show that we can find b^+, b^- such that $(\vec{x}, b^+, b^-) \in A$.

For each i , let

$$b_i^+ := \begin{cases} x_i, & \text{if } x_i \geq 0 \\ 0, & \text{if } x_i < 0 \end{cases}$$

$$b_i^- := \begin{cases} 0, & \text{if } x_i \geq 0 \\ -x_i, & \text{if } x_i < 0 \end{cases}$$

Then, $x_i = b_i^+ - b_i^-, b_i^+ \geq 0, b_i^- \geq 0$, and $b_i^+ + b_i^- = |x_i| \leq M$ and so $\sum_{i=1}^n b_i^+ + b_i^- \leq M$ and this implies that if $\vec{x} \in B$, then there exists b^+, b^- such that $(\vec{x}, b^+, b^-) \in A$.

Next, let $(\vec{x}, b^+, b^-) \in A$ be given. Then $x_i = b_i^+ - b_i^-, b_i^+ \geq 0, b_i^- \geq 0, \sum_{i=1}^n b_i^+ + b_i^- \leq M$. Notice:

$$\sum_{i=1}^n x_i = \sum_{i=1}^n b_i^+ - b_i^- \leq \sum_{i=1}^n b_i^+ + b_i^- \leq M$$

Because for each $i, b_i^+ \geq 0, b_i^- \geq 0$. There are two cases: $x_i \geq 0$, and $x_i < 0$.

Case (i):

$$x_i \geq 0 \Rightarrow x_i = |x_i| = b_i^- b_i^-$$

Case (ii):

$$x_i \leq 0 \Rightarrow -x = |x| = b_i^- - b_i^+ \leq b_i^+ + b_i^-$$

which implies that

$$\sum_{i=1}^n x_i \leq \sum_{i=1}^n b_i^+ + b_i^- \leq M.$$

Hence

$$\text{if } (\bar{x}, \bar{b}^+, \bar{b}^-) \in A, \bar{x} \in B.$$

Therefore, the set of feasible x is the same for each set, so minimizing the objective function over one set will yield the same result as minimizing over the other.

4.3 Standard Constraint Formulation

MatLab's QuadProg package minimizes the function $\bar{x}^T H \bar{x} - \bar{f}^T \bar{x}$, subject to the constraints $A\bar{x} \leq \bar{b}$, and $Aeq\bar{x} = \bar{beq}$, where H , A , and Aeq are $k \times k$ square matrices, and \bar{f} , \bar{b} and \bar{beq} are $k \times 1$ vectors (k is the number of decision variables). In this project, QuadProg is run on the optimization problem:

$$\begin{aligned} \text{Minimize} \quad & \bar{x}^T \Omega \bar{x} - \frac{1}{\lambda} \bar{\mu}^T \bar{x} \\ \text{s.t.} \quad & x_i = b_i^+ - b_i^- \\ & b_i^+ \geq 0 \\ & b_i^- \geq 0 \\ & \sum_{i=1}^n b_i^+ + b_i^- \leq M \end{aligned}$$

$$\begin{aligned}
x_i &= y_i^+ - y_i^- \\
y_i^+ &\geq 0 \\
y_i^- &\geq 0 \\
\sum_{i=1}^n y_i^+ + y_i^- &\leq L \sum_{i=1}^n |x_i^{old}|
\end{aligned}$$

First, the matrices H , A , and Aeq need to be augmented to allow for the split constraint variables. These variables will be in the objective function, but they should not have any impact on the objective function value. Therefore, if n is the number of stocks in the portfolio, then augment H with $4n$ zeros in each row and column, so that it will be a square matrix with dimensions $5n \times 5n$. Similarly, augment \vec{f} with $4n$ zeros, so that it is a vector with dimensions $5n \times 1$.

The inequality and equality constraints need to be translated into matrix and vector form. The constraints are rearranged so that all decision variables are on the left side, and all inequalities are “less than or equal to”. The (rearranged) list of inequality constraints is: $-b_i^+ \leq 0$, $-b_i^- \leq 0$, $\sum_{i=1}^n b_i^+ + b_i^- \leq M$, $-y_i^+ \leq 0$, $-y_i^- \leq 0$, $\sum_{i=1}^n y_i^+ + y_i^- \leq L \sum_{i=1}^n |x_i^{old}|$. The list of equality constraints is: $-x_i + b_i^+ - b_i^- = 0$, $-x_i + y_i^+ - y_i^- = 0$. Now, the matrices A and Aeq , and the vectors \vec{b} and \vec{beq} can be defined. Let 0 denote an $n \times n$ matrix of zeros, I denote an $n \times n$ identity matrix, $\vec{0}$ denote an $n \times 1$ matrix of zeros, and $\vec{1}$ denote an $n \times 1$ matrix of ones.

$$A = \begin{bmatrix} 0 & -I & 0 & 0 & 0 \\ 0 & 0 & -I & 0 & 0 \\ 0 & 0 & 0 & -I & 0 \\ 0 & 0 & 0 & 0 & -I \\ \vec{0}^T & \vec{1}^T & \vec{1}^T & \vec{0}^T & \vec{0}^T \\ \vec{0}^T & \vec{0}^T & \vec{0}^T & \vec{1}^T & \vec{1}^T \end{bmatrix}, \vec{b} = \begin{bmatrix} \vec{0} \\ \vec{0} \\ \vec{0} \\ \vec{0} \\ M \\ L \sum_{i=1}^n |x_i^{old}| \end{bmatrix}$$

$$A_{eq} = \begin{bmatrix} -I & I & -I & 0 & 0 \\ -I & 0 & 0 & I & -I \end{bmatrix}, \vec{b}_{eq} = \begin{bmatrix} \vec{0} \\ \vec{xOld} \end{bmatrix}$$

Each column of A and A_{eq} corresponds to a vector of decision variables. The first corresponds to \vec{x} , the second to \vec{b}^+ , the third to \vec{b}^- , the fourth to \vec{y}^+ , and the fifth to \vec{y}^- . Also, every row corresponds to a constraint. With the constraints in a standard linear form, QuadProg can be used in the following way to perform portfolio optimization.

4.4 Description of Algorithm

Inputs:

λ = risk tolerance

L = turnover constraint parameter

M = booksize constraint

Choose which constraints to use (none, booksize only, turnover only, book-size and turnover)

Choose what to plot (portfolio weights, portfolio utility, portfolio value)

- Obtain the data (get risk matrices and return vectors for each time step).
- $numStocks \leftarrow$ the number of stocks in the portfolio
- $numTimePeriods \leftarrow$ the number of time periods for which to optimize
- $xOld \leftarrow$ equally weighted portfolio of $numStocks$ stocks

- for $i = 1$ to numTimePeriods
 - select appropriate μ and ω
 - run quadprog with linearized constraints to find x
 - $xOld \leftarrow x$
- Plot portfolio weights, utility, and value as requested

Chapter 5

Results

5.1 Data Management

5.1.1 Source of Data

The data used in this project was downloaded from Yahoo Finance. It consists of daily closing prices of stocks over an 80-day period. Four stocks were selected that are known to have low volatility: Gap, Lucent Technologies, Verizon Communications, and Disney. Also four stocks that are known for their high volatility were selected: DayStar Technologies, Salesforce.com, BriteSmile, and iMergent. For each sets of stocks, the daily returns were used to compute the $\vec{\mu}$ (average return) and Ω (risk) matrices.

5.1.2 Definition of Time Periods

The turnover constraint is implemented over several time periods. Therefore, to test the constraint several time periods worth of data were generated. Eleven $\vec{\mu}$ and eleven Ω matrices were calculated using 30 days worth of data for each period shifted five days. The optimization problem was solved across all of the time periods.

5.1.3 Verifying Validity of Optimization

To further verify the linearization of the constraints, the unconstrained problem was solved with a set of four stocks and compared to a loosely constrained problem on the same set of stocks. Constraints were chosen that would be satisfied by the solution to the unconstrained problem, i.e., allowing large book sizes and amounts of turnover between time periods. On each set of stocks, the solution to the unconstrained problem was identical to the solution of the loosely constrained problem. This provided further verification that the linearization of the constraints was valid.

5.2 Evolution of Portfolios

Figure 5.1 shows the evolution of a portfolio over 11 time periods with constraints on book size and turnover. The algorithm defaulted to an equally-weighted portfolio in the first time period. In this example, the book size is 10.

There are several important features about the dynamics of the portfolio shown by Figure 5.1. The constraint on book size limits portfolio book size less than or equal to a particular value. Between the first two time periods the book size actually decreases in value. By the final time step our book size has decreased in value from 10 to 7. Between each time step, there is a certain amount of turnover conducted by the algorithm. As the portfolio evolves through time steps, the distribution in weights of the portfolio becomes increasingly different from the original portfolio. Finally, the weight of dark blue stock in

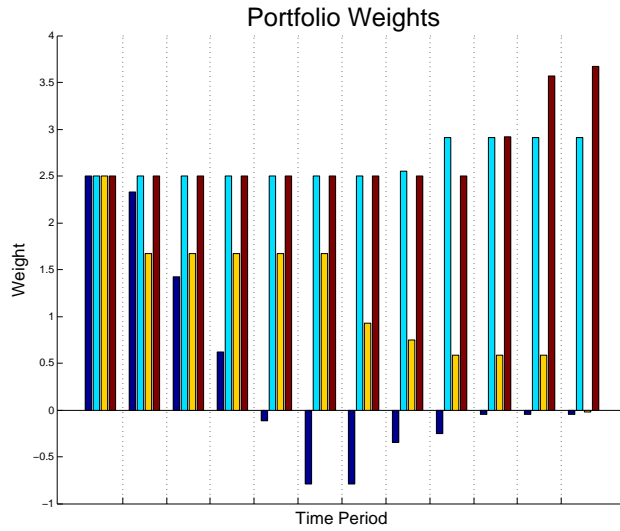


Figure 5.1: Evolution of Portfolio Weights

the middle time periods become negative values, reiterating the allowance of short-selling.

5.3 Portfolio Value Calculations

Portfolio value is defined as the amount of money an investor would have if he sold all his stocks and closed all his short positions plus all of his earnings. Let $w_i(t)$ represent the price of stock i at time t , $w_i x_i$ be the current wealth of stock i , MMA equal the amount of money in bank, $\Delta MMA_i(t)$ symbolize the

change in balance from time $t - 1$ to time t , and $\text{Value}(t)$ be the portfolio value.

At time t , the change in balance caused by stock i is going to equal the change in weights from time $t - 1$ to time t multiplied by the price of stock i at time t . More generally, the overall change will equal the dot product of the vector of prices and the vector of change in weights.

$$\Delta MMA(t) = \vec{w}^T(t) [\vec{x}(t) - \vec{x}(t - 1)] \quad (5.1)$$

The current balance MMA will equal the sum of all changes in balance. Portfolio value will equal the current balance plus the present wealth of the portfolio.

$$\text{Value}(t) = \vec{w}^T(t)\vec{x}(t) + MMA = \sum_{i=1}^n (w_i(t)x_i(t)) + MMA \quad (5.2)$$

5.4 Comparing Value and Utility Plots

5.4.1 Efficient vs. Equally Weighted Plots

To evaluate the performance of our portfolio, it was compared to a portfolio that was equally weighted at each time step. In Figure 5.2 the efficient portfolio either matched or exceeded the equally-weighted portfolio in actual performance. As shown in Figure 5.3, the efficient portfolio easily outperformed the equally-weighted portfolio in expected utility.

The next set of graphs are explorations of the effects of different allowances of booksize and turnover.

5.4.2 Booksize Plots

First, the algorithm was run for different allowances of booksize. Figure 5.4 illustrates the principle that generally relaxing the booksize allows for greater

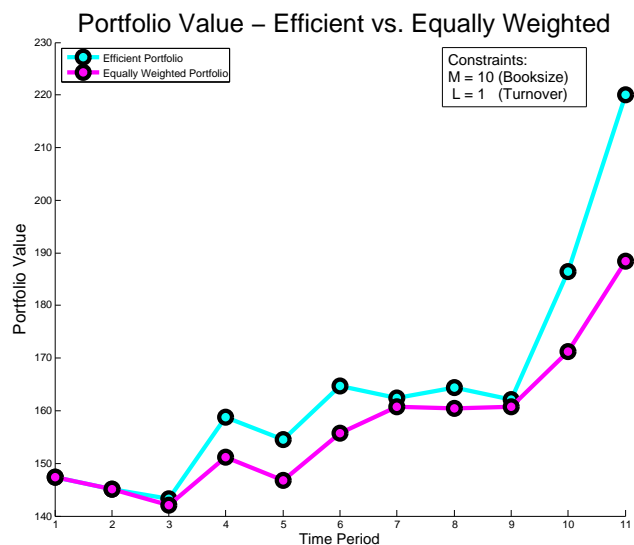


Figure 5.2: Portfolio Value: Efficient vs. Equally Weighted

return (and as a result portfolio value).

Figure 5.5 shows the expected utility of the same portfolios. The results are counterintuitive. A portfolio with a loosely constrained booksize has a greater set of feasible portfolios than a portfolio with a tightly constrained booksize. Actually, the smaller set is a subset of the larger set. However, since there is a constraint on turnover, these sets no longer necessarily intersect. As a result, it is possible to obtain a better utility with a more tightly constrained booksize.

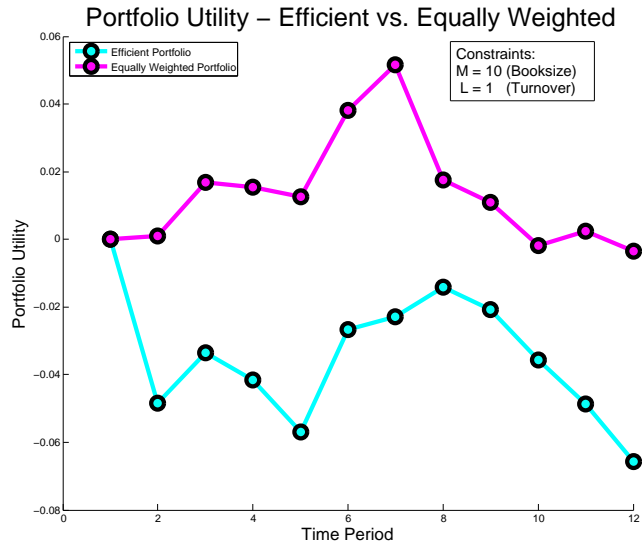


Figure 5.3: Portfolio Utility: Efficient vs. Equally Weighted

5.4.3 Turnover Plots

Next the algorithm was run for different allowances of turnover (Figure 5.6). Judging by the latter time periods, it appears that allowing larger amounts in turnover is a great idea. These results are deceiving as the calculation of portfolio value does not consider transaction costs.

Figure 5.7 shows the expected utility of the previous portfolios. This graph may also seem counterintuitive, similar to the utility plot of different booksizes.

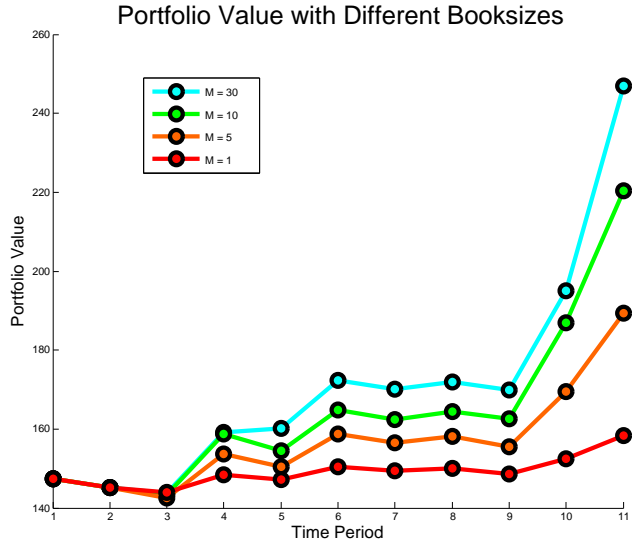


Figure 5.4: Portfolio Value for Different Booksizes

It is important to note that similar utility does not imply similar portfolio. In the first time step, it is possible for a loosely constrained portfolio to make an enormous change in its distribution of weights, while a tightly constrained portfolio will not. As a result, in the second time step the tightly constrained portfolio might be closer to achieve a portfolio with good utility. In fact, the loosely constrained portfolio may only be able to achieve a portfolio with poor utility. As a result, the tightly constrained portfolio may outperform the loosely constrained one. Even without considering transaction costs, allowing large

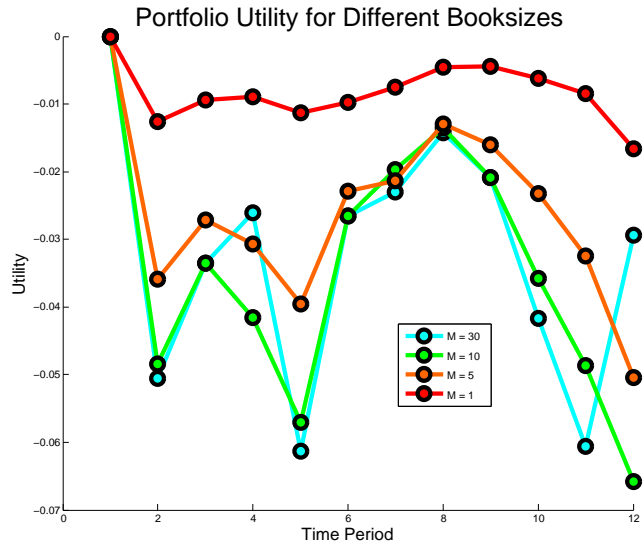


Figure 5.5: Portfolio Utility for Different Booksizes

amounts of turnover is a bad idea since your portfolio may not be able to consistently achieve good utility.

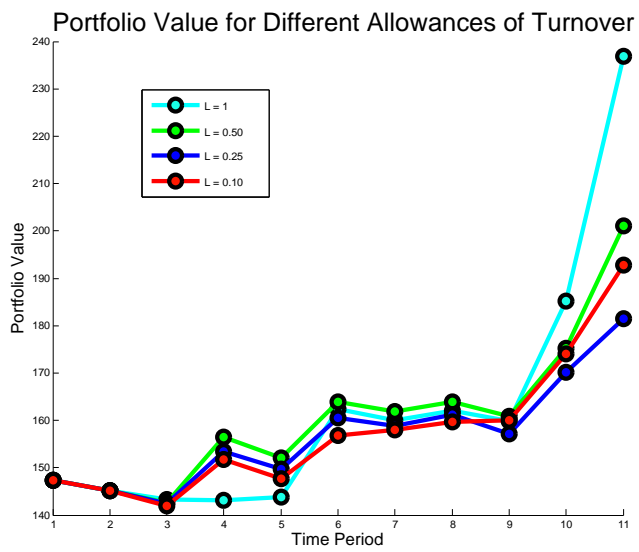


Figure 5.6: Portfolio Value for Different Allowances of Turnover

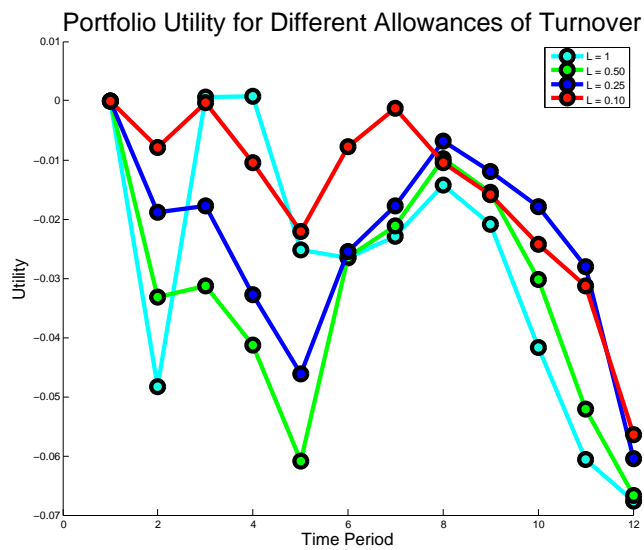


Figure 5.7: Portfolio Utility for Different Allowances of Turnover

Chapter 6

Explorations

Once the turnover and booksize constraints were implemented, it was possible to extend the idea of constraining turnover to limit transaction costs in two more ways. The first, called *turnover budgeting*, involves dynamically implementing turnover at different time steps. The second, called *threshold trading*, involves only trading if the expected change in utility is large enough.

6.1 Turnover Budgeting

With turnover budgeting, the basic idea is that a different amount of turnover is allowed at each time step. This translates to the parameter L in the turnover constraint being different at different times. One way to demonstrate this idea is to consider a set of eleven time periods. If L is initially set to .2, then the resulting portfolios can be seen in Figure 6.1.

The total amount of turnover for this set of time periods is 2.2, which means that the turnover constraint is tight at every time period. Changing the parameter L to a vector \vec{L} , where L_j is the amount of turnover allowed at time period

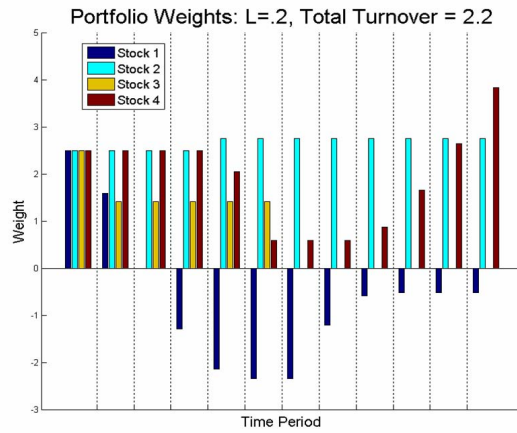


Figure 6.1: Non Turnover Budgeted Portfolios

j allows a dynamic turnover implementation. Particularly, if the total turnover is kept the same, then valid comparisons between the turnover budgeted and non-turnover budgeted portfolios can be made. In Figure 6.2, the total turnover was kept at 2.2, but the vector \vec{L} was set to $[\cdot 1, \cdot 1, \dots, 1.2]$. The fact that the turnover budgeted portfolios had better utility and value, as seen in Figure 6.3 than the non-turnover budgeted portfolios shows that turnover budgeting can be a good idea. The interesting problem that arises with turnover budgeting is when is the optimal time to change L , based on covariance and risk matrices.

6.2 Threshold Trading

If investors are only using turnover constraints to limit their transaction costs, a problem may arise. The turnover constrained portfolios may involve very small changes, gaining very small changes in utility. These small changes may still cost a significant amount of money (as there is often a fixed transaction

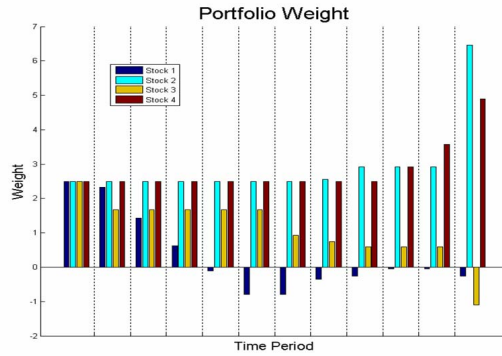


Figure 6.2: Turnover Budgeted Portfolios

cost associated with trades). A possible remedy to this situation is to require that the expected change in utility is less than some parameter $P < 0$ (since utility is being minimized). In other words, run the optimization, and if the best portfolio that can be achieved in the feasible region doesn't lower the utility by enough, don't make the trade. The equation used for the threshold criteria is if

$$\frac{(\vec{x}^T \Omega \vec{x} - \vec{\mu}^T \vec{x}) - (\vec{x}_{old}^T \Omega \vec{x}_{old} - \vec{\mu}^T \vec{x}_{old})}{|\vec{x}_{old}^T \Omega \vec{x}_{old} - \vec{\mu}^T \vec{x}_{old}|} < P < 0 \quad (6.1)$$

then make the trade. Figure 6.4 is an example of a set of time periods where the portfolios have been optimized using threshold trading.

As can be seen in Figure 6.5, this may not be the best idea, since the value and expected utility are both worse for threshold trading portfolios. This was the case for all examples tried.

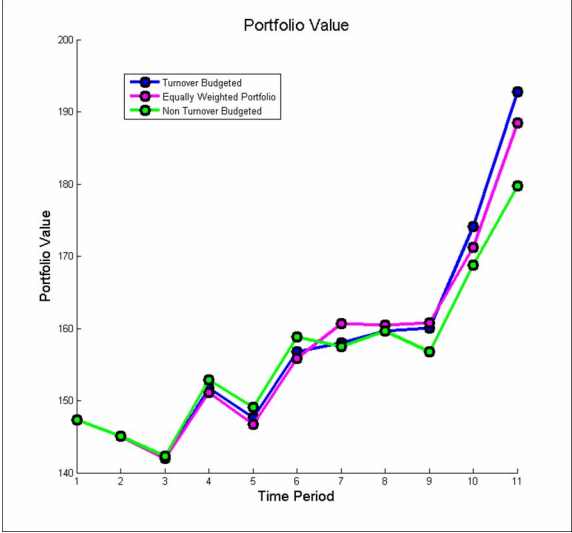


Figure 6.3: Turnover Budgeted Values

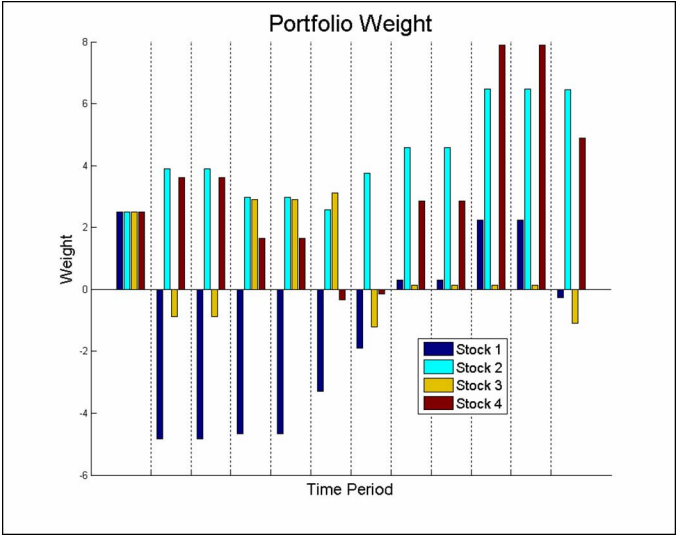


Figure 6.4: Threshold Traded Portfolios

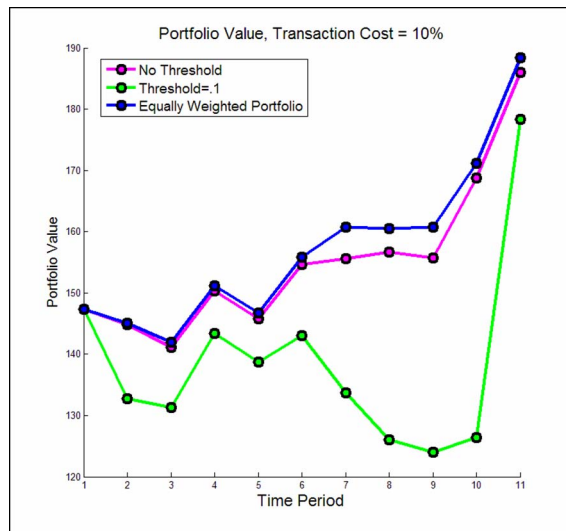


Figure 6.5: Threshold Traded Values

Chapter 7

Conclusion

This portfolio efficiently implemented multi-period portfolio optimization with non-smooth constraints on booksize and turnover. To use a standard quadratic programming package, these constraints were linearized using split variables.

After running the algorithm, the resulting portfolios' actual values and expected utilities were compared to those of an equally-weighted portfolio. The optimal portfolio had consistently better performance. Next, the algorithm was run on portfolios with constant allowances of turnover but different booksizes. This showed that relaxing the booksize constraint allows for higher returns. The algorithm was also run on portfolios with constant allowances of booksize but different constraints on turnover. The resulting portfolios demonstrated that relaxing turnover does not necessarily lead to higher returns and expected utility. Making a huge change in one time step could prevent moving to a portfolio with a good utility.

Turnover budgeting implies dynamic allowance of turnover. Adding

turnover budgeting yielded better expected utility and portfolio value on the examples tested. To further extend the idea of turnover budgeting, optimal turnover distribution would have to be considered. Threshold trading involves choosing whether or not to trade based on expected change in utility. On the examples tried, threshold trading consistently performed worse than benchmark portfolios.

This work could be extended in several ways. The ideas of turnover budgeting and threshold trading could be further developed to more accurately model trading with transaction costs. Also, the algorithm could be tested on larger sets of stocks. The efficiency of the algorithm would need to be improved so that it could be used on a realistic portfolio (consisting of about 5000 stocks).

Chapter 8

Appendices

8.1 Appendix A: MatLab Code

%optifolio takes all the inputs and calculates optimal portfolios

lambda=0; %if lambda=0, then vary lambda to generate efficient frontiers

L = .1; %turnover

M = 30; %booksize

ConstraintOption = 3; %which constraints to use

PlotOption = 2; % what to plot

ExitFlagOption = 1; %1 if you want exitflags, 0 if not

PeriodLength = 5; %the amount of shift between periods

Window = 30; %the number of trading days per period

```
[Xmat]=PlotPortfolios(lambda,L,M,ConstraintOption,PlotOption,ExitFlagOption,PeriodLength,
Window);
```

```
%*****
function [Xmat] = PlotPortfolios(lambda,L,M,Choice,PlotOption,ExitFlagOption,PeriodLength,
Window)
%takes as input lambda (risk tolerance), L (amount of turnover allowed),
%M(booksize), and Choice, which is an input that decides which constraints will be
enforced
% if choice = 0, unconstrained
% if choice = 1, booksize only
% if choice = 2, turnover only
% if choice = 3, booksize and turnover
```

%compute the matrix of asset weights and turnover amounts by running

%Turnover123

if lambda == 0

```
    Xmat = PlotPortfoliosLam(L,M,Choice,PlotOption,ExitFlagOption,PeriodLength,Window);
```

else

```
[TurnoverMatrix,exitflagmatrix,BooksizeMat,Xmat,ClosingPrices,BigMu,BigOmega,NumStocks,
NumTimePeriods]=CalculateTurnover(lambda,L,M,Choice,PeriodLength,Window);
```

Xmat

TurnoverMatrix

```
%%FinalTurn=[0;0;TurnoverMatrix];
```

```
Xonly=Xmat(1:NumStocks,1:NumTimePeriods+1)
```

```
%%Xmat=[Xmat;FinalTurn']
```

```
Xmat=[Xmat;TurnoverMatrix']
```

```
stock=zeros(NumTimePeriods,NumStocks);
```

```
stockE=zeros(NumTimePeriods,NumStocks);
```

```
deltaMMA=zeros(NumTimePeriods,NumStocks);
```

```
deltaMMAE=zeros(NumTimePeriods,NumStocks);
```

```
MMA=zeros(NumTimePeriods,NumStocks);
```

```
MMAE=zeros(NumTimePeriods,NumStocks);
```

```
portfolio=zeros(NumTimePeriods,NumStocks);
```

```
portfolioE=zeros(NumTimePeriods,NumStocks);
```

```
Xequal=10/NumStocks*ones(NumStocks,NumTimePeriods+1);
```

```

for i=1:NumStocks
    for k=1:NumTimePeriods
        stock(k,i)=Xonly(i,k)*ClosingPrices(k,i); %find the amount invested in stock at
each time step
        stockE(k,i)=Xequal(i,k)*ClosingPrices(k,i); %same for equally weighted

    end

    for k=2:NumTimePeriods
        deltaMMA(k,i)=-(Xonly(i,k)-Xonly(i,k-1))*ClosingPrices(k,i); %find the change in
the MMA at each time step
        deltaMMAE(k,i)=-(Xequal(i,k)-Xequal(i,k-1))*ClosingPrices(k,i); %same for equally
weighted
    end

    for k=2:NumTimePeriods
        MMA(k,i)=sum(deltaMMA(1:k,i)); %find the value of the MMA
        MMAE(k,i)=sum(deltaMMAE(1:k,i)); %same for equally weighted
    end

    for k=1:NumTimePeriods
        portfolio(k,i)=stock(k,i)+MMA(k,i); %find the portfolio value
        portfolioE(k,i)=stockE(k,i)+MMAE(k,i); %same for equally weighted
    end

end

totalportfolio=zeros(NumTimePeriods,1);
totalportfolioE=zeros(NumTimePeriods,1);

for i=1:NumTimePeriods
    totalportfolio(i)=sum(portfolio(i,:));
    totalportfolioE(i)=sum(portfolioE(i,:));
end

Utility=[];
UtilityEq=[];
BigOmega=[zeros(NumStocks,NumStocks);BigOmega];
BigMu=[zeros(1,NumStocks);BigMu];

for i=1:NumTimePeriods+1
    Utility=[Utility;Xonly(:,i)']*BigOmega((i-1)*NumStocks+1:(i-1)*NumStocks+NumStocks,:);
*Xonly(:,i)-1/lambda*BigMu(i,:)*Xonly(:,i)];
    UtilityEq=[UtilityEq;Xequal(:,i)']*BigOmega((i-1)*NumStocks+1:(i-1)*
*NumStocks+NumStocks,:)*Xequal(:,i)-1/lambda*BigMu(i,:)*Xequal(:,i)];
end

```

```
if PlotOption==1 %plot turnover, booksize, and weights
    TimePlot = [];
    for i=1:NumTimePeriods+1
        TimePlot=[TimePlot,i];
    end
    hold on
    scatter(TimePlot,TurnoverMatrix,'^','m') %plot the turnover at each time step
    scatter(TimePlot,BooksizeMat,'p','g')
    TurnoverMatrix
    BooksizeMat
    bar(Xmat(1:NumStocks,:),'group') %on the same graph, plot the portfolio weights at
each time step
    xlabel('Time Period')
    ylabel('Weight')
    title('Portfolio Weight, Booksize, Turnover')
    legend('Turnover','Booksize','Stock 1', 'Stock 2', 'Stock 3', 'Stock 4',6);
    hold off
elseif PlotOption==2 %plot return of portfolio vs. equally weighted
    TimePlot = [];
    for i=1:NumTimePeriods
        TimePlot=[TimePlot,i];
    end
    hold on
    scatter(TimePlot, totalportfolio,'p')
    scatter(TimePlot, totalportfolioE,'o')
    xlabel('Time Period')
    ylabel('Return')
    title('Portfolio Returns')
    legend('Optifolio','Equally Weighted Portfolio',2);
    hold off
elseif PlotOption==3 %plot utility of portfolio vs. equally weighted
    TimePlot = [];
    for i=1:NumTimePeriods+1
        TimePlot=[TimePlot,i];
    end

    figure
    hold on
    scatter(TimePlot, Utility,'p')
    scatter(TimePlot, UtilityEq,'o')
    xlabel('Time Period')
    ylabel('Utility')
    title('Portfolio Utilities')
    legend('Optifolio','Equally Weighted Portfolio',2);
    hold off
end

if ExitFlagOption ==1
    exitflagmatrix
end
```

```

end

end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [EFMat] = PlotPortfoliosLam(L,M,Choice,PlotOption,ExitFlagOption,PeriodLength,
Window)
%takes as input L (amount of turnover allowed), M(booksize),
%and Choice, which is an input that decides which constraints will be enforced
% if choice = 0, unconstrained
% if choice = 1, booksize only
% if choice = 2, turnover only
% if choice = 3, booksize and turnover

[EFMat1,NumStocks,NumTimePeriods]=CalculateTurnoverLam(10,M,Choice,PeriodLength,Window);
[EFMat2,NumStocks,NumTimePeriods]=CalculateTurnoverLam(1,M,Choice,PeriodLength,Window);
[EFMat3,NumStocks,NumTimePeriods]=CalculateTurnoverLam(.1,M,Choice,PeriodLength,Window);
[EFMat4,NumStocks,NumTimePeriods]=CalculateTurnoverLam(.01,M,Choice,PeriodLength,Window);

% for i=1:2:NumTimePeriods*2
%     figure
%     scatter(EFMat1(:,i),EFMat1(:,i+1));
% end
%
% figure
% for i=1:2:NumTimePeriods*2
%     hold on
%     scatter(EFMat1(:,i),EFMat1(:,i+1));
% end

figure
hold on
scatter(EFMat1(:,3),EFMat1(:,4),'p');
scatter(EFMat2(:,3),EFMat2(:,4),'o');
scatter(EFMat3(:,3),EFMat3(:,4),'^');
scatter(EFMat4(:,3),EFMat4(:,4),'s');

xlabel('Risk')
ylabel('Return')
title('Efficient Frontiers: Time Period = 2, M=30')
legend('L=10','L=1','L=.1','L=.01',4);

EFMat = EFMat1;

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [TurnMatrix,Exitflags,BooksizeMat,Xmat,StockPrices,BigMu,BigOmega,NumStocks,
NumTimePeriods] = CalculateTurnover(lambda,L,M,Choice,PeriodLength,Window)

```

```

[BigMu,BigOmega,StockPrices, NumStocks, NumTimePeriods]=GetData(PeriodLength,Window); %run
data to get the risk and return matrices at each time step

%TimePeriods = size(BigMu,1); %figure out the number of time steps

%Weights=[];
Utilities=zeros(NumTimePeriods,1);
Exitflags=zeros(NumTimePeriods,1);

xold=[10/NumStocks*ones(NumStocks,1);zeros(4*NumStocks,1)];
Xmat=[xold(1:NumStocks);0;0];
Weights=[xold(1:NumStocks)'];
%LMat=[.01*ones(NumTimePeriods-1,1);50]; %if you want to do some sort of
%risk-budgeting thing
for i=1:NumTimePeriods
    CurrentMu=BigMu(i,:); %select the return vector for the current time step
    CurrentOmega=BigOmega((i-1)*NumStocks+1:(i-1)*NumStocks+NumStocks,:); %select the risk
matrix for the current time step
    %[x,fval,exitflag,portrisk,portreturn]=RunOptimizer(CurrentMu,CurrentOmega,lambda,
xold,LMat(i),M,Choice,NumStocks,NumTimePeriods); %risk
    %budgeted
    [x,fval,exitflag,portrisk,portreturn]=RunOptimizer(CurrentMu,CurrentOmega,lambda,xold,
L,M,Choice,NumStocks,NumTimePeriods); %get the portfolio weights, risk, and return by
running TrackTurnover123

    Weights=[Weights;x(1:NumStocks)']; %matrix of asset weights in the portfolio
    Utilities(i)=fval;
    Exitflags(i)=exitflag;
    xold=x; %reset the old x to be the current x, for use in the next time step
    Xmat=[Xmat,[xold(1:NumStocks);portrisk;portreturn]]; %input the current portfolio
weights, risk, and return into Xmat
end

TurnMatrix=zeros(NumTimePeriods+1,1); %make a matrix of the amount of turnover at each
time period

for i=2:NumTimePeriods+1
    TurnMatrix(i)=sum(abs(Weights(i-1,:)-Weights(i,:))); %amt of turnover = sum(abs(x_i^{
new}-x_i^{old}))
end

BooksizeMat = zeros(NumTimePeriods+1,1); %make a matrix of the booksize at each time
period

for i=1:NumTimePeriods+1
    BooksizeMat(i)=sum(abs(Weights(i,:))); %booksize = sum(abs(x_i))
end

```

```

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [EFMat,NumStocks,NumTimePeriods] = CalculateTurnoverLam(L,M,Choice,PeriodLength,Window)

[BigMu,BigOmega,StockPrices, NumStocks, NumTimePeriods]=GetData(PeriodLength,Window); %run data to get the risk and return matrices at each time step
NumTimePeriods=2;

EFMat = [];

xMatold=[10/NumStocks*ones(NumStocks,100);zeros(4*NumStocks,100)];
for i=1:NumTimePeriods
    CurrentMu=BigMu(i,:); %select the return vector for the current time step
    CurrentOmega=BigOmega((i-1)*NumStocks+1:(i-1)*NumStocks+NumStocks,:); %select the risk matrix for the current time step
    EFMati=[];
    XMati=[];
    count=0;
    for lambda = .1:.1:10
        count=count+1;
        xold = xMatold(:,count);
        [x,fval,exitflag,portrisk,portreturn]=RunOptimizer(CurrentMu,CurrentOmega,lambda,xold,L,M,Choice,NumStocks,NumTimePeriods); %get the portfolio weights, risk, and return by running TrackTurnover123
        EFMati = [EFMati;portrisk,portreturn];
        XMati=[XMati,x];
    end
    EFMat = [EFMat,EFMati];
    xMatold=XMati; %reset the old x to be the current x, for use in the next time step
end

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [mutrans, omega, P, NumStocks, NumTimePeriods] = GetData(PeriodLength,Window) % last edit 2/15
%Assume that ((NumDays-(Window+1))/PeriodLength) is an integer

M = csvread('HVClosel18Two.csv',1,0); %M becomes matrix of excel data
[NumDays,NumStocks] = size(M);
NumTimePeriods = ((NumDays - (Window + 1))/PeriodLength) + 1;

%Maybe Truncate Excel Matrix

```

```

mutrans=[]; %initialize mu matrix
omega=[]; %initialize omega matrix
N = zeros(NumDays,NumStocks); %inverted excel data
P = zeros(NumTimePeriods,NumStocks); %vector of stock prices for each time period
R = zeros(NumDays - 1,NumStocks); %matrix of returns
r = zeros(Window,NumStocks); %inititalize vector of returns

for i = 1:NumDays
    for j = 1:NumStocks
        N(i,j) = M((NumDays+1)-i,j); %becomes inverted M
    end
end

for i = 1:NumTimePeriods
    for j = 1:NumStocks
        P(i,j) = N(PeriodLength*(i-1)+(Window +1),j); %populate vector of stock prices
    end
end

for i = 1:(NumDays - 1)
    for j = 1:NumStocks
        R(i,j) = (N(i+1,j) - N(i,j))/N(i,j); %populate matrix of returns
    end
end

for k = 1:NumTimePeriods %k-1 refers to time period
    for i = 1:Window %i refers to trading day
        for j = 1:NumStocks %j refers to a particular asset
            r(i,j) = R((PeriodLength*(k-1))+i,j); %r(i,j) equals the return of asset j at
trading day (5*(k-1)+i)

            end
        end

        mutrans = [mutrans;mean(r)]; %auguments mean vector transpose at time period k-1 to
mutrans
        omega = [omega;cov(r)]; %augments covariance matrix at time period k-1 to omega
    end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [x,fval,exitflag,portrisk,portreturn] = RunOptimizer(mu,omega,lambda,xold,L,M,
Choice,NumStocks,NumTimePeriods)
warning off all
% runs quadprog

%options = optimset('MaxIter',10000,'TolPCG',.0001);

if Choice==0 %fully invested only

```

```

%Aeq=[1,1,1,1];
%beq=1; % Aeq*x=beq ==> sum(x_i)=1
[x,fval,exitflag] = quadprog(2*omega,-1/lambda*mu); %min risk-1/lambda*return s.t.
Aeqx=beq
portrisk=x'*omega*x; %calculate portfolio risk and return
portreturn=mu*x;

elseif Choice==1 %booksize

    omega=[omega,zeros(NumStocks,NumStocks*2);zeros(NumStocks*2,NumStocks*3)]; % augment
omega for 4 positive and 4 negative split variables (b_i^+ and b_i^-)
    mu=[mu,zeros(1,NumStocks*2)]; % augment mu for 4 positive and 4 negative split
variables

    %some new constraints
    %A=[zeros(4,4),-eye(4,4),zeros(4,4);zeros(4,8),-eye(4,4);eye(4,4),-eye(4,4),zeros
(4,4);-eye(4,4),zeros(4,4),-eye(4,4);zeros(1,4),ones(1,8)];
    %b=[zeros(16,1);M]; % Ax<=b ==> b_i^+ >=0, b_i^- >=0, sum(b_i^+ + b_i^-
%<= M (booksize)

    A=[zeros(NumStocks,NumStocks),-eye(NumStocks,NumStocks),zeros(NumStocks,NumStocks);
zeros(NumStocks,NumStocks*2),-eye(NumStocks,NumStocks);zeros(1,NumStocks),ones(1,
NumStocks*2)]; %old constraints ONLY
    b=[zeros(NumStocks*2,1);M]; % Ax<=b ==> b_i^+ >=0, b_i^- >=0, sum(b_i^+ + b_i^-

    Aeq=[-eye(NumStocks,NumStocks),+eye(NumStocks,NumStocks),-eye(NumStocks,NumStocks)];
    beq=[zeros(NumStocks,1)];

    % Aeqx=beq ==> b_i^+ - b_i^- =x_i , sum(x_i = 1)

    %[x,fval,exitflag]=quadprog(2*omega,-1/lambda*mu,A,b,Aeq,beq,[],[],[],options); %min
risk-1/lambda*return s.t. Ax<=b, Aeqx=beq
    [x,fval,exitflag]=quadprog(2*omega,-1/lambda*mu,A,b,Aeq,beq,[],[],xold(1:
NumStocks*3));
    portrisk=x'*omega*x; %calculate portfolio risk and return
    portreturn=mu*x;

elseif Choice==2 %turnover

    omega=[omega,zeros(NumStocks,NumStocks*2);zeros(NumStocks*2,NumStocks*3)]; %augment
omega for 4 positive (y_i^+) and 4 negative (y_i^-) split variables
    mu=[mu,zeros(1,NumStocks*2)]; %augment mu

    A=[zeros(NumStocks,NumStocks),-eye(NumStocks,NumStocks),zeros(NumStocks,NumStocks);
zeros(NumStocks,NumStocks*2),-eye(NumStocks,NumStocks);zeros(1,NumStocks),ones(1,
NumStocks*2)];
    b=[zeros(NumStocks*2,1);L*sum(abs(xold(1:NumStocks)))];

```



```

% Ax<=b ==> y_i^+ >=0, y_i^- >=0, sum(y_i^+ + y_i^- <= L*sum(old x's)

Aeq=[-eye(NumStocks,NumStocks),eye(NumStocks,NumStocks),-eye(NumStocks,NumStocks)];
beq=[-xold(1:NumStocks)];

% Aeqx=beq ==> y_i^+ - y_i^- =x_i - x_i^{old}, sum(x_i = 1)

%[x,fval,exitflag]=quadprog(2*omega,-1/lambda*mu,A,b,Aeq,beq,[],[],[],options); %min
risk-1/lambda*return s.t. Ax<=b, Aeqx=beq
[x,fval,exitflag]=quadprog(2*omega,-1/lambda*mu,A,b,Aeq,beq,[],[],xold(1:
NumStocks*3));
portrisk=x'*omega*x; %calculate portfolio risk and return
portreturn=mu*x;

elseif Choice==3 %turnover, booksize
omega=[omega,zeros(NumStocks,NumStocks*4);zeros(NumStocks*2,NumStocks*5);zeros
(NumStocks*2,NumStocks*5)]; %augment omega w/ y_i^+,y_i^-,b_i^+,b_i^-
mu=[mu,zeros(1,NumStocks*2),zeros(1,NumStocks*2)]; %augment mu

%new stuff
% A=[zeros(4,4),-eye(4,4),zeros(4,12); %-y_i^+
% zeros(4,8),-eye(4,4),zeros(4,8); %-y_i^-
% zeros(4,12),-eye(4,4),zeros(4,4); %-b_i^+
% zeros(4,16),-eye(4,4); %-b_i^-
% zeros(1,4),ones(1,8),zeros(1,8); %sum(y_i^+ +y_i^-)
% zeros(1,12),ones(1,8); %sum(b_i^+ +b_i^-)
% eye(4,4),-eye(4,4),zeros(4,4),zeros(4,4),zeros(4,4);
% -eye(4,4),zeros(4,4),eye(4,4),zeros(4,4),zeros(4,4);
% eye(4,4),zeros(4,4),zeros(4,4),-eye(4,4),zeros(4,4)
% -eye(4,4),zeros(4,4),zeros(4,4),zeros(4,4),eye(4,4)];
%
% b=[zeros(16,1);L*sum(abs(xold(1:4)));M;xold(1:4);-xold(1:4);zeros(4,1);zeros(4,1)];
%16 zeros, L*sum(old x's), booksize

A=[zeros(NumStocks,NumStocks),-eye(NumStocks,NumStocks),zeros(NumStocks,NumStocks*3);
%-y_i^+
zeros(NumStocks,NumStocks*2),-eye(NumStocks,NumStocks),zeros(NumStocks,
NumStocks*2); %-y_i^-
zeros(NumStocks,NumStocks*3),-eye(NumStocks,NumStocks),zeros(NumStocks,NumStocks)];
%-b_i^+
zeros(NumStocks,NumStocks*4),-eye(NumStocks,NumStocks); %-b_i^-
zeros(1,NumStocks),ones(1,NumStocks*2),zeros(1,NumStocks*2); %sum(y_i^+ +y_i^-)
zeros(1,NumStocks*3),ones(1,NumStocks*2)]; %sum(b_i^+ +b_i^-)

b=[zeros(NumStocks*4,1);L*sum(abs(xold(1:NumStocks)));M]; %16 zeros, L*sum(old x's),
booksize

% Ax<=b ==>
% y_i^+ >= 0
% y_i^- >= 0
% b_i^+ >= 0

```

```

% b_i^- >= 0
% sum(y_i^+ + y_i^-) <= L*sum(old x's)
% sum(b_i^+ + b_i^-) <= M

Aeq=[ -eye(NumStocks,NumStocks), eye(NumStocks,NumStocks), -eye(NumStocks,NumStocks), ↵
zeros(NumStocks,NumStocks*2); %-x_i + y_i^+ - y_i^-
-eye(NumStocks,NumStocks), zeros(NumStocks,NumStocks*2), eye(NumStocks,NumStocks), - ↵
eye(NumStocks,NumStocks)];

beq=[ -xold(1:NumStocks); zeros(NumStocks,1)];

%Aeqx=beq ==>
% y_i^+ - y_i^- = x_i - x_i^{old}
% b_i^+ - b_i^- = x_i

%[x,fval,exitflag]=quadprog(2*omega,-1/lambda*mu,A,b,Aeq,beq,[],[],[],options);%min ↵
risk-1/lambda*return s.t. Ax<=b, Aeqx=beq
[x,fval,exitflag]=quadprog(2*omega,-1/lambda*mu,A,b,Aeq,beq,[],[],xold);
%[x,fval,exitflag]=fmincon(@myfun(x,omega,mu,lambda),xold,A,b,Aeq,beq);
portrisk=x'*omega*x;%calculate portfolio risk and return
portreturn=mu*x;

end

end

```

8.2 Appendix B: PowerPoint Presentation

Portfolio Optimization with Transaction Costs

Jessica Clark and Sean Mulready
optifolio@wpi.edu



Advisor: Dr. Arthur C. Heinricher

Project Goals

- Implement multi-period portfolio optimization for non-smooth constraints
- Explore the effects of transaction constraints on portfolio value and utility

Portfolio Optimization

- Investment Strategy: Invest in each of n assets
- x_i = number of shares held of asset i
- Amount invested in each asset can be positive (buying long), zero, or negative (selling short)
- Goal: minimize risk for a given return

3

Worcester Polytechnic Institute

Risk and Return

$$\text{Expected Portfolio Risk} = \bar{x}^T \Omega \bar{x} = \sum_{i,j} x_i \sigma_{i,j} x_j$$

where $\sigma_{i,j}$ is the covariance between stock i and stock j

$$\text{Expected Portfolio Return} = \bar{\mu}^T \bar{x} = \sum_i \mu_i x_i$$

where μ_i is the return for stock i

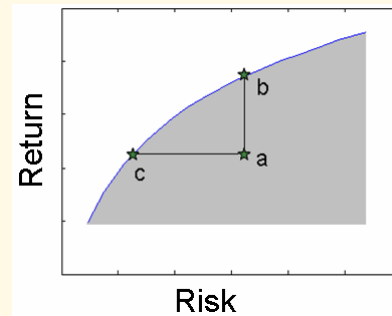
Reference: Markowitz, H. M. (1952). Portfolio Selection,
Journal of Finance, Vol. 7, Iss. 1, p. 77-91.

4

Worcester Polytechnic Institute

Efficient Frontiers

- An efficient portfolio has the lowest risk for a given return, as well as the highest return for that amount of risk
- An efficient frontier is the curve made up of all efficient portfolios



5

Classical Optimization Problem

- Minimize risk subject to return
- Vary λ to get an efficient frontier

$$\begin{aligned} \min \bar{x}^T \Omega \bar{x} - \frac{1}{\lambda} \bar{\mu}^T \bar{x} & \quad (\text{minimize utility}) \\ \text{s.t. } \sum x_i = F & \quad (\text{fully invested constraint}) \end{aligned}$$

6

Booksize vs. Wealth

Time 0

Stock	Price	Shares (x_i)	Action	Net Bank
1	\$10		Buy 10	
2	\$10		Sell Short 10	

7

Worcester Polytechnic Institute

Booksize vs. Wealth

Time 0

Stock	Price	Shares (x_i)	Action	Net Bank
1	\$10	10	Buy 10	-\$100
2	\$10	-10	Sell Short 10	+\$100

$$\text{Booksize} = |10| + |-10| = 20$$

$$\text{Wealth} = (-\$100) + (\$100) = \$0$$

8

Worcester Polytechnic Institute

Booksize vs. Wealth

Time 1

Stock	Price	Shares (x_i)	Action	Net Bank
1	\$5	10		-\$100
2	\$15	-10		+\$100

$$\text{Booksize} = |10| + |-10| = 20$$

$$\text{Wealth} = (-\$100) + (\$100) = \$0$$

9

Worcester Polytechnic Institute

Booksize vs. Wealth

Time 1

Stock	Price	Shares (x_i)	Action	Net Bank
1	\$5	10	Sell 10	-\$100
2	\$15	-10	Close Position	+\$100

$$\text{Booksize} = |10| + |-10| = 20$$

$$\text{Wealth} = (-\$100) + (\$100) = \$0$$

10

Worcester Polytechnic Institute

Booksize vs. Wealth

Time 1

Stock	Price	Shares (x_i)	Action	Net Bank
1	\$5	0	Sell 10	-\$100 +\$50
2	\$15	0	Buy 10	+\$100 -\$150

11

Worcester Polytechnic Institute

Booksize vs. Wealth

Time 1

Stock	Price	Shares (x_i)	Action	Net Bank
1	\$5	0	Sell 10	-\$50
2	\$15	0	Buy 10	-\$50

$$Wealth = (-\$50) + (-\$50) = -\$100$$

12

Worcester Polytechnic Institute

Transaction Costs

Difficult to model because many factors involved:

- Commission
- Bid/Ask spread
- Market impact

Ref:

Grinold, Richard and Kahn, Ronald. (2000). Active Portfolio Management.
Scherer, Bernd. (2002). Portfolio Construction and Risk Budgeting. Risk Waters
Group, London.

13

Worcester Polytechnic Institute

Turnover Constraint

- Limit turnover \Rightarrow Limit transaction costs
- Limits the total amount of change in the portfolio at each time step

$$\sum_i |x_i(t+1) - x_i(t)| \leq L \sum_i |x_i(t)|$$

14

Worcester Polytechnic Institute

Optimization Problem

- Goal: Find efficient portfolios at multiple time periods subject to constraints on the **booksize** and **turnover**

$$\begin{aligned} \min \bar{x}^T \Omega \bar{x} - \frac{1}{\lambda} \bar{\mu}^T \bar{x} & \quad (\text{minimize utility}) \\ \text{s.t. } \sum_i |x_i| \leq M & \quad (\text{booksize constraint}) \\ \sum_i |x_i - x_i^{old}| \leq L \sum_i |x_i^{old}| & \quad (\text{turnover constraint}) \end{aligned}$$

15

Worcester Polytechnic Institute

Linearization

Original Problem:

$$\begin{aligned} \min \bar{x}^T \Omega \bar{x} - \frac{1}{\lambda} \bar{\mu}^T \bar{x} \\ \text{s.t. } \sum_i |x_i| \leq M \\ \sum_i |x_i - x_i^{old}| \leq L \sum_i |x_i^{old}| \end{aligned}$$

16

Worcester Polytechnic Institute

Linearization

Linearized Problem:

$$\min U = x^T \Omega x - \frac{1}{\lambda} \mu^T x$$

s.t. booksize constraint :

turnover constraint :

$$b_i^+ - b_i^- = x_i$$

$$y_i^+ - y_i^- = x_i$$

$$b_i^+ \geq 0$$

$$y_i^+ \geq 0$$

$$b_i^- \geq 0$$

$$y_i^- \geq 0$$

$$\sum (b_i^+ + b_i^-) \leq M$$

$$\sum y_i^+ + y_i^- \leq L \sum |x_i^{old}|$$

17

Worcester Polytechnic Institute

Proof (Booksize Linearization)

Let $A = \{(x, b^+, b^-) : x = b^+ - b^-, b^+ \geq 0, b^- \geq 0, b^+ + b^- \leq M\}$

Let $B = \{x : |x| \leq M\}$

We want to show that minimizing utility over A is equivalent to minimizing utility over B.

$$\Rightarrow \forall x \in B, \exists b^+, b^- \text{ s.t. } (x, b^+, b^-) \in A \quad (1)$$

and

$$\text{if } (x, b^+, b^-) \in A, x \in B \quad (2)$$

18

Worcester Polytechnic Institute

Proof (cont.)

(2): if $(x, b^+, b^-) \in A$, show $x \in B$

$$b^+ - b^- \leq b^+ + b^- \leq M$$

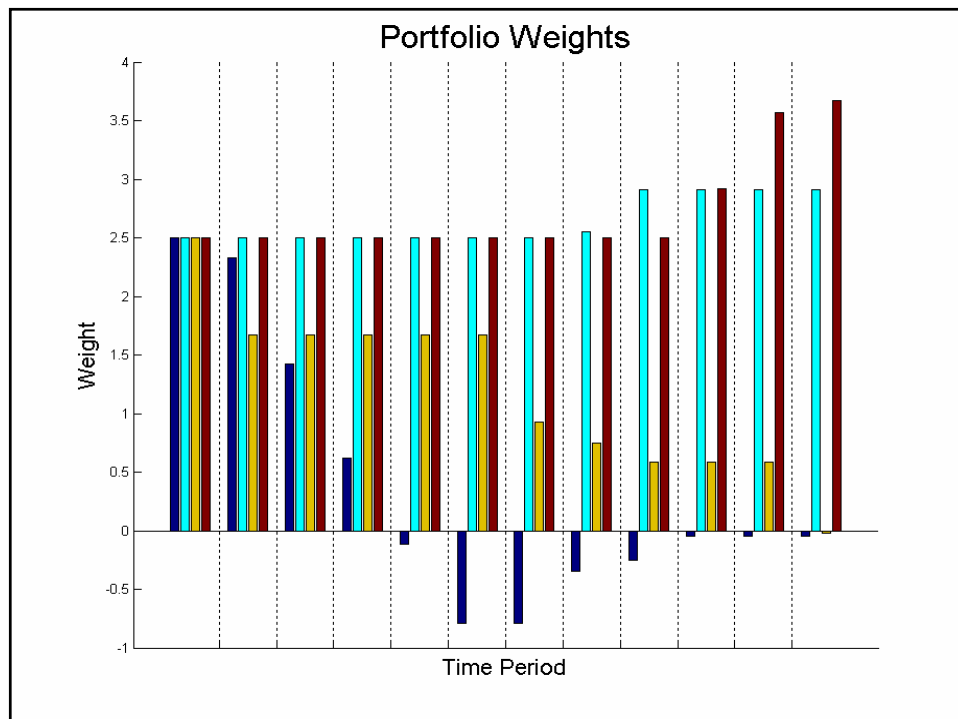
if $x \geq 0$, then $x = |x| \leq b^+ + b^- \leq M$

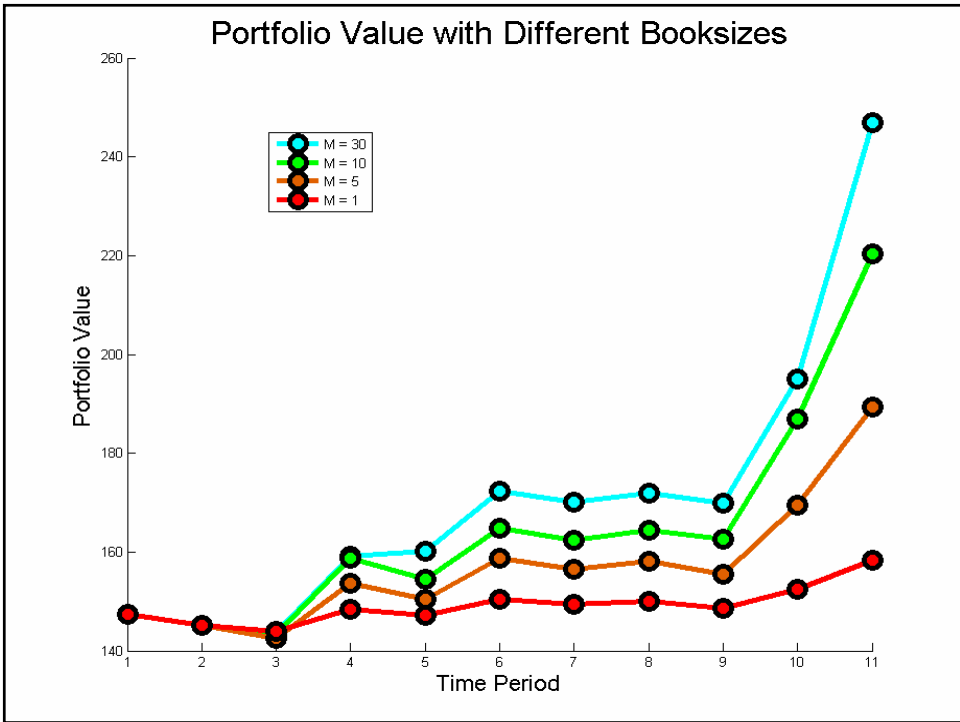
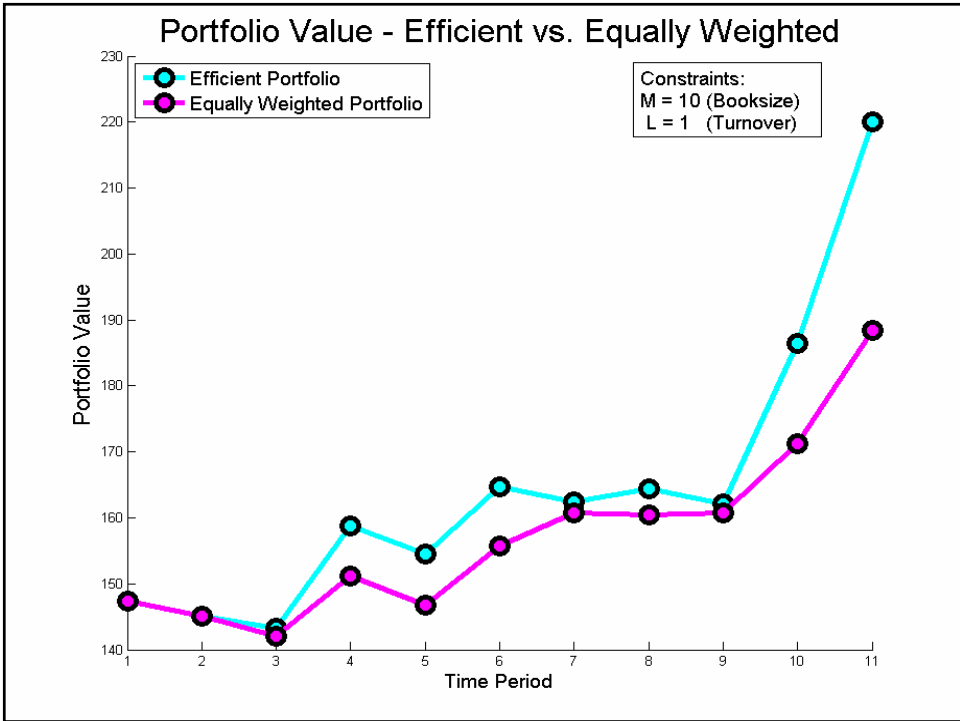
if $x < 0$, then $x = -|x| \leq b^- - b^+ \leq b^+ + b^- \leq M$

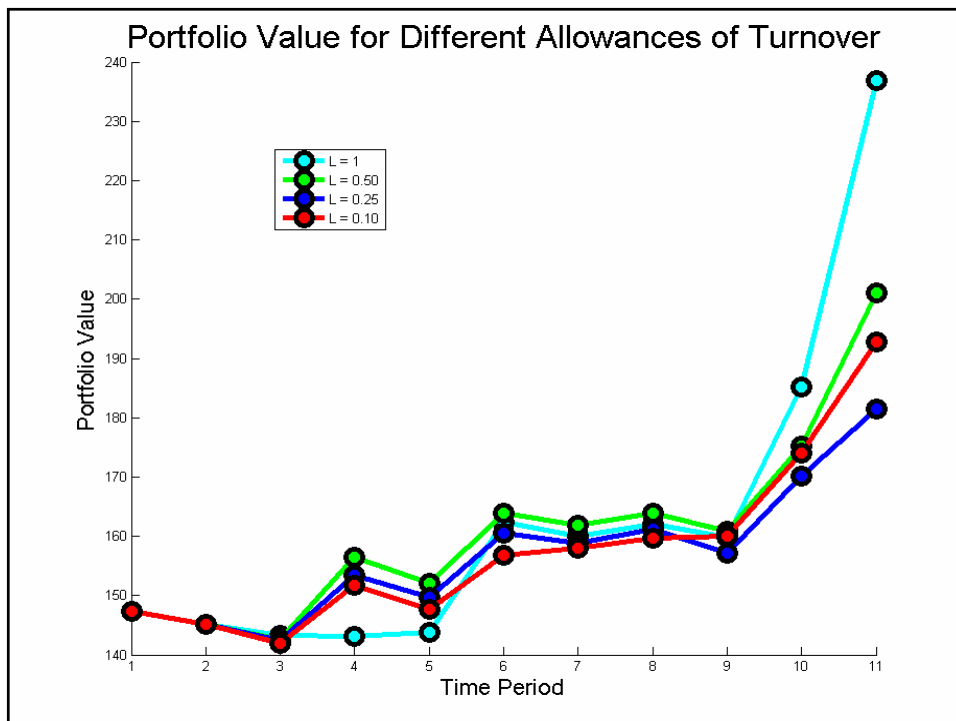
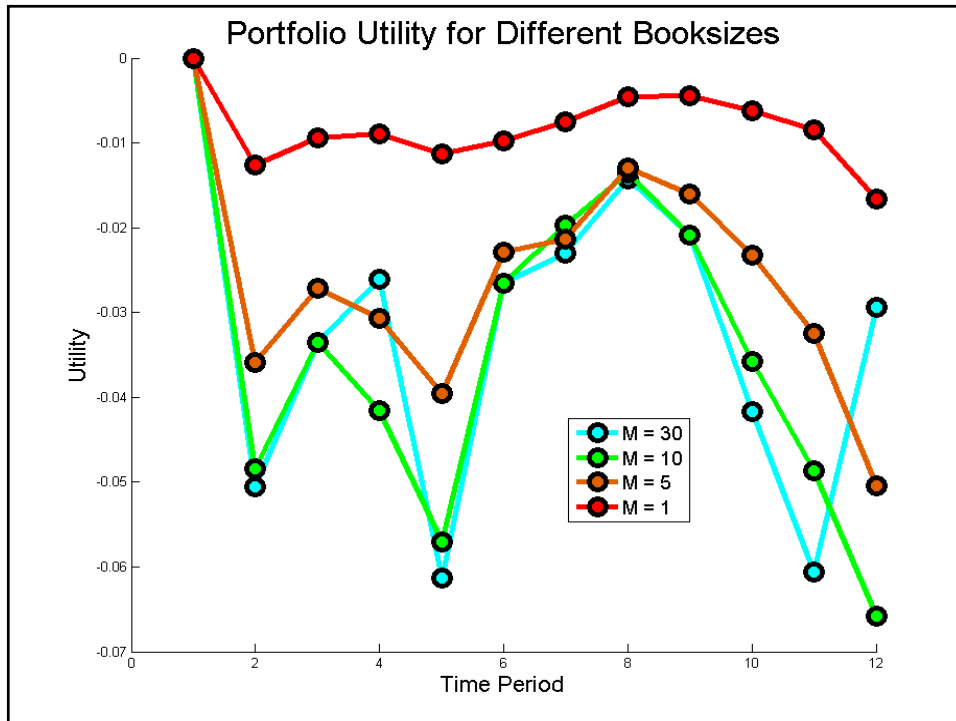
$$\Rightarrow |x| \leq M$$

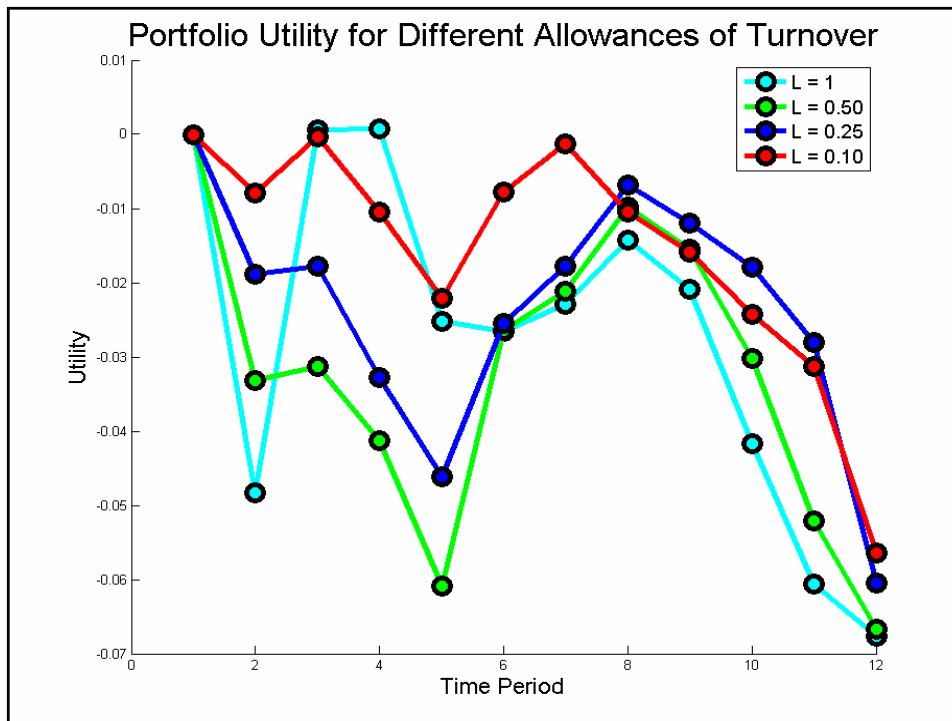
19

Worcester Polytechnic Institute







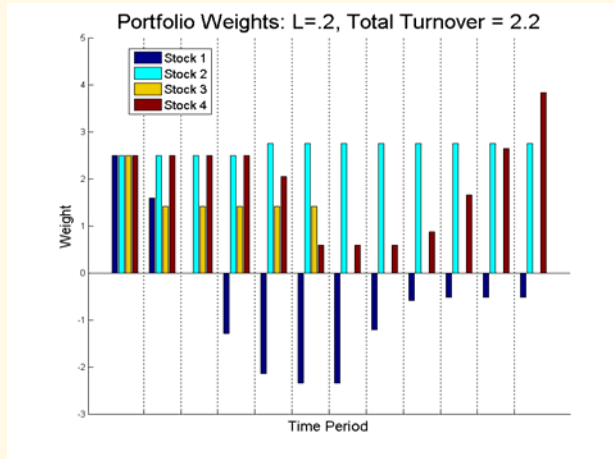


Turnover Budgeting

- Atomistic: Current implementation allows the same amount of turnover at each time period
- Holistic : Can we allow different amounts of turnover to get similar utility at the end?
- Issue: What are criteria for turnover allocation?

Turnover Budgeting

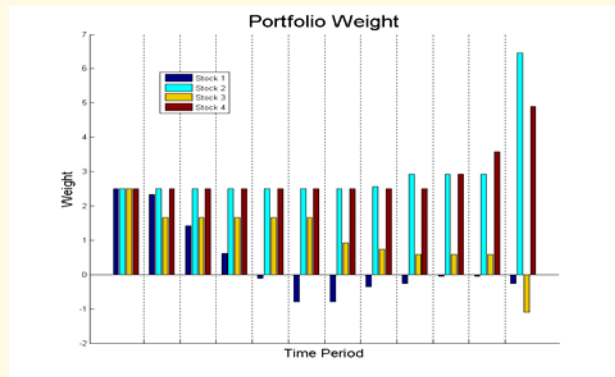
- Example: $L=.2$
- Total turnover = 2.2



27

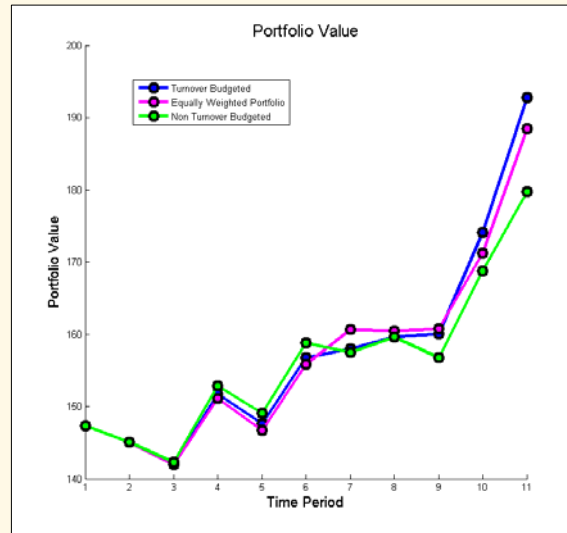
Turnover Budgeting

- $L = .1$
on every step except last
- Final Step: $L = 1.2$
- Total turnover = 2.2



28

Turnover Budgeting



29

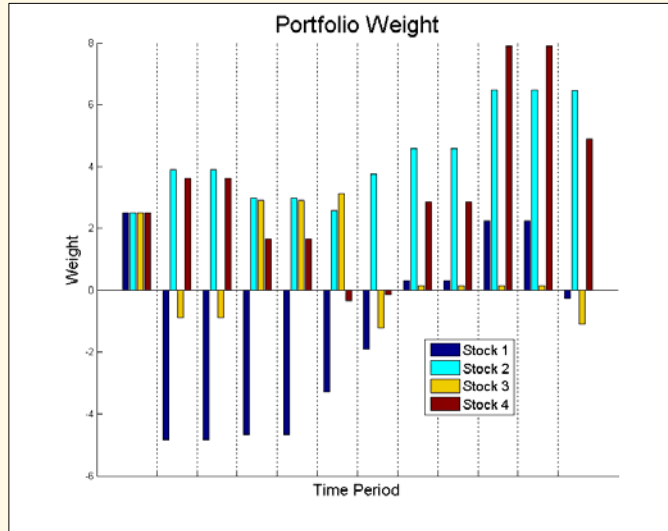
Threshold Trading

- Does it make sense to make a bunch of small trades?
- Idea: Only make the trade if the expected change in utility is large enough

$$\frac{(x^T \Omega x - \mu^T x) - (x_{old}^T \Omega x_{old} - \mu^T x_{old})}{|x_{old}^T \Omega x_{old} - \mu^T x_{old}|} < P < 0$$

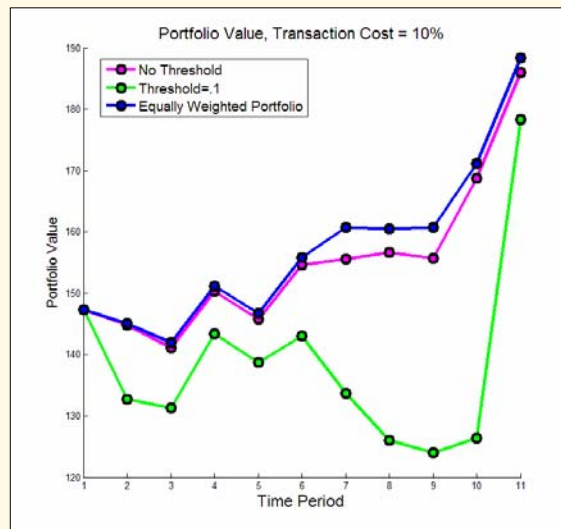
30

Threshold Example



31

Comparison of Values



32

Summary

- Splitting constraint variables allowed use of standard QP package
- Optimization yielded better value and utility than equally weighted portfolio
- Transaction costs can be limited by constraining turnover statically, dynamically, or using a threshold for trading

33

Worcester Polytechnic Institute



Bibliography

- [1] Bade, William, Jessica Clark, and Alex Mills. (2005). Portfolio Optimization with Non-Smooth Constraints. A Report for the Research Experience for Undergraduates in Industrial Math and Statistics at Worcester Polytechnic Institute.
- [2] Grinold, Richard and Kahn, Ronald. (2000). Active Portfolio Management.
- [3] Markowitz, H. M. (1952). Portfolio Selection, Journal of Finance, Vol. 7, Iss. 1, p. 77-91.
- [4] Scherer, Bernd. (2002). Portfolio Construction and Risk Budgeting. Risk Waters Group, London.