Understanding trade-offs in optimisation: Linear and Quadratic Optimisation.

Researcher: Junayd Ismail

University Of Sussex - BSc Economics

ji99@sussex.ac.uk

Supervisor: Pawel Dziewulski

University Of Sussex - Senior Lecturer in Economics

p.k.dziewulski@sussex.ac.uk

(June - August 2024)

Abstract

This study explores the limitations and benefits of linear and quadratic programming in portfolio optimisation by leveraging computational rigour and mathematical robustness. Transforming a well-known quadratic optimisation algorithm into a linear model, we aim to provide a comparative analysis of both methods data from the S&P 500. The analysis highlights and explains the reasoning behind the inherent biases of linear programming, which often leads to non-diversified portfolios as well as the quadratic programmes superior performance in balancing risk and returns, resulting in more diversified portfolios and why this is important to industry workers in finance and tech sectors.

Key Findings

1. **Linear Programming Bias**:
    - The linear programming model tends to concentrate investments in a single stock due to the imbalanced scaling of variables within the objective function. This issue arises because the linear model struggles to adequately balance mean absolute deviations and mean returns.
2. **Quadratic Programming Superiority**:
    - Quadratic programming effectively accounts for the complexities of stock correlations through the covariance matrix, resulting in a well-diversified portfolio. This model demonstrates the ability to also balance between the preferences of a trader and determine outputs that will accurately depict these pretences, whereas the linear programme tends to struggle with this in the case of maximising returns and minimising variance.

3. **Computational Efficiency**:

    - While the linear model is computationally more efficient, taking on average 0.13 seconds to solve, it fails to achieve the same level of accuracy and robustness as the quadratic model, which takes about 0.4 seconds. The quadratic model's longer computation time is justified by its significantly better performance and accurate representation of portfolio risk.

Understanding the trade-offs between linear and quadratic programming in portfolio optimisation is crucial for financial decision-making. The findings illustrate that while linear programming offers computational speed, it lacks the necessary complexity to handle real-world financial data effectively. Quadratic programming, despite its higher computational demand, provides a more accurate and reliable method for optimising portfolios, balancing risk, and maximising returns. This study reinforces the need to choose the appropriate optimisation model based on the specific requirements of the financial application, ensuring that investment strategies are both efficient and effective.

This comparative analysis contributes to the ongoing discourse on portfolio optimisation, emphasising the importance of aligning mathematical models with practical financial needs to achieve optimal investment outcomes.

Structure of the Study

We begin with a detailed review of the theoretical underpinnings of linear and quadratic programming, including their mathematical formulations and computational considerations. Following this, we delve into the methods used for portfolio optimisation, highlighting the role of convex optimisation [1] in shaping these models. The core of the essay presents the comparative analysis, where we apply both LP and QP models to the S&P 500, detailing the process of model formulation, optimisation. Finally, we discuss the results, offering insights into the practical implications of our findings and understanding the use of standardised models.

Introduction

Optimisation is a fundamental aspect of decision-making in various fields, including economics, finance, and engineering. Among the many techniques available, linear and quadratic programming stand out due to their wide applicability and theoretical elegance. This essay aims to provide a comparative analysis of these two optimisation methodologies, focusing on their practical implementation and performance in the context of portfolio optimisation.

Linear programming (LP) and quadratic programming (QP) are powerful tools for solving optimisation problems with linear and quadratic objective functions, respectively. LP deals with problems where both the objective function and constraints are linear, while QP extends this to include quadratic terms in the objective function, allowing for the modelling of more complex relationships such as those found in financial risk management.

Objective Functions and Constraints

In optimisation problems, the objective function is the mathematical expression that needs to be maximised or minimised. It represents the goal of the optimisation process. For instance, in portfolio optimisation, the objective function might be to maximise returns and / or minimise risk. In linear programming, this function is a linear combination of decision variables. In quadratic programming,

the objective function includes quadratic terms, which allows for the consideration of more intricate relationships among variables.

Constraints are the conditions that the solution must satisfy. These are usually inequalities or equalities that define the feasible region within which the optimal solution lies. Constraints ensure that the solution adheres to certain rules or limits, such as budget restrictions, resource limitations, or specific requirements in portfolio optimisation like not investing more than a certain percentage in any single asset.

