Imperial College London

Department of Electrical and Electronic Engineering

Final Year Project 2016

Project Title:	**Optimal Portfolio Updates with Transaction Costs**
Student:	**Efstathios Alex Voinas**
CID:	**00841767**
Course:	**EE3**
Project Supervisor:	**Dr. Imad M. Jaimoukha**
Second Marker:	**Prof. Alessandro Astolfi**

Abstract

A frequent question in finance is: "How to maximise the return of an investment while trying to minimise its risk?". A widely used method to significantly reduce the exposure of an investment to risk is by grouping multiple different assets in a single portfolio. The algorithm created in this project is a tool addressed for everyday investing. The user gives as input parameters found from analysis (i.e. securities' expected returns) and determines the amount of risk to be taken or the lowest wanted return. As output portfolio weights, that match the optimal point on the efficient frontier, are given. The algorithm supports both the creation of a new investment position as well as the update of an old one, taking into account the transaction costs. The main theoretical context is consisted of the Mean-Variance Portfolio Analysis, part of H. Markowitz Modern Portfolio Theory. The concept of closely tracking a benchmark, as well as modelling the transaction costs are considered. The appropriate portfolio weights are calculated in order to match the discretised nature of market securities. The main method used is convex optimisation where feasible calculation time and computational effort are taken into account. All codes are implemented in MATLAB using the convex programming software CVX. The algorithm successfully gives the optimal portfolio weights that the investor should obtain on a specific time period according to its risk, return and transaction costs desires. Additionally, the decline of the risk (variance) as the number of securities increases and the benchmark tracking error effect were observed and discussed.

Acknowledgements

This page is dedicated to all the people that played even a small role in the realisation of the project. I would first like to thank my supervisor Dr. Jaimoukha for introducing me to the world of optimisation and for everything he has done so far to help me whenever I needed something. I would also like to thank my tutor Dr. Papavassiliou for the dozens of conversations we had about anything, I learned a lot - especially of the stuff that cannot be found in any bibliography. I want to thank my flatmates Alex and Diyar for supporting me whenever possible and also about all the good memories we had even in stressed times. Also, I would like to thank my parents Maria and Mihali for their support and being close to me, even from the other side of Europe. Last but not least I would like to thank Anastasia, especially for the times that give me strength and courage without even saying a word. Finally, a big thank you to all my friends, especially to those that when I told them I was doing my final year project, asked me about the subject. This was helpful guys, repetition is mother of learning.

Contents

1 Introduction **1**

1.1 Background 1

1.2 Valuing Methods 1

1.3 Portfolio Theory & Optimisation 2

2 Aims & Deliverables **3**

3 Notation **3**

4 Theory **4**

4.1 Mean-Variance Analysis 5

4.1.1 Efficient Frontier 5

4.1.2 Minimum Variance Portfolio 6

4.2 Benchmark Exposure 6

4.3 Transaction Costs 6

5 The Algorithm **8**

5.1 Main 8

5.2 Optimal Portfolio Update Function 9

5.2.1 Optimisation 9

5.2.2 Discretisation 12

5.2.3 Transaction Costs 13

6 Experiments **14**

6.1 Efficient Frontier 15

6.2 Benchmark Constraint Effect 17

6.3 Portfolio Variance & Number of Assets 18

6.4 Transaction Costs Aversion Coefficient Effect 19

7 Real Life Case Study **20**

7.1 Starting New Investment Position 20

7.2 Portfolio Update 21

8 Future Work **22**

9 References **23**

10 Appendix 1 **25**

10.1 MATLAB MaxReturn Optimisation Function 25

10.2 MATLAB MinVar Optimisation Function 26

10.3 MATLAB Experiment 1 Main 27

10.4 MATLAB Experiment 2 Main 28

10.5 MATLAB Experiment 3 Main 29

10.6 MATLAB Experiment 4 Main 30

10.7 MATLAB Experiment 4 MaxReturn 32

10.8 MATLAB Case Study Main 33

10.9 MATLAB Case Study `OptPUfunc` Function (Max Return full vers.) 36

11 Appendix 2 **40**

1 Introduction

1.1 Background

Mathematical programming or optimisation is used whenever, under some conditions, a decision needs to be made on a set of possible options. Optimisation problems occur in both the fields of engineering and finance and great effort is put in order to make the algorithms not only functional but efficient too. Although they have different notation, both fields use the same mathematical approach in order to deal with these situations.

This project gives an answer to the question: "How to maximise the return on an investment while trying to minimise its risk?". Expected return and risk (or volatility for securities) are the main assets' characteristics. A way for significantly reducing the exposure of an investment to risk is by investing in many different assets and grouping them in a portfolio. This method is called *portfolio diversification*. The bigger the number of assets, the less the volatile is the portfolio. The amount of risk that can be eliminated by an appropriate asset combination is the non-systematic (or *diversifiable*) risk while the systematic (or *non-diversifiable*) risk involves various unexpected events, such as a flood or a war break out, and it cannot be eliminated.[1][2]

1.2 Valuing Methods

For securities both risk and return can be analysed from the *fundamental*, *technical* and *sentimental* point of view. In the fundamental analysis, analysts try to estimate the overall value of the company that is issuing the security in order to make assumptions. Aside from the markets' quantitative factors, the general economic context is taken into account, including macroeconomic and company-specific quantitative/qualitative factors.[3] Technical analysis is based only on market statistics. In terms of the volatility, analysts use historical charts to calculate the variance per item and the covariance in between items. An asset's risk is directly measured from its variance, while how it performs regarding to others is calculated by their covariance matrix. The expected return is estimated in a same manner using only quantitative models on historical values in order to make predictions for the future.[4] Thirdly, the sentimental analysis covers factors such as global events or opinions that have an indirect, yet very significant, effect on the securities' price and return.[5]

1.3 Portfolio Theory & Optimisation

The main parameter that affects the portfolio synthesis is the correlation of the included assets' behaviours which are indicated by their covariance matrix. A simple example to illustrate the effect of correlation is to consider a simple two-asset type portfolio. We then distinguish the two extremes, the assets being: (a) perfectly positive correlated (Figure 1a) and (b) perfectly negative correlated (uncorrelated) (Figure 1b). In the first case the two assets have the same volatility and their return graphs present identical fluctuations. In that case an optimal risk-averse portfolio cannot be created as any combination is equally exposed to the initial assets volatility. In the second case, each of the two return graphs has exactly the opposite variations. Therefore, the optimal portfolio, in that case, has equal weights of a half per asset which cancels the portfolios overall volatility (i.e. when the one presents loss, the other cancels it with the same amount of return and vice versa). Theoretically, this is the best possible case if we want to eliminate the diversifiable risk but it is impossible to actually achieve it.[1] For this reason, in reality many assets of different characteristics are put together in different proportions in order to get as close as possible to the best result. Portfolio managers constitute their portfolios from a wide range of assets, such as from different industries or even from different countries, according to their forecasts.[6]

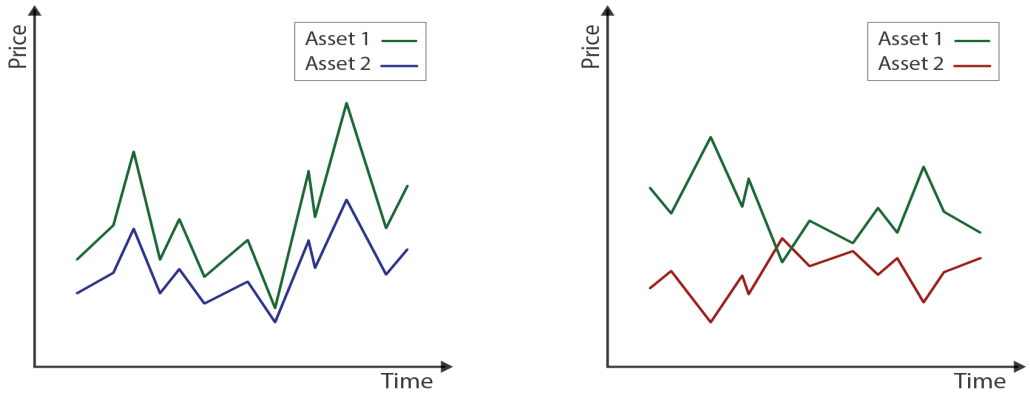


Figure 1: Assets 1&2: (a) perfectly correlated, (b) perfectly uncorrelated.

Our goal is achieved through solving an optimisation problem that finds the best portfolio weights for a set of predetermined assets. The investor might be either starting a new investment position or already having an initial portfolio in which case the program optimally updates it. Risk, return and transaction costs are all taken into account.

2 Aims & Deliverables

The purpose of this project is to apply some optimisation tools developed in engineering into common problems that arise in finance. The main problem is to get the optimal portfolio weights so that either the return is maximised, the risk is minimised or a trade-off in between according to the user's desires. Secondly, the user can control the transaction load by choosing how much he wants to spend on transaction costs. For example, this helps in a situation which the investor wants to liquify all his investments even though it is not the optimal decision in terms of return.

The deliverable is a feasible algorithm that addresses both professionals, handling dozens of portfolios, or individual investors. It takes as input data regarding the behaviour of particular assets, predicted or calculated from market analysis. As output it gives the optimal portfolio updates (buy/sell/stay, per asset) for the user to make, measuring all figures and parameters.

All coding and simulations are made via MATLAB using the convex programming software CVX. In particular, the semidefinite programming extension is used for the posture of the problem.

3 Notation

The notation used to illustrate the theoretical context is summarised below. For each of the symbols that appear in the code, its representation is shown in parenthesis.