The choice of objective function and constraints directly impacts the optimisation results. For example, a poorly defined objective function might lead to suboptimal solutions, while inadequate constraints could result in solutions that are impractical or violate important conditions.

Practical Implications of Optimisation in Finance

The implications of using linear and quadratic programming for optimisation are profound, especially in the realm of portfolio optimisation. Linear programming, with its simpler structure, tends to be more computationally efficient. However, it may oversimplify relationships and fail to capture important aspects of financial risk. Quadratic programming, on the other hand, allows for a more nuanced approach by incorporating the variance-covariance matrix of asset returns into the objective function. This leads to a more accurate representation of risk but at the cost of increased computational complexity.

By comparing linear and quadratic programming models, we can better understand the trade-offs involved. Linear models may provide quicker solutions but might not fully address risk factors and can face limitations, whereas quadratic models offer a detailed risk assessment but require more resources to solve and face less limitations.

Linear Programming

The Basic Linear Programming Model

The classical LP model aims at determining the values of decision variables that maximise or minimise a linear objective function, subject to a set of linear constraints.

A function is made linear by its composition of variables:

$$f(x) = ax + by = \mathbf{Z}$$

This composition, where no variable is equivalent to another (for no squares), is an example of a linear programme, where there are linear equations and no exponentials that can otherwise be quadratic equations.

Example Problem Statement

Consider a company that produces and sells two products: books and calculators. The objective is to determine the number of units of each product to produce to maximise the total sales revenue, subject to production costs and time constraints.

Decision Variables:

Let:

- b be the number of books produced.
- c be the number of calculators produced.

The objective is to maximise the total sales revenue, and the production cost should not exceed $27,000 per month. The total production time available is 43,200 minutes per month. It takes 5 minutes to produce a book and 15 minutes to produce a calculator. The number of books and calculators produced cannot be negative.

Linear Programming Formulation

$$\text{Maximise } S = 20b + 18c$$

subject to

$$5b + 4c \leq 27{,}000$$

$$5b + 15c \leq 43{,}200$$

$$b \geq 0, c \geq 0$$

With this linear optimisation problem, the Simplex method or interior-point methods can be utilised with moderate computational effort to solve. We will be using the interior-point method as it is more beginner friendly, easier to illustrate and typically more efficient.

Matrix Formulation

Linear programming problems can be effectively represented and solved using linear algebra, particularly using matrices. This representation is not only compact but also leverages powerful computational tools from linear algebra.

Objective Function and Constraints in Matrix Form

Let:

- x be the vector of decision variables: $x = \begin{pmatrix} b \\ c \end{pmatrix}$
- c be the vector of coefficients in the objective function: $c = \begin{pmatrix} 20 \\ 18 \end{pmatrix}$
- A be the matrix of coefficients in the constraints: $A = \begin{pmatrix} 5 & 4 \\ 5 & 15 \end{pmatrix}$
- b be the vector of constants on the right-hand side of the constraints: $b = \begin{pmatrix} 27{,}000 \\ 43{,}200 \end{pmatrix}$

Steps to Solve the Problem

To find the optimal solution, we need to identify the feasible region defined by the constraints and then evaluate the objective function within this region. The solution involves finding the intersection

points of the constraints and determining the optimal values of b and c that maximise the sales revenue.

1. Identify the Feasible Region

The feasible region is the set of points that satisfy all the constraints. From these constraints, we can determine the boundary lines:

$$5b + 4c \leq 27{,}000$$

$$b = \frac{27{,}000 - 4c}{5}$$

$$\textit{Assume } c = 0$$

$$b_{intercept} = \frac{27{,}000}{5} = 5400$$

$$5b + 4c \leq 27{,}000 => c = \frac{27{,}000 - 5b}{4}$$

$$\textit{Assume } b = 0$$

$$c_{intercept} = \frac{27{,}000}{4} = 6750$$

$$5b + 15c \leq 43{,}200$$

$$b = \frac{43{,}200 - 15c}{5}$$

$$b_{intercept2} = \frac{43{,}200}{5} = 8640$$

$$c_{intercept2} = \frac{43{,}200}{15} = 2880$$

$$b \geq 0, c \geq 0$$

We express the inequalities as equalities to find the lines that form the boundaries of the feasible region. This step helps in visualising the feasible region on a graph, where the solution must lie.

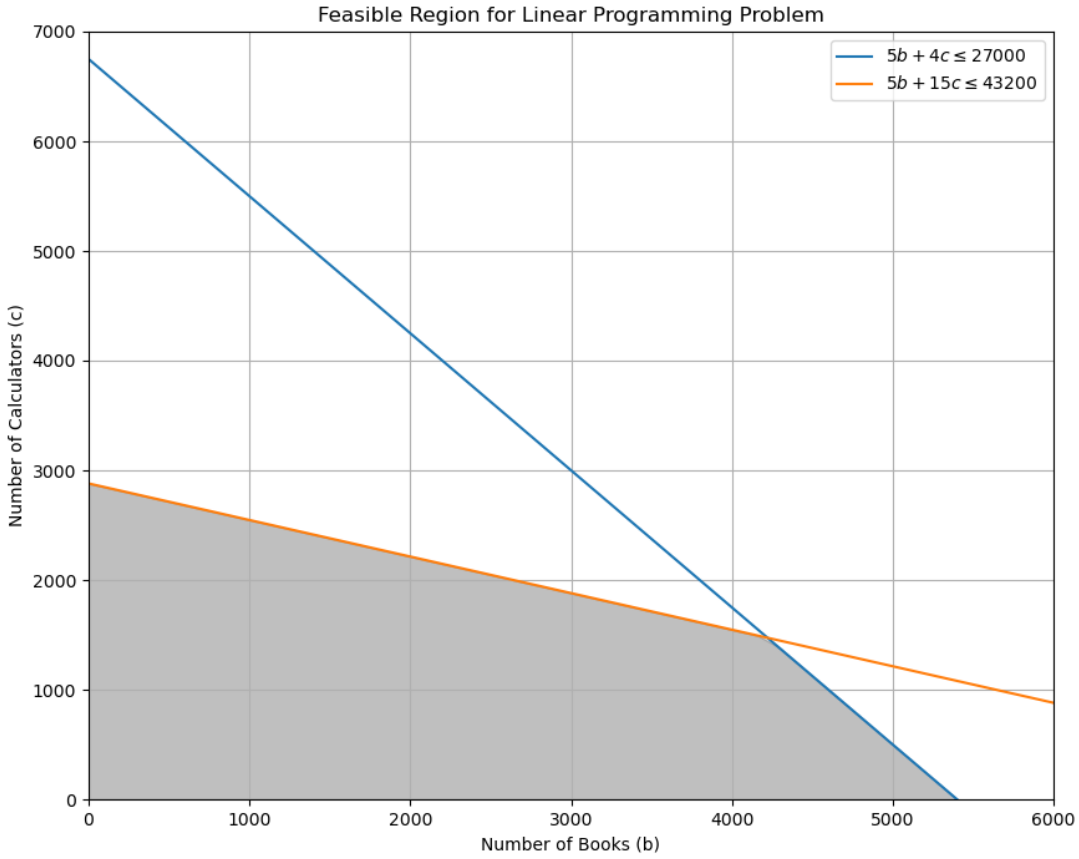


Figure 1: Feasible Region for LP, Interior-point Method: Python (NumPy, Matplotlib).

Optimal Solution:

After solving the system of inequalities, the optimal number of books and calculators to produce are: $b = 4221.82,\ c = 1472.73$. This is represented at the point of intersection of the constraints (4221.82, 1471.73) as represented by (1).

Resulting in maximum sales revenue of: $S = 20(4221.82) + 18(1472.73) = 110{,}945.45$

Benefits of Linear Programming

1. **Efficiency:** Linear programming can handle large-scale problems and provide optimal solutions that satisfy all constraints in a quick process.

By understanding and applying the principles of linear programming, businesses and researchers can optimise their operations, enhance productivity, and achieve better financial performance. The use of matrices in LP further enhances the computational efficiency, making it a powerful tool in both theoretical and applied optimisation problems and this will be relevant particularly in the applied portfolio optimisation further in the study.

Quadratic Programming

The QP model aims at determining the values of decision variables that maximise or minimise a quadratic