$\mathbb{R}^m$	= set of all real valued $m \times 1$ vectors
$\mathbb{R}^{m \times n}$	= set of all real valued $m \times n$ matrices
A^T (`A'`)	= transpose of matrix A
$A \geq 0$ ($A \leq 0$)	= positive (negative) semidefinite matrix A
$O_{m \times n}$	= $m \times n$ null matrix
I_m (`eye(m)`)	= identity matrix of dimension m
$\mathbf{1}$ (`e`)	= ones vector

m (`m`)	= number of assets
b_t (`bt`)	= total budget (invested and free combined)
b_{t_0} (`bt0`)	= initial invested budget
c (`c`)	= free budget (amount to be invested)
$\mathbf{y}$ (`y`)	= expected returns per asset vector
y_0 (`y0`)	= minimum portfolio expected return
y_1 (`y1`)	= overall portfolio expected return
g (`g`)	= maximum accepted portfolio variance (risk)
Σ (`S`)	= symmetric covariance matrix
$\mathbf{x}$ (`x`)	= optimal portfolio weights (decision) vector
$\mathbf{x_0}$ (`x0`)	= initial portfolio weights vector
$\mathbf{x_d}$ (`xd`)	= optimal portfolio update weights vector
χ (`chi`)	= optimal portfolio amounts vector
$\mathbf{p}$ (`p`)	= asset prices vector
$\mathbf{N_t}$ (`numt`)	= optimal portfolio's asset quantities vector
$\mathbf{N_0}$ (`num0`)	= portfolio's initial asset quantities vector
$\mathbf{N}$ (`num`)	= traded asset quantities vector
λ (`lambda`)	= risk aversion coefficient
λ_{TC} (`ltc`)	= transaction costs aversion coefficient
TC (`tc`)	= total transaction costs for the update
$\mathbf{z}$ (`z`)	= indicator function vector
α (`alpha`)	= constant amount in transaction costs
β (`beta`)	= percentage on the traded amount in transaction costs
$\mathbf{x_b}$ (`xb`)	= benchmark index weights vector (normalised)
d_{max} (`d_max`)	= maximum overall difference from benchmark index

4 Theory

The theoretical context incorporated into the algorithm mainly involves the *Mean-Variance Analysis* (part of the Nobel prize winning paper on Modern Portfolio Theory from Harry Markowitz, 1952), which fundamental concept is that of *Efficient Frontier*. Additionally, the *Benchmark Exposure* constraint and the *Transaction Costs* modelling, are also elaborated.

4.1 Mean-Variance Analysis

The Mean-Variance Analysis is based on the idea that there is a trade-off between the risk and return in an investment. The more volatile (higher variance) an asset's price is, the higher its expected return, but in the same manner it could end up having huge losses. Therefore, high expected return is associated with high risk and vice versa. Through the Mean-Variance Analysis, portfolio managers group assets in a way to maximise the overall expected return for a given level of risk, denoted by the portfolio's variance.[8]

4.1.1 Efficient Frontier

The Efficient Frontier (EF) is the curve showing the best possible portfolio mixtures that can be achieved with specific assets regarding the risk-return trade-off. It can be interpreted in two ways: it either shows the maximum return possible that can be achieved for a specific level of risk or the minimum risk that can be achieved for a specific return. An illustration can be seen in Figure 1. All points on the right of the curve are the possible good portfolios. The optimal of them all are those as close as possible to the EF curve. There is an exception for all the points below the horizontal line passing through the minimum variance point. For each one of these points, the same amount of risk can be achieved with higher return (by moving upwards on the graph) or the same amount or return can be achieved with lesser risk (by moving leftwards on the graph). The goal of the portfolio managers is to set the portfolio weights in such a way to be on the EF curve (if possible) according to the investors' risk-return desires.[1][8]

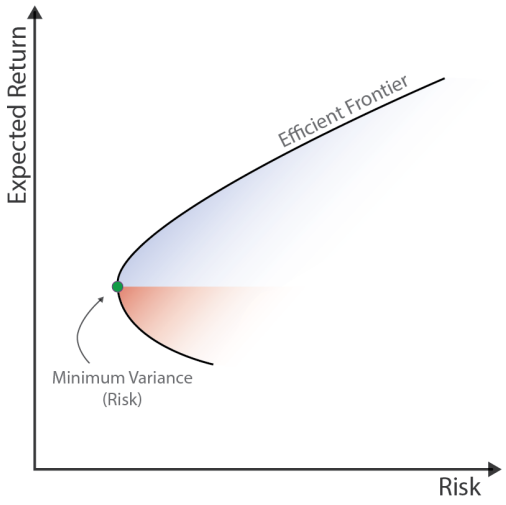


Figure 2: The Efficient Frontier curve: (a) optimal portfolios (curve), (b) good portfolios (blue), (c) inefficient portfolios (red), (d) the Minimum Variance Portfolio point (green).

4.1.2 Minimum Variance Portfolio

The Minimum Variance Portfolio refers to a portfolio which assets are weighted in a way to give the lowest possible overall portfolio variance. (Figure 1) This approach is used whenever the investors want to "play it safe" and eliminate the exposure to volatility (variance) without caring about how high the return would be. Mathematically this can be expressed with the minimisation problem:

$$min\{\mathbf{x}^T \Sigma \mathbf{x} | \mathbf{1}^T \mathbf{x} = 1\}$$

Where for m assets in the portfolio: $\mathbf{x}, \mathbf{1} \in \mathbb{R}^m$ and $\Sigma \in \mathbb{R}^{m \times m}$.[8]

4.2 Benchmark Exposure

Sometimes the portfolio managers want to assign the assets' weights so as to stay relatively close to a benchmark. For example, stock investors that pool stocks from a specific index or index fund managers would like to closely follow the index market capitalisation weights (or benchmark weights), such as of S&P 500.[7] Mathematically this can be expressed as:

$$|\mathbf{x} - \mathbf{x_b}| \leq d_{max}$$

Where $\mathbf{x}, \mathbf{x_b} \in \mathbb{R}^m$ and d_{max} is the overall maximum wanted difference.

4.3 Transaction Costs

The transaction costs is the amount paid in order to perform a financial trade. The most common transaction costs are fees and commissions that the institutions take in order to perform the transaction on behalf of the investors. The transaction costs, even though indirect, they should be taken into account and modelled appropriately as they can heavily reduce the returns.[7][9] For example, when a portfolio update is to take place, some of the optimal proposed weights might be very close to the initial ones making the transaction costs higher than the actual transaction. Situations like that must be avoided.

Sometimes modelling the transaction costs might require complex nonlinear functions. In this project the transaction costs are modelled with a constant term plus a percentage on the traded amount. These are similar to the conditions that individuals face on their everyday investing through mediators (i.e. banks, brokers). Normally, a transaction

is charged with: an amount coming from a percentage, that does not fall below a certain minimum, regardless of the overall traded amount. Usually the percentage is close to $0.5\% - 1\%$, while the lower bound amount is around £5−£12[10] (similar for other currencies).[11]

In order to model the previously mentioned condition a piecewise function had to be incorporated in the program. This would have harshly increased the algorithm's computational effort and therefore it was avoided. By always including both the constant amount and the percentage, the transaction costs calculated by the program are increased. This is not a big issue, as:

(a) For very large amounts the constant amount is a small percentage of the overall expense so its presence is not problematic. In the same manner for small amounts the percentage has a tiny effect.
(b) The increased calculated transaction costs mean only that the investor will actually pay less in reality keeping a small amount of money uninvested and free to use.

The transaction cost associated with the trading of $\chi(i)$ amount of asset i, modelled with a constant plus a percentage, mathematically can be shown as:

$$TC_i(\chi_i) = \alpha_i|z_i| + \beta_i|\chi(i)|$$

Where α_i and β_i are the constant amount and the percentage respectively, and z_i is the indicator function for which we have:

$$z_i = \begin{cases} 1, & \chi(i) > 0 \\ 0, & \chi(i) = 0 \\ -1, & \chi(i) < 0 \end{cases}$$

The absolute value of the indicator function is taken as the transaction costs can only be positive (paid by the investor) no matter what his move is: i.e. buying (positive $\chi(i)$) or selling (negative $\chi(i)$). For the total transaction costs for a portfolio update we have:[7]

$$TC = \sum_{i=1}^{m} TC_i = \alpha^T|\mathbf{z}| + \beta^T\chi$$

5 The Algorithm

The basic tool used for the code implementation is MATLAB. The set up is done through a `Main` function which calls the optimal portfolio update function `OptPUfunc`. The optimisation part is implemented using the convex programming software CVX and particularly its semidefinite extension.

5.1 Main

All inputs are declared in the `Main`.[1] Firstly, all the parameters related to the considered assets are declared, these are: their number (`m`), expected returns vector (`y`), covariance matrix (`S`) and prices vector (`p`). Secondly, the user's desires on the risk-return trade-off are set. These include the maximum accepted risk level (`g`), the minimum wanted overall expected return (`y0`) and the risk-aversion coefficient (`lambda`). The risk-aversion coefficient (denoted λ) works as a regulator between the portfolio's overall return and risk. By setting its value the user chooses how much risk he wants to accept.[2]Thirdly, the initial portfolio weights vector (`x0`) is inputted. In the case of zero initial investments all its entries are set to zero. Additionally, the budget is set, constituted of two parts: the money out of the market (to be invested) (`c`) and the initial invested budget (`bt0`). Next, the parameters regarding the transaction fees and charges are declared, these are: the constant term (`alpha`), the percentage (`beta`) and the transaction costs aversion coefficient (`ltc`). Similar to the risk-aversion coefficient, the transaction costs aversion coefficient (denoted λ_{TC}), is a regulator between the overall return and the transaction costs. Through it the user can choose how much "damage" the transaction costs will inflict to the overall return amount.[3] Finally, the benchmark characteristics are inputted, these are: the benchmark index weights vector (`xb`) (normalised to the assets of our case) and the total accepted differentiation amount (`d_max`).

After all inputs are declared, in order to find the optimal policy, `OptPUfunc` function is ran for all possible combinations of buy/sell/stay per asset. This is done with the use of vector `z` (which then works as the indicator function-z inside `OptPUfunc`). After all policies are found the one that gives the biggest overall return `y0` is finally outputted.

[1]Shown fully in the Appendix.

[2]See **Risk Aversion** paragraph in section 5.2.1.

[3]See section 5.2.3.

The pseudocode of these is as follows:

Data: zz=[-1,0,1]'
store all possible combinations of zz entries (for each asset) in matrix zzz;
for *each column of* zzz **do**
| through OptPUfunc find y0;
end
store y0 for the first case to vector qq;
for *a column less than those of* zzz **do**
| **if** y0 *is larger than the previous* **then**
| | store it in qq overwriting the previously stored;
| **end**
end

5.2 Optimal Portfolio Update Function

The optimal portfolio update function OptPUfunc,[4] has three main parts: (a) the optimisation, (b) "discretisation" and (c) the transaction costs calculation part.

5.2.1 Optimisation

The optimisation part has three different versions according to the problem that is being solved each time. As previously stated the user is given the freedom to choose between: the maximisation of returns, the minimisation of the variance (risk) or through the use of a risk-aversion coefficient λ choose a point in between. For each case a different function is used, but as will be shown below, all three of them are just variations of the same concept.

Semidefinite programming is used. It requires both the expression to be optimised and the constraints to be linear. Its difference from linear programming is that it allows the constraint inequalities to have as elements positive semidefinite matrices.[12] In order to meet the above requirements all nonlinear components (i.e. quadratic) were changed to positive semidefinite matrices using the *Schur complement*.[5]

[4] Shown in the Appendix.

[5] The Schur complement states that: $M = \begin{bmatrix} A & B \\ C & D \end{bmatrix}$, where: $A \in \mathbb{R}^{m_1 \times m_1}$, $B \in \mathbb{R}^{m_1 \times n_1}$, $C \in \mathbb{R}^{n_1 \times m_1}$ and $D \in \mathbb{R}^{n_1 \times n_1}$ invertible, is equivalent to: $\frac{M}{D} = A - BD^{-1}C$, where: $M \in \mathbb{R}^{(m_1+n_1)\times(m_1+n_1)}$.[13]

Max Return The optimisation that is being solved in order to maximise the overall return (regardless of how much risk is taken[6]) is as follows:

1. $\max_{\mathbf{x_d}}\{y_0\}$ **subject to:**
2. $\mathbf{y}^T(\mathbf{x_0}+\mathbf{x_d}) - \lambda_{TC}\big(\alpha^T|\mathbf{z}|\cdot {b_t}^{-1} + \beta^T|\mathbf{z}\circ\mathbf{x_d}|\big) \geq y_0$
3. $\begin{bmatrix} g & (\mathbf{x}+\mathbf{x_d})^T \\ (\mathbf{x}+\mathbf{x_d}) & \Sigma^{-1} \end{bmatrix} \geq 0$
4. $\mathbf{1}^T(\mathbf{x_0}+\mathbf{x_d})\cdot b_t + \alpha^T|\mathbf{z}| + \beta^T|\mathbf{z}\circ\mathbf{x_d}|\cdot b_t \leq b_t$
5. $0 \leq x_0(i) + x_d(i) \leq 1, \quad \forall i = 1,\ldots,m$
6. $0 \leq \mathbf{1}^T(\mathbf{x_0}+\mathbf{x_d}) \leq 1$
7. $\big(z(i)=0 \rightarrow x_d(i)=0\big), \quad i=1,\ldots,m$
   $\big(z(i)=1 \rightarrow x_d(i)\in[0,1]\big), \quad i=1,\ldots,m$
   $\big(z(i)=-1 \rightarrow x_d(i)\in[-1,0]\big), \quad i=1,\ldots,m$
8. $\begin{bmatrix} d_{max} & (\mathbf{x_0}+\mathbf{x_d}-\mathbf{x_b})^T \\ (\mathbf{x_0}+\mathbf{x_d}-\mathbf{x_b}) & I_m \end{bmatrix} \geq 0$

Each of the above lines is explained below.

1. This is a maximisation of the overall return (denoted as y_0) and the variable that changes is the proportions of the portfolio update vector (denoted as $\mathbf{x_d}$).

2. The left hand side on this inequality is the expression to be maximised. Instead of putting it on top, the overall return `y0` is maximised and everything must be greater or equal to it, which practically is the same thing. In this expression the updated portfolio's return is "damaged" with the transaction costs which are controlled by the transaction costs aversion coefficient (λ_{TC}).[7]

[6] There is an upper bound accepted risk amount (`g`) that is given by the user in `Main`.

[7] For detailed explanation of - λ_{TC} - the transaction costs aversion coefficient see section 5.2.3.

3. This is the inequality that ensures that the updated portfolio's variance will stay below the lower level `g`:

$$(\mathbf{x_0} + \mathbf{x_d})^T \Sigma (\mathbf{x_0} + \mathbf{x_d}) \geq g$$

Then by using the Schur complement we end up to the constraint (3) in the optimisation.

4. This constraint ensures that the amount that invested with the updated portfolio $(\mathbf{1}^T(\mathbf{x_0} + \mathbf{x_d}) \cdot b_t)$, plus the transaction costs spent are less than the initial total budget.

5. This simply indicates that each entry in the updated portfolio is assigned a reasonable value. As the entries are percentages they should be in $[0, 1]$.

6. Similar to the above, but this time the sum of all entries should be in $[0, 1]$ as it refers to the percentage of the portfolio as a whole. (i.e. percentage of the overall budget)

7. Through this condition the indicator function (created through the `Main`) is inputted in the optimisation. The entries of the vector `z` control what the entries of the update vector `xd` are. For a `z`'s zero entry the updated proportion for that specific asset is set to zero. Whenever `z(i)` is in $\{-1, 1\}$, the update vector's entry is assigned with the same sign.

8. This is the benchmark tracking constraint. Instead of the actual magnitude of the deviations vector $\big((\mathbf{x_0} + \mathbf{x_d}) - \mathbf{x_b}\big)$ its square is minimised. This is expressed as the inner product of the vector with itself:

$$\big((\mathbf{x_0} + \mathbf{x_d}) - \mathbf{x_b}\big)^T \big((\mathbf{x_0} + \mathbf{x_d}) - \mathbf{x_b}\big) \leq d_{max}$$

Which then same as to (3) by using the Schur complement it is turned to inequality (8).

Min Variance The optimisation aiming to minimise the portfolio variance (or risk) regardless of the overall expected return[8] is only slightly different from the previous case. The only difference concerns the type of optimisation that it is being solved. In this case we have the minimisation rather than a maximisation. The variable being minimised is `g`. Therefore, the optimisation problem is posed as: $\min_{\mathbf{x_d}} \{g\}$ and everything else stays the same.

Risk Aversion In this case a maximisation problem is being solved. Now instead of maximising only the overall return (`y0`), the variance term is also added in the expression multiplied by λ - the risk aversion coefficient. This is written as:

$$\max_{\mathbf{x_d}} \left\{ (\mathbf{x_0} + \mathbf{x_d})^T \mathbf{y} - \lambda (\mathbf{x_0} + \mathbf{x_d})^T \Sigma (\mathbf{x_0} + \mathbf{x_d}) - \lambda_{TC} \big(\alpha^T |\mathbf{z}| \cdot {b_t}^{-1} + \beta^T |\mathbf{z} \circ \mathbf{x_d}| \big) \right\}$$

Comparing with the **Max Return** case; their only difference is that the above takes the place of line (1), while the lines (2) and (3) are removed. (they are actually incorporated in the expression)

The risk aversion coefficient (λ) is a real positive number which value is set by the user. Its purpose is to control the weighing of the optimisation on risk (portfolio's variance). The higher its value is, the more risk averse portfolios the optimisation will output. For example, for a high λ the optimisation will settle to a relatively non-risky portfolio making the overall expected return somewhat smaller. The trade-off here is that an amount of potential expected return is sacrificed in order to make the rest of returns more secure. The opposite happens for low values of λ. The risk aversion coefficient is normally within the range of $[2 - 3]$. All values approximately below 1.5 are considered too risky and therefore are avoided.[14] In this project the risk aversion coefficient was not used, instead it was broken into the two parts of **Max Return** and **Min Variance**. In this way not only its effect was explored (in an indirect way) but also the results in each case were seen into more detail.

5.2.2 Discretisation

The optimisation outputs portfolio update vectors (`xd`) with entries that do not match discrete asset amounts. In order to fix this, the final portfolio (`x0+xd`) is multiplied with

[8] There is a lower bound on the overall wanted return (`y0`) that is inputted by the user in `Main`.

the total available budget (`bt-tc0`)[9]to get the non-discretised amounts to be invested. Then for each asset, its non-discrete amount is compared with the geometric sequence: [quantity]×[price]. Finally, the greater term of the sequence that is less than the non-discrete amount is stored in the final portfolio amounts vector `chi`.

In order to perform the above, the difference of the discretised from the non-discretised amount is left uninvested. This amount is within the sum of unit prices per asset. This is not very troubling as the amount per asset that is left out of market is within its unit price. A similar situation where there was an uninvested amount left was seen in section 4.3 regarding the modelling of the transaction costs.

The "discretisation" code part [10] can be shown with the pseudocode below:

for *every asset* **do**
  initialise quantity;
  while *quantity×price is less the suggested (non-discretised) amount* **do**
    increase quantity;
  end
  if *quantity less or equal to zero* **then**
    set the final portfolio amounts vector entry to zero;
  else
    if *indicator function entry is zero* **then**
      set the final portfolio amounts vector entry to the initial amount (no change);
    else if *indicator function entry is in* $\{-1, 1\}$ **then**
      set the final portfolio amounts vector entry to (quantity-1)×price;
  end
end

5.2.3 Transaction Costs

The transaction costs appear in three instances in the optimal portfolio update function `OptPUfunc`. Firstly, they are incorporated in the optimisation part of the code. In this way, the optimisation outputs the optimal update vector with them taken into account.

[9]For `tc0` see section 5.2.3.
[10]See the Appendix for the full "discretisation" code.

In the code transaction costs are shown as:

```
ltc*((alpha/bt)'*abs(z)+beta'*(z.*xd))
```

The above expression is the same with the one for the total transaction costs in section 4.3 (with `(z.*xd)` playing the role of the traded amount vector) except everything is divided with the total budget in order to meet the optimisation's needs. Whenever the transaction costs are incorporated in the code, their amount (`tc`) and the portfolio amounts vectors (`chi`) are expressed as percentages of the total budget (`bt`). The transaction costs term is controlled by the transaction costs aversion coefficient (λ_{TC}).

The transaction costs aversion coefficient - λ_{TC}, similar to the risk aversion coefficient is a real positive set by the user. Its function is to control the weighing of the optimisation on transaction costs. The higher it is, the smaller the transaction volumes are ensuring more conservative updates with smaller transaction costs. For λ_{TC} small the optimisation does not take into account the amount spent for the transaction rather than only maximising the return on the investment. Small λ_{TC} are also used whenever the investor wants to liquify its investments (i.e. in times of financial turbulence when all expected returns become negative). In this case, the optimisation will basically suggest to "sell everything" merely taking into account how much will be spent in order to perform the transaction. The transaction costs aversion coefficient dissimilar to the risk aversion coefficient ranges from zero to infinity according to the desires of the user.[7]

After the optimisation is finished the above amount is stored in the variable `tc0`. The amount spent in transaction costs are not available in the total budget after the update has taken place. For this reason, in the "discretisation" part of the code the available budget is stated as (`bt-tc0`). Finally, the transaction costs are calculated after the "discretisation" for the final actual amounts that will be traded (`tc`).

6 Experiments

Some experiments were carried out in order to demonstrate the full range of capabilities of the algorithm. Pieces of the program were isolated and tested independently in order to measure how some specific input parameters affected the overall result. The experiments include: the extraction of EF, the illustration of the benchmark constraint effect, how

the number of assets affect the portfolio's variance and the effect of the transaction costs aversion coefficient λ_{TC}.

For the showcase purposes the stocks of Procter and Gamble Company (PG/$83.35), KeyCorp. (KEY/$11.65) and Apple Inc. (AAPL/$97.46),[11] all parts of the S&P 500 index, were used. They are characterised as low, medium and medium-high risky, respectively. Past figures, up to the 15th of November 1987,[12] were downloaded into Excel spreadsheets. The returns were calculated from the daily open/close prices from which the covariance matrix was calculated.[15][16][17] (shown on Table 1)

	PG	KEY	AAPL
PG	0.00015241	0.00006108	0.00005548
KEY	0.00006108	0.00050032	0.00010811
AAPL	0.00005548	0.00010811	0.00057614

Table 1: Covariance Matrix of PG, KEY, AAPL.

Estimations for their 12-month forward expected returns were also found on the web.[18][19][20] Finally, their weights on the S&P 500 index, needed for the creation of the benchmark vector, were also obtained.[21] (Table 2)

	PG	KEY	AAPL
12-month Forward Expected Returns	1.98%	20.17%	23.13%
Benchmark Vector Weights	0.284902524	0.013046969	0.702050507

Table 2: Expected Returns/Benchmark Weights for PG, KEY, AAPL.

This small three-asset portfolio, even though it can not resemble a real life thirty-asset one, is a good example to experiment and draw conclusions on.[2]

6.1 Efficient Frontier

For the calculation of the EF only the **Max Return** algorithm was used.[13] It was isolated from the benchmark, transaction costs and discretisation effects. Increasing values of the upper bound portfolio variance g were inputted, where the maximum return for each case

[11] Prices at the opening of the 15th of June 2016.

[12] This was the day that KEY was traded on the market for the first time. The other two companies' shares were being traded from before. This date was chosen in order to fully match day-by-day the past trading period and perform the statistics accurately.

[13] See Appendix 1.

was extracted. Total of 160 points were used. The range for g was between the lowest possible (within the range of $\pm 10^{-7}$) that these assets could achieve and a value that the portfolio was being fully risky (i.e. 100% invested in AAPL).

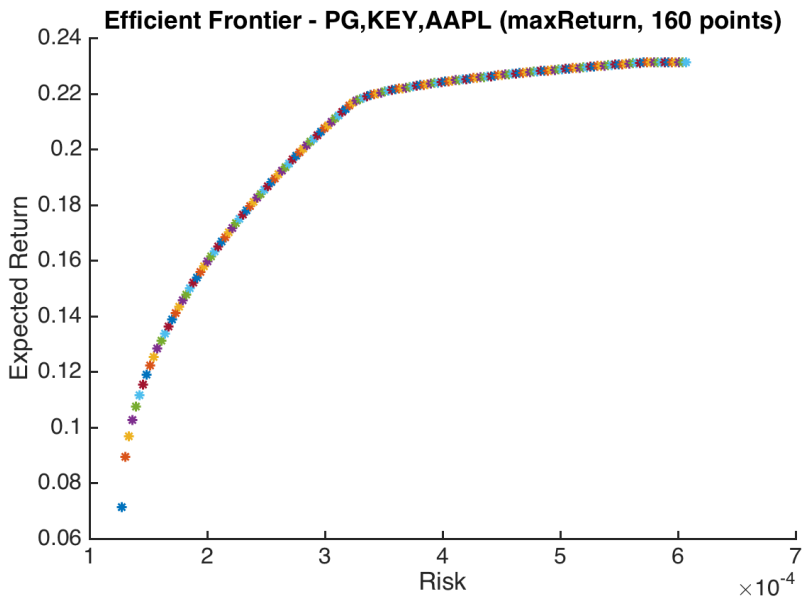


Figure 3: Efficient Frontier for PG, KEY, AAPL.

In Figure 3 the increase of the maximum expected return as the risk (variance) is increased can be seen. For a larger number of assets the curve becomes more smooth. The curve has as limit the expected return of the riskiest one (AAPL in this case: $\approx 23\%$) which is also the highest.[20] Ten points on the curve were chosen in order to exhibit how the portfolio weights change as the accepted portfolio risk increases. (Figure 4)

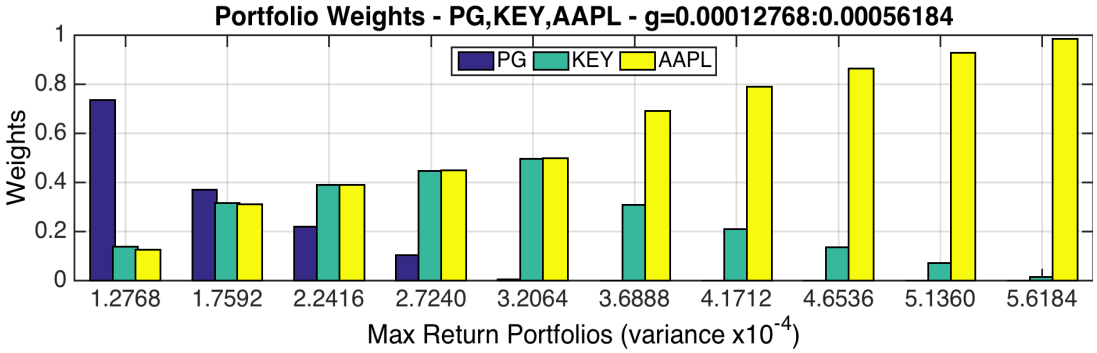
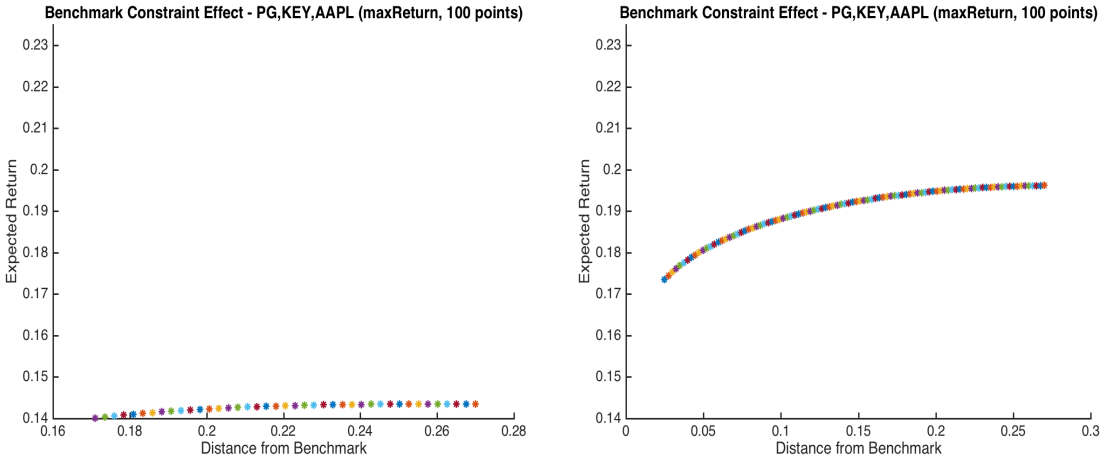


Figure 4: Portfolio weights for g in the range $[0.00012768, 0.00056184]$.

From the above figure it can be observed that as the user accepts to tolerate more risk on the investment the weights in the portfolio accumulate more on the risky assets. For the less risky case it can be seen that PG (less risky) is chosen for $\approx 75\%$, while for the most risky case the entire portfolio is constituted only out of AAPL (most risky).

6.2 Benchmark Constraint Effect

The second experiment involved the effect of the benchmark constraint on the portfolio weights and overall return. Again only **Max Return** algorithm was used.[14] Five of the points used in Figure 4 were used as examples in order to demonstrate the behaviour of the expected return as the distance from the benchmark vector was increasing. The range of `d_max` was from 0.024961 (where the change on results started being observable), until a maximum value (= 0.27) that the constraint's role was minimal.

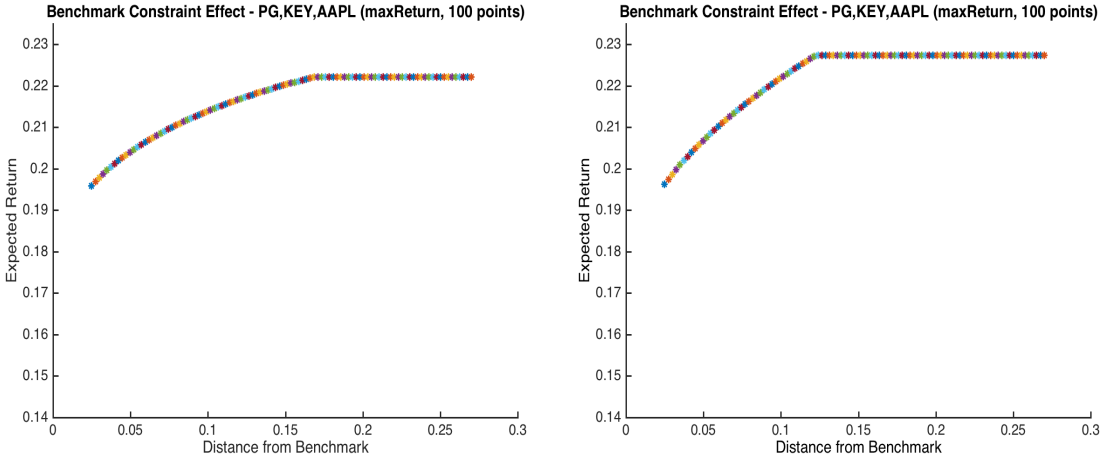


(a) (b)

Figure 5: The benchmark constraint effect for low accepted upper bound variance.

Specifically, the points used had variances ($\times 10^{-4}$) of: 1.7592 (Figure 5.a), 2.7240 (Figure 5.b), 3.6888 (Figure 6.a), 4.6536 (Figure 6.b). In all figures it can be seen that as `d_max` increases the expected return also increases. As the effect of the constraint diminishes the expected returns (and therefore portfolio weights) approach the EF's optimal values. In Figures 6 (a) and (b) it can be seen that the expected returns hit the maximum possible value (portfolio constituted 100% by AAPL). In Figures 5 (a) and (b) the curves are smoother. This happens as the weights in the portfolio are more equally distributed

[14] See Appendix 1.



(a) (b)

Figure 6: The benchmark constraint effect for high accepted upper bound variance.

(lower variance portfolios). In those cases, larger `d_max` is needed in order to make them approach their maximum. The more risk is involved the more abruptly the curves reach their upper bound.

6.3 Portfolio Variance & Number of Assets

For the experiment of how the number of assets affect the portfolio variance the **Min Variance** algorithm was used.[15] In MATLAB 36 assets with randomised statistics were created.[16] In each iteration the program was taking one more asset into account, performing the optimisation for different expected returns (`y0`) in each case (Figure 7, vertically aligned points). In Figure 7 it can be seen that the portfolio variance (which involves the non-systematic risk) tends to zero as the number of assets increase.[1]

[15] See Appendix 1.

[16] Their covariance matrix and expected returns vector can be seen in Appendix 2.

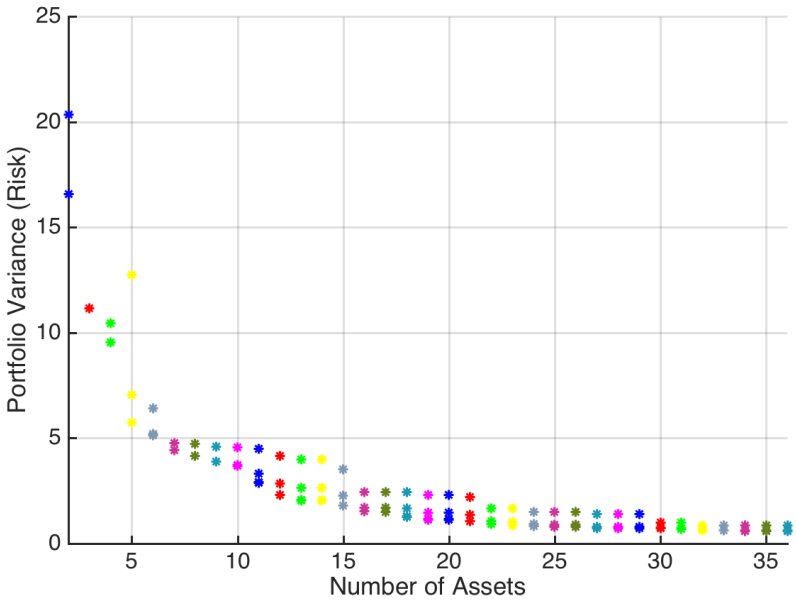


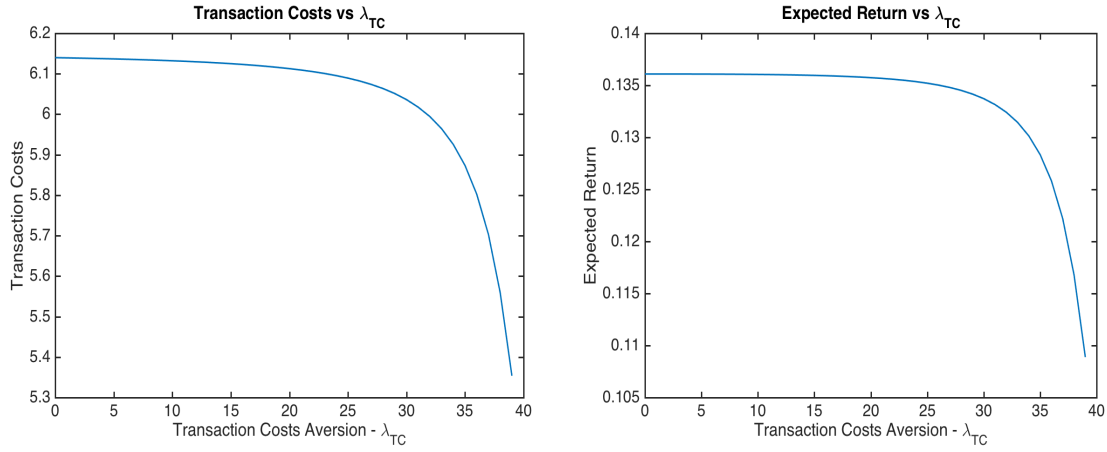
Figure 7: Portfolio Variance as the number of included assets increases.

6.4 Transaction Costs Aversion Coefficient Effect

For the final individual experiment the **Max Return** algorithm including the transaction costs constraints was used.[17] The purpose was to observe the effect of the transaction costs aversion coefficient (`ltc`) on the optimal decision that the optimisation outputs. It is known that as it increases the program weighs more the transaction costs. The transaction costs model coefficients were: \$1 for `alpha` and 0.5% for `beta`.[18] For this reason the algorithm was ran for fifty times for (`ltc`) increasing from 0 to 49. As expected, as (`ltc`) increased the optimisation was outputting smaller overall trade moves, reducing the transaction costs volume (Figure 8.a) and the overall return (Figrure 8.b).

[17] See Appendix 1.

[18] This pricing model was taken from the website of electronic brokerage company *Interactive Brokers*.[11]



(a) (b)

Figure 8: The effect of the increasing transaction costs aversion coefficient: (a) on the transaction costs volume and (b) on the overall expected return.

7 Real Life Case Study

In this section the algorithm was ran in its full version resembling real life situations.[19] The benchmark effect, the transaction costs and the "discretisation" code, along with the **Max Return** optimisation were all put together. Both cases of either starting a new investment position or updating an old one were explored. The same assets as before were used (figures illustrated in Tables 1 & 2) as if a move was to be made on the 15th of June 2016.

7.1 Starting New Investment Position

The set-up used for the representation of the "starting a new investment position" case can be seen in the table below:

Name	Symbol	Value
Maximum Accepted Variance	`g`	0.0005
Transaction Costs Aversion coefficient	`ltc`	5
Maximum Distance from Benchmark	`d_max`	0.01
Total Budget (uninvested)	`c`	$10000
Number of Assets previously held	`num0`	0 per asset

Table 3: Set-up for the new investment case.

[19] Shown fully commented in Appendix 1.

The results after the execution of the code are illustrated below:

Name	Symbol	Value
Number of each asset bought	`num`	$[24, 36, 76]$
Transaction Costs amount spent	`tc`	\$52.1338
Total Budget Invested (final)	`numt'*p`	\$9826.8
Total Budget Uninvested (final)	`bt-(e'*chi+tc)`	\$121.1062
New Portfolio Return	`y1`	18.37%

Table 4: Results for new investment case.

7.2 Portfolio Update

The set-up used for the representation of the updating of an already existing invested position case can be seen in the table below:

Name	Symbol	Value
Maximum Accepted Variance	`g`	0.0004
Transaction Costs Aversion coefficient	`ltc`	5
Maximum Distance from Benchmark	`d_max`	0.06
Budget (uninvested)	`c`	\$2000
Number of Assets previously held	`num0`	$[100, 50, 25]$
Budget Invested (initially)	`num0'*p`	\$11354

Table 5: Set-up for the portfolio update case.

The results after the execution of the code are illustrated below:

Name	Symbol	Value
Quantities Update Vector	`num`	$[-87, 82, 83]$
Transaction Costs amount spent	`tc`	\$68.7351
Number of Assets Invested (final)	`numt`	$[13, 132, 108]$
Total Budget Invested (final)	`numt'*p`	\$13147
Total Budget Uninvested (final)	`bt-(e'*chi+tc)`	\$138.2349
New Portfolio Return	`y1`	20.71%

Table 6: Results for the portfolio update case.

8 Future Work

As far as future work is concerned the algorithm could generally become faster requiring less computational effort. This will also help for the instances at which some money is undesirably left uinvested. With a faster algorithm recursive methods could be applied in order to get as close as possible to the optimal result. Another addition is to make the extraction of past figures and statistics from the web via a specially programmed script.

Transaction Costs In the transaction costs modelling an additional quadratic term could be added. In this way it would resemble real world complex pricing patterns. For example the case of a huge demand of a particular asset could be incorporated with a threshold on the number of items (i.e. shares). The algorithm could also become available on treating piecewise transaction costs functions. A way to achieve this is by making the indicator function, instead of only buy/sell/stay, have more segments that cover specific percentages on the total transaction volume.

Derivatives The algorithm could be made to handle derivatives too. Extra code part will be needed in order to handle the complex derivatives' return functions. The comparably easier ones to be modelled are the *futures contracts*.[1]

9 References

[1] Berk, J. B., DeMarzo, P. M., & dawsonera. *Corporate Finance* (Third edition, Global ed.). Pearson, Boston, Mass., 2014.

[2] Meir Statman. *How Many Stocks Make a Diversified Portfolio?*. Journal of Financial and Quantitative Analysis, Vol. 22, No. 3. (Sep., 1987), pp. 353-363.

[3] A.S. Suresh. *A Study on Fundamental and Technical Analysis*. International Journal of Marketing, Financial Services & Management Research, Vol. 2, No. 5. (May, 2013), pp. 44-59.

[4] Yingzi Zhu, Guofu Zhou. *Technical analysis: An asset allocation perspective on the use of moving averages*. Journal of Financial Economics, Vol. 92. (Mar., 2009), pp. 519-544.

[5] Shradha Tulankar, Rahul Athale, Sandeep Bhujbal. *Sentiment Analysis of Equities using Data Mining Techniques and Visualizing the Trends*. International Journal of Computer Science Issues, Vol. 10, Issue 4, No. 2. (Jul., 2013), pp. 265-269.

[6] Bruno H. Solnik. *Why Not Diversify Internationally Rather Than Domestically?*. Financial Analysts Journal, Vol. 30, No. 4. (Jul./Aug., 1974), pp. 48-54.

[7] Frank Fabozzi, Petter Kolm, Dessislava Pachamanova, Sergio Focardi. *Robust Portfolio Optimization and Management*. John Wiley & Sons, Inc., Hoboken, New Jersey, 2007.

[8] Harry Markowitz. *Portfolio Selection*. The Journal of Finance, Vol. 7, No. 1. (Mar., 1952), pp. 77-91.

[9] Douglas W. Allen. *0740 Transaction Costs*. Department of Economics, Simon Fraser University. (1999)

[10] *MarketMaster®and Investment Clubs rates and charges*. Retrieved from https://www.barclaysstockbrokers.co.uk/Accounts/MarketMaster/pages/Rates-and-charges.aspx.

[11] *Stocks, ETFs and Warrants - Fixed Pricing Structure*. Retrieved from https://www.interactivebrokers.co.uk/en/index.php?f=1590&p=stocks1

[12] Lieven Vandenberghe, Stephen Boyd. *Semidefinite Programming*. SIAM Review, Vol. 38, No. 1. (Mar., 1996), pp. 49-95.

[13] Fuzhen Zhang. *The Schur Complement and its Applications*. Springer US, New York City, New York, 2005.

[14] Nestor Gandelman, Ruben Hernandez-Murillo. *Risk Aversion at the Country Level*. Research Division, Federal Reserve Bank of St. Louis, Working Paper Series. (Feb., 2014).

[15] *The Procter and Gamble Company (PG)*. (2016, June 15). Retrieved from https://finance.yahoo.com/q?s=PG.

[16] *KeyCorp. (KEY)*. (2016, June 2015). Retrieved from http://finance.yahoo.com/q?s=KEY.

[17] *Apple Inc. (AAPL)*. (2016, June 15). Retrieved from http://finance.yahoo.com/q?s=AAPL.

[18] *Equities, Procter & Gamble Co*. (2016, June 15). Retrieved from http://markets.ft.com/research/Markets/Tearsheets/Forecasts?s=PG:NYQ.

[19] *Equities, KeyCorp*. (2016, June 15). Retrieved from http://markets.ft.com/research/Markets/Tearsheets/Forecasts?s=KEY:NYQ.

[20] *Equities, Apple Inc.*. (2016, June 15). Retrieved from http://markets.ft.com/research/Markets/Tearsheets/Forecasts?s=AAPL:NSQ.

[21] *S&P 500 Companies by Weight*. (2016, June 15). Retrieved from http://slickcharts.com/sp500.

10 Appendix 1

10.1 MATLAB MaxReturn Optimisation Function

```
function [y0,x]=maxReturn(m,g,y,S,xb,d_max)

cvx_begin sdp

% Vector of ones (according to the number of products)
e=ones(m,1);
% Variance upper bound (to be minimised)
variable y0(1);
% Set vector of choice percentages
variable x(m);

% Maximise the return
maximize(y0);
subject to
% Keep the portfolio variance below the upper bound "g"
[g,x';
    x,inv(S)]>=0;
% All choice percentages add up to 1
e'*x==1;
% Keep the return above the lower bound "y0"
y'*x-y0>=0;
% Keep all choice percentages in the interval [0,1] to make sense
for i=1:m,
    x(i)>=0;
    x(i)<=1;
end
cvx_end

end
```

10.2 MATLAB MinVar Optimisation Function

```
function [g,x]=minVar(m,y0,yy,SS)

cvx_begin sdp

% Vector of ones (according to the number of products)
e=ones(m,1);

% Variance upper bound (to be minimised)
variable g(1);

% Set vector of choice percentages
variable x(m);

% Minimize the variance
minimize(g);
subject to
% Keep the portfolio variance below the upper bound "g"
[g,x';
    x,inv(SS)]>=0;
% All choice percentages add up to 1
e'*x==1;
% Keep the return above the lower bound "y0"
yy'*x-y0>=0;
% Keep all choice percentages in the interval [0,1] to make sense
for i=1:m,
    x(i)>=0;
    x(i)<=1;
end
cvx_end

end
```

10.3 MATLAB Experiment 1 Main

```
clear all; clc;

m=3;         % Number of assets
g=0.00058;    % Portfolio Variance, upper bound (upper to show experiment)
% g=0.0001277;    % Portfolio Variance, upper bound (lower possible)
d_max=0.5;  % Index Difference, upper bound threshold

% Returns
y=[0.0198;0.2017;0.2313];

% Covariance matrix
S=[0.00015241    0.00006108      0.00005548;
    0.00006108   0.00050032      0.00010811;
    0.00005548   0.00010811      0.00057614];


% Vector for the index percentage values
xb=[0.284902524; 0.013046969; 0.702050507];

hold on;
for i=0:159
    g=0.00012768+i*0.000003015;
    [y0,x]=maxReturn(m,g,y,S,xb,d_max);
    if i==0||i==16||i==32||i==48||i==64||i==80||i==96||i==112||i==128||
    i==144
        q1((i+16)/16,1:3)=x';
        qv1(1,(i+16)/16)=g;
        yv1(1,(i+16)/16)=y0;
    end
    plot(g,y0,'*');
end
```

```
title('Efficient Frontier - PG,KEY,AAPL (maxReturn, 160 points)')
xlabel('Risk') % x-axis label
ylabel('Expected Return') % y-axis label
hold off;

figure
bar(q1,1.1);
ylim([0 1]);
xlim([0.5 10.5]);
title('Portfolio Weights - PG,KEY,AAPL - g=0.00012768:0.00056184')
set(gca,'XTickLabel',{'1.2768', '1.7592', '2.2416', '2.7240', '3.2064',
'3.6888', '4.1712', '4.6536', '5.1360', '5.6184'})
xlabel('Max Return Portfolios (variance x10^-^4)') % x-axis label
ylabel('Weights') % y-axis label
legend('PG','KEY','AAPL','Location','north','Orientation','horizontal');
grid on;
```

10.4 MATLAB Experiment 2 Main

```
clear all; clc;

m=3;        % Number of assets
% d_max=0.024961;  % Index Difference, upper bound threshold (lower possible
d_max=0.25;
% Vector of returns
y=[0.0198;0.2017;0.2313];

% Covariance matrix
S=[0.00015241   0.00006108      0.00005548;
    0.00006108  0.00050032      0.00010811;
    0.00005548  0.00010811      0.00057614];
```

```
% Set examples of portfolio variances
for k=0:9
    g(k+1,1)=0.00012768+k*(16*0.000003015);
end

% Vector for the index percentage values
xb=[0.284902524; 0.013046969; 0.702050507];

for i=1:5
    figure
    hold on;
    for j=0:99
        d_max=0.024961+j*0.00247539;
        [y0,x]=maxReturn(m,g(2*i,1),y,S,xb,d_max);
        plot(d_max,y0,'*');
    end
    ylim([0.14 0.235]);
    title('Benchmark Constraint Effect - PG,KEY,AAPL (maxReturn, 100 points)
    xlabel('Distance from Benchmark') % x-axis label
    ylabel('Expected Return') % y-axis label
    hold off;
end
```

10.5 MATLAB Experiment 3 Main

```
clear all; clc;

% Number of assets
m=36;
% Generate a random vector of returns
y=abs(randn(m,1));
% Overall return, lower bound
y0=sum(y)/m; y0p=y0/3;
% Generate a random Covariance matrix
```

```
S=randn(m); S=S*S';
for j=1:5
        y0=j*y0p;
        for i=2:36
                yy=y(1:i,1);
        SS=S(1:i,1:i);
        % Run the minimum variance algorithm
        [g,x]=minVar(i,y0,yy,SS);
                gg(j,i)=g;
        end
end
C = {'m','b','r','g','y',[.5 .6 .7],[.8 .2 .6],[.4 .5 .1],
[.1 .6 .7],'m','b','r','g','y',[.5 .6 .7],[.8 .2 .6],
[.4 .5 .1],[.1 .6 .7],'m','b','r','g','y',[.5 .6 .7],
[.8 .2 .6],[.4 .5 .1],[.1 .6 .7],'m','b','r','g','y',
[.5 .6 .7],[.8 .2 .6],[.4 .5 .1],[.1 .6 .7]};
grid on;
hold on;
for k=2:36
    plot(k,gg(:,k),'*','Color',C{k});
end
xlim([2 36]);
xlabel('Number of Assets') % x-axis label
ylabel('Portfolio Variance (Risk)') % y-axis label
hold off;
```

10.6 MATLAB Experiment 4 Main

```
clear all; clc;

m=3;        % Number of assets
g=0.00012768;    % Portfolio Variance, upper bound
% Generate vector of ones
e=ones(m,1);
```

```
% Returns
y=[0.0198;0.2017;0.2313];

% Covariance matrix
S=[0.00015241   0.00006108      0.00005548;
    0.00006108  0.00050032      0.00010811;
    0.00005548  0.00010811      0.00057614];

%--------------------INITIAL-PORTFOLIO----------------------------
% Prices vectors
p=[83.35; 11.65; 97.46];

% Total budget;
bt=10000;

%--------------------T.COSTS-QUADRATIC-MODEL----------------------
% Generate alpha vector
alpha=1*ones(m,1);

% Generate beta vector
beta=0.005*ones(m,1);
%-----------------------------------------------------------------
%--------------------OPTIMISATION---------------------------------
z=ones(m,1);
for k=0:49
    % Transaction Costs Aversion Coefficient
    ltc=1*k;
    [x,y0,tc]=maxReturn(bt,m,g,y,S,alpha,beta,z,ltc);
    o(1,k+1)=y0;
    o(2,k+1)=ltc;
    o(3,k+1)=tc;
    o(4:m+3,k+1)=x(1:m,1);
```

```
end
plot(o(2,1:40),o(1,1:40));
title('Expected Return vs \lambda_T_C')
xlabel('Transaction Costs Aversion - \lambda_T_C') % x-axis label
ylabel('Expected Return') % y-axis label
figure
plot(o(2,1:40),o(3,1:40));
title('Transaction Costs vs \lambda_T_C')
xlabel('Transaction Costs Aversion - \lambda_T_C') % x-axis label
ylabel('Transaction Costs') % y-axis label
```

10.7 MATLAB Experiment 4 MaxReturn

```
function [x,y0,tc]=maxReturn(bt,m,g,y,S,alpha,beta,z,ltc)

cvx_begin sdp

% Vector of ones (according to the number of products)
e=ones(m,1);

% Overall Return lower bound (to be maximised)
variable y0(1);

% Set vector for portfolio weights
variable x(m);

% Maximise the overall return
maximize(y0)
subject to
%
y'*x-ltc*((alpha/bt)'*abs(z))-(ltc*beta'*(z.*x))-y0>=0;
% Keep the portfolio variance below the upper bound "g"
[g,x';
    x,inv(S)]>=0;
```

```
%
x'*e*bt+alpha'*abs(z)+bt*beta'*(z.*x)<=bt;
%
for jj=1:m
    x(jj)>=0;
    x(jj)<=1;
end
%
e'*x>=0;
e'*x<=1;
%
for ii=1:m
    if z(ii)==0
        x(ii)==0;
    elseif z(ii)==1
        x(ii)>=0;
        x(ii)<=1;
    elseif z(ii)==-1
        x(ii)<=0;
        x(ii)>=-1;
    end
end
cvx_end
tc=alpha'*abs(z)+beta'*(z.*x)*bt;
y0=y'*x;
end
```

10.8 MATLAB Case Study Main

```
clear all; clc;

m=3;          % Number of assets
ltc=5;       % Transaction Costs Aversion Coefficient
g=0.0004;    % Portfolio Variance, upper bound
```

```
% Generate vector of ones
e=ones(m,1);

% Set vector of returns
y=[0.0198;0.2017;0.2313];

% Set the covariance matrix
S=[ 0.00015241   0.00006108      0.00005548;
    0.00006108   0.00050032      0.00010811;
    0.00005548   0.00010811      0.00057614];

%--------------------INITIAL-PORTFOLIO-----------------------------
% Set initial portfolio asset numbers
num0=[100;50;25];
% num0=zeros(m,1);

% Set the considered asset prices vector
p=[83.35; 11.65; 97.46];

 % Unused Extra Budget
c=2000;

% Budget initially invested
bt0=num0'*p;

% Total budget;
bt=bt0+c;

% Set initial portfolio weights
x0=num0.*p/bt;
% x0=zeros(m,1);
```

```
%--------------------T.COSTS-QUADRATIC-MODEL----------------------
% Generate alpha (constant) vector
alpha=1*ones(m,1);

% Generate beta (percentage) vector
beta=0.005*ones(m,1);

%--------------------INDEX----------------------------------------
% Set vector for the index percentage values
xb=[0.284902524; 0.013046969; 0.702050507];

% Index Difference, upper bound threshold
d_max=0.06;

%-------------------------------------------------------------------
%--------------------OPTIMISATION---------------------------------

% Set up the indicator function vector pool
zz=[-1,0,1]';
zzz=allcomb(zz,zz,zz);

% Run the optimal portfolio function for each indicator function vector
for ii=1:3^m
    z=zzz(ii,:)';
    [y1,xd,tc,chi,num,numt]=OptPUfunc(bt,m,g,x0,y,p,num0,S,alpha,beta,
    z,xb,d_max,ltc);
    % Store the results
    o(1,ii)=y1;
    o(2:m+1,ii)=xd(1:m,1);
    o(m+2:2*m+1,ii)=chi(1:m,1);
    o(2*m+2:3*m+1,ii)=num;
    o(3*m+2:4*m+1,ii)=numt;
    o(4*m+2,ii)=tc;
```

```
    o(4*m+3:5*m+2,ii)=z;
end

% Find the case where the overall return is maximum
qq(:,1)=o(:,1);
for jj=1:(3^m)-1
    if o(1,jj+1)>o(1,jj)
        qq(:,1)=o(:,jj+1);
    end
end

y1=qq(1,1);              % Store the overall expected return
xd=qq(2:m+1,1);          % Store the update portfolio weights vector
chi=qq(m+2:2*m+1,1);     % Store the updated portfolio amounts vector
num=qq(2*m+2:3*m+1,1);   % Store the traded asset quantities vector
numt=qq(3*m+2:4*m+1,1); % Store the updated invested asset quantities
tc=qq(4*m+2,1);          % Store the total transaction costs
z=qq(4*m+3:5*m+2,1);     % Store the chosen indicator function vector
```

10.9 MATLAB Case Study `OptPUfunc` Function (Max Return full vers.)

```
function [y1,xd,tc,chi,num,numt]=OptPUfunc(bt,m,g,x0,y,p,num0,S,alpha,beta,
z,xb,d_max,ltc);

cvx_begin sdp

% Vector of ones (according to the number of products)
e=ones(m,1);

% Overall Return lower bound (to be maximised)
variable y0(1);

% Set vector of differences
variable xd(m);
```

```
%-------------------------------------------------------------------

% Maximise the overall return
maximize(y0)
subject to

% Maximise the expression of returns , "damaged" by transaction costs
y'*(x0+xd)-ltc*(alpha/bt)'*abs(z)-ltc*beta'*(z.*xd)-y0>=0;


% Keep the portfolio variance below the upper bound "g"
[g,(x0+(xd))';
    (x0+(xd)),inv(S)]>=0;

% The line below is equivalent to the:
% The amount to be sold minus the amount to be bought
% less or equal to the initially uninvested amount
(x0+xd)'*e*bt+alpha'*abs(z)+bt*beta'*(z.*xd)<=bt;

% Final weights have reasonable values
for jj=1:m
    x0(jj)+xd(jj)>=0;
    x0(jj)+xd(jj)<=1;
end

% Final overall portfolios have reasonable sums
e'*(x0+xd)>=0;
e'*(x0+xd)<=1;

% Indicator Function perform as stated
for ii=1:m
    if z(ii)==0
        xd(ii)==0;
```

```
    elseif z(ii)==1
        xd(ii)>=0;xd(ii)<=1;
    elseif z(ii)==-1
        xd(ii)<=0;xd(ii)>=-1;
    end
end


% Track the index with a difference upper bound "d_max"
[d_max,(x0+xd-xb)';
    (x0+xd-xb),eye(m)]>=0;

cvx_end


% Transaction Costs (before discretisation)
tc0=alpha'*abs(z)+beta'*(z.*xd)*bt;

%--------DISCRETISATION-------------------------------------------------
for i=1:m
    j=0;
    % Stay on the line (#assets)x(price)
    % Increase number until suggested portion is met
    while j*p(i)<(x0(i)+xd(i))*(bt-tc0)
        j=j+1;
    end
    % Store the below-level amount only, on chi_d
    if j<=0
        chi(i,1)=0;
    else
        if z(i)==0
            chi(i,1)=num0(i)*p(i);
        elseif z(i)==1 || z(i)==-1
```

```
            chi(i,1)=(j-1)*p(i);
        end
    end

end

% Find the number of items per asset vector
numt=chi./p;
% Calculate the update quantities vector
num=numt-num0;
% Calculate final transaction costs
tc=alpha'*abs(z)+beta'*abs(chi);

% Return final overall return changed to the new traded amounts
y1=(chi'*y)/bt;

end
```

11 Appendix 2

0.620090051558262
0.286673862460466
0.598015714251220
0.245532675353110
1.78073717512705
2.34723922894497
1.71359529882238
0.237127260967104
0.619625814831921
0.720160379657291
0.0406571736321824
0.658981428583564
0.630515032828478
0.609625091181260
0.782334970498853
2.43658435807580
0.302407416537707
0.0583198538886379
0.574133962393711
0.195212401051107
0.0505312775255591
1.75577504072859
0.257357623581956
0.749541885371127
0.570764308999752
0.494233375953117
0.991440438825251
1.07714011343418
0.776841798206994
2.25984035153099
0.564376691130426
0.901491435092922
0.394675531568292
0.00485442371891243
0.436918628028167
1.13007271999667

Figure 9: Expected Returns vector for 36 randomised assets.

42878	-9.6317	-5.8943	1.8088	-17.98	-3.9589	-2.7067	1.1546	4.6604	6.7759	-11237	2.3842	-6.3976	-1.0881	2.1926	-1.1736	9.6218	0.93391	-6.2551	5.2508	4.0638	5.5737	-11172	-0.98951	-10348	-1.9673	7.3103	5.1415	8.6344	4.0731	-9.8102	0.17025	-5.7098	-1.4899	2762	2.5741
-9.6317	42696	6.2071	6.0564	-0.85847	8.0369	-12656	1.3868	4.3133	-8.3589	-0.88539	-5.1291	5.6742	8.7815	5.0588	-8635	-6.4521	1.2077	0.81123	2.8352	3.1756	1.2857	10497	-1.4658	8.2499	2.1704	-9.5166	5322	0.43701	0.53627	0.67477	7.3641	1.3517	-9.8426	-6.7606	-6.01
-5.8943	6.2071	34791	3.9894	-1.8478	-6.1191	7.8896	-0.10227	2.3152	-6.7766	-0.53282	-3.9232	2.2134	2.0618	-9.1702	4.0129	-5.5184	-2.4412	-0.10888	0.31738	-4.8357	-7.8508	12204	10792	-7451	1.44	-2.2568	3.3612	0.62842	6.7346	-6.4506	0.45563	2.0687	-2.0275	-6.9429	6626
1.8088	6.0564	3.9894	29974	1.7662	-5.4324	11046	-1.2356	3.5014	7.8489	-5.8333	2.8847	2.5567	6.1399	-1.6498	-0.49309	2.1515	2.9222	3.1125	3.3933	-7.8046	-7.0317	-5.7159	-7.6567	3.9212	-2.3028	-9892	2217	-3.4244	-0.79201	-9.9551	-11068	0.62995	-5.6509	2.1458	1.8335
-17.98	-0.85847	-1.8478	1.7662	45548	9.7682	4.4007	1.83	-2.6399	4.1197	-0.47113	-2168	5912	2343	2.0921	-4.9289	-2.7119	0.065108	3.9762	3.0119	-4.6986	-5.5828	9998	0.15933	-0.10357	-5.7681	-4.9556	-13005	4.3126	3.2316	3.2556	-10741	4.0042	-5373	0.49863	8.2785
-3.9589	8.0369	-6.1191	-5.4324	9.7682	41597	-10523	7.6768	3.6308	2.6857	4.3215	-1.2908	0.042719	11717	-5.0899	7.5772	2415	-2236	-1.0708	1.5955	4.4383	5.4432	2.8831	-4.7936	-9.6187	-3.8423	0.1458	-4.0117	1.5054	5.9702	5296	-2.1346	-2.3546	-1.5324	1436	-11312
-2.7067	-12656	7.8896	11046	4.4007	-10523	42821	-4.2987	-3.1125	3.7244	2465	2.5198	-2.1065	7.3533	-2.6371	6.2785	13014	-1.8211	-3.1958	3.7558	-18053	-12115	-7.2695	-7.9502	13419	5.2949	-1.2605	-0.93033	-7.2854	-5.7493	-11882	-4.0881	-1.6792	3.8716	5.1835	7.0535
1.1546	1.3868	-0.10227	-1.2356	1.83	7.6768	-4.2987	41664	-2.2709	-8.6566	0.1263	1231	0.92625	6.7432	4.6076	11903	10.91	-6.0648	4415	7.6556	-0.20381	-2234	1023	7686	-2.8135	8.0995	1.9149	1.7045	-7.6103	1.9132	1.3458	8.0436	6438	-5.5675	0.21667	2.5235
4.6604	4.3133	2.3152	3.5014	-2.6399	3.6308	-3.1125	-2.2709	27.09	8.3304	5.3402	-6.1966	-2.8828	5.6704	1.9058	-3.3143	-3.2885	6279	-6.3475	2.3614	-8.1195	3.1848	-3.3128	-6.0588	1.2591	3.3142	-3.4532	7.7485	-3.0897	2.6052	5.8192	11949	-1.7902	1.8011	-6.0511	3.1625
6.7759	-8.3589	-6.7766	7.8489	4.1197	2.6857	3.7244	-8.6566	8.3304	44619	10.23	-3375	-4.9031	3.3284	-2.7663	-2997	3.9101	5.8179	0.40829	0.93675	0.062956	-2.0827	-7.0007	-7.8762	-13.84	1.2586	-5.0565	-8225	-0.15275	1.9146	-5.3121	-7.7447	-6.3773	5.6435	6.5002	5.55
-11237	-0.88539	-0.53282	-5.8333	-0.47113	4.3215	2465	0.1263	5.3402	10.23	28053	-7.3181	-1777	7.5945	-3.7127	3.9221	-9.1436	-1.5777	0.29083	-9233	4.0204	-1.8994	2.9116	2.4317	-4.5798	-0.66773	4386	0.44937	-2.3639	-1.3878	8.8222	3572	0.2569	6.2701	3.5012	-0.13977
2.3842	-5.1291	-3.9232	2.8847	-2168	-1.2908	2.5198	1231	-6.1966	-3375	-7.3181	34347	-4.4292	-7.1421	-2434	-10552	6.0683	6.8207	-9.0925	1.7123	7828	-5.9948	-6.2629	-2.8342	5.2781	3.4041	-8.8475	-2257	-13678	0.93951	0.38578	2.0672	11119	0.20023	14914	-8.8896
-6.3976	5.6742	2.2134	2.5567	5912	0.042719	-2.1065	0.92625	-2.8828	-4.9031	-1777	-4.4292	35445	-9902	4.6152	3.1707	-8.0276	4.9056	4.8618	-7.8772	-2.0982	7821	6.6663	-3187	-15618	-6.0037	-0.036794	-4.5454	-6.8609	-4.7188	-5.4153	-5.4092	-5.1904	-8.9414	-5.1768	-9.8843
-1.0881	8.7815	2.0618	6.1399	2343	11717	7.3533	6.7432	5.6704	3.3284	7.5945	-7.1421	-9902	31314	-2.6894	1.2834	1.1889	-9.0341	-4.9632	2.8694	0.36589	-10312	-2.2519	-6034	0.13568	-0.35532	5.5318	0.62627	-3.4373	5.7016	4.9347	1.0297	7.39	0.61767	3.3432	0.38086
2.1926	5.0588	-9.1702	-1.6498	2.0921	-5.0899	-2.6371	4.6076	1.9058	-2.7663	-3.7127	-2434	4.6152	-2.6894	28989	-1.4159	9.0667	-3258	-4.6827	6.3222	3.6721	-0.65975	-1.4715	1.8748	2.9308	5.0835	-4.7455	3.4426	2.1614	-8.2586	12.48	6.3912	5.2452	-2.9178	1.0729	10864
-1.1736	-8635	4.0129	-0.49309	-4.9289	7.5772	6.2785	11903	-3.3143	-2997	3.9221	-10552	3.1707	1.2834	-1.4159	34391	0.95247	-6.0769	5.7986	-0.9131	-7209	4.9124	-1.7737	0.7036	-5.4427	4.6255	6.2685	6.9869	6.8295	-8.1611	1.9161	-5.4027	-1.6495	0.48333	-6.0321	4.1892
9.6218	-6.4521	-5.5184	2.1515	-2.7119	2415	13014	10.91	-3.2885	3.9101	-9.1436	6.0683	-8.0276	1.1889	9.0667	0.95247	45931	-9.9286	-8.9713	6.5243	-7.9585	2.33	-8.8126	-1.6038	17714	15775	-1.5729	-2.4019	-10867	-2.1186	-5745	3.5401	-0.79419	4.6676	1.3084	3.0535
0.93391	1.2077	-2.4412	2.9222	0.065108	-2236	-1.8211	-6.0648	6279	5.8179	-1.5777	6.8207	4.9056	-9.0341	-3258	-6.0769	-9.9286	18889	7.1665	1784	-1.1542	2.8242	-1.6164	-3803	2.5025	-4.4885	-7.8693	-0.21099	-0.22507	-1.1648	-1.8174	-0.1955	-4.6828	0.18676	-0.022348	-1.0364
-6.2551	0.81123	-0.10888	3.1125	3.9762	-1.0708	-3.1958	4415	-6.3475	0.40829	0.29083	-9.0925	4.8618	-4.9632	-4.6827	5.7986	-8.9713	7.1665	31546	1.1365	-3.2938	2894	3.3814	2.4806	2854	-7.1483	-0.70362	-3.6498	11963	3.8163	-4.0936	-7373	-8.2749	5.4559	-2.7639	-2.6169
5.2508	2.8352	0.31738	3.3933	3.0119	1.5955	3.7558	7.6556	2.3614	0.93675	-9233	1.7123	-7.8772	2.8694	6.3222	-0.9131	6.5243	1784	1.1365	23973	2.9804	-2.6371	-1.2188	-1.7893	8.1676	11456	-3.0269	4.1516	2.1174	-2.7851	1.7013	6.8808	3.0033	-5.7616	5.0573	4.7021
4.0638	3.1756	-4.8357	-7.8046	-4.6986	4.4383	-18053	-0.20381	-8.1195	0.062956	4.0204	7828	-2.0982	0.36589	3.6721	-7209	-7.9585	-1.1542	-3.2938	2.9804	37019	0.32258	6.5048	4.4004	-5.7201	0.84514	8.1209	1.5375	-1.0336	2.2691	12648	-1.3057	2.1371	1.7224	17173	-9.0196
5.5737	1.2857	-7.8508	-7.0317	-5.5828	5.4432	-12115	-2234	3.1848	-2.0827	-1.8994	-5.9948	7821	-10312	-0.65975	4.9124	2.33	2.8242	2894	-2.6371	0.32258	28021	-3478	-2.1309	-2142	-1.8299	3.0813	7.7917	6.4943	0.14126	-2.3607	0.46373	-8.4806	3.4608	1.3598	-6732
-11172	10497	12204	-5.7159	9998	2.8831	-7.2695	1023	-3.3128	-7.0007	2.9116	-6.2629	6.6663	-2.2519	-1.4715	-1.7737	-8.8126	-1.6164	3.3814	-1.2188	6.5048	-3478	32932	13864	-4.2985	-4.9806	0.056801	-2.9752	7.9012	6.8336	0.41624	2.8223	-7513	-4.2538	-4.3976	-0.21228
-0.98951	-1.4658	10792	-7.6567	0.15933	-4.7936	-7.9502	7686	-6.0588	-7.8762	2.4317	-2.8342	-3187	-6034	1.8748	0.7036	-1.6038	-3803	2.4806	-1.7893	4.4004	-2.1309	13864	30304	-4.05	3.0987	-2.5425	0.24887	11171	2.8429	2.1365	-0.089058	-5.0203	-3681	-1.7093	14692
-10348	8.2499	-7451	3.9212	-0.10357	-9.6187	13419	-2.8135	1.2591	-13.84	-4.5798	5.2781	-15618	0.13568	2.9308	-5.4427	17714	2.5025	2854	8.1676	-5.7201	-2142	-4.2985	-4.05	69208	22474	2.2421	4.7849	-7.8555	-9.3925	3.7214	9.8465	-4.0238	11182	-2.3619	4.0739
-1.9673	2.1704	1.44	-2.3028	-5.7681	-3.8423	5.2949	8.0995	3.3142	1.2586	-0.66773	3.4041	-6.0037	-0.35532	5.0835	4.6255	15775	-4.4885	-7.1483	11456	0.84514	-1.8299	-4.9806	3.0987	22474	45393	13.45	6.1523	-11394	-7.4402	0.66598	8.5252	2.3113	0.90675	0.80416	4.9531
7.3103	-9.5166	-2.2568	-9892	-4.9556	0.1458	-1.2605	1.9149	-3.4532	-5.0565	4386	-8.8475	-0.036794	5.5318	-4.7455	6.2685	-1.5729	-7.8693	-0.70362	-3.0269	8.1209	3.0813	0.056801	-2.5425	2.2421	13.45	38897	3.1015	0.46668	2.3165	0.27625	-1.8891	1.5666	8.1434	-0.66799	-2.4661
5.1415	5322	3.3612	2217	-13005	-4.0117	-0.93033	1.7045	7.7485	-8225	0.44937	-2257	-4.5454	0.62627	3.4426	6.9869	-2.4019	-0.21099	-3.6498	4.1516	1.5375	7.7917	-2.9752	0.24887	4.7849	6.1523	3.1015	24793	5.6312	-8.9267	6.0224	11489	0.10374	-2.8534	3.1185	-0.23423
8.6344	0.43701	0.62842	-3.4244	4.3126	1.5054	-7.2854	-7.6103	-3.0897	-0.15275	-2.3639	-13678	-6.8609	-3.4373	2.1614	6.8295	-10867	-0.22507	11963	2.1174	-1.0336	6.4943	7.9012	11171	-7.8555	-11394	0.46668	5.6312	37375	4.5922	1.8955	-10311	-2139	-1.3665	0.13804	11919
4.0731	0.53627	6.7346	-0.79201	3.2316	5.9702	-5.7493	1.9132	2.6052	1.9146	-1.3878	0.93951	-4.7188	5.7016	-8.2586	-8.1611	-2.1186	-1.1648	3.8163	-2.7851	2.2691	0.14126	6.8336	2.8429	-9.3925	-7.4402	2.3165	-8.9267	4.5922	26603	-3.3017	-6.8525	1.1598	8809	0.19698	4.6427
-9.8102	0.67477	-6.4506	-9.9551	3.2556	5296	-11882	1.3458	5.8192	-5.3121	8.8222	0.38578	-5.4153	4.9347	12.48	1.9161	-5745	-1.8174	-4.0936	1.7013	12648	-2.3607	0.41624	2.1365	3.7214	0.66598	0.27625	6.0224	1.8955	-3.3017	35.02	10.88	16843	-0.79196	0.29369	6.2087
0.17025	7.3641	0.45563	-11068	-10741	-2.1346	-4.0881	8.0436	11949	-7.7447	3572	2.0672	-5.4092	1.0297	6.3912	-5.4027	3.5401	-0.1955	-7373	6.8808	-1.3057	0.46373	2.8223	-0.089058	9.8465	8.5252	-1.8891	11489	-10311	-6.8525	10.88	34842	1.0253	-1.1568	-3.7431	-4.8208
-5.7098	1.3517	2.0687	0.62995	4.0042	-2.3546	-1.6792	6438	-1.7902	-6.3773	0.2569	11119	-5.1904	7.39	5.2452	-1.6495	-0.79419	-4.6828	-8.2749	3.0033	2.1371	-8.4806	-7513	-5.0203	-4.0238	2.3113	1.5666	0.10374	-2139	1.1598	16843	1.0253	42318	-8.3042	4.8521	3659
-1.4899	-9.8426	-2.0275	-5.6509	-5373	-1.5324	3.8716	-5.5675	1.8011	5.6435	6.2701	0.20023	-8.9414	0.61767	-2.9178	0.48333	4.6676	0.18676	5.4559	-5.7616	1.7224	3.4608	-4.2538	-3681	11182	0.90675	8.1434	-2.8534	-1.3665	8809	-0.79196	-1.1568	-8.3042	24932	5.8029	1216
2762	-6.7606	-6.9429	2.1458	0.49863	1436	5.1835	0.21667	-6.0511	6.5002	3.5012	14914	-5.1768	3.3432	1.0729	-6.0321	1.3084	-0.022348	-2.7639	5.0573	17173	1.3598	-4.3976	-1.7093	-2.3619	0.80416	-0.66799	3.1185	0.13804	0.19698	0.29369	-3.7431	4.8521	5.8029	35434	-9.5569
2.5741	-6.01	6626	1.8335	8.2785	-11312	7.0535	2.5235	3.1625	5.55	-0.13977	-8.8896	-9.8843	0.38086	10864	4.1892	3.0535	-1.0364	-2.6169	4.7021	-9.0196	-6732	-0.21228	14692	4.0739	4.9531	-2.4661	-0.23423	11919	4.6427	6.2087	-4.8208	3659	1216	-9.5569	41417

Figure 10: Covariance Matrix for 36 randomised assets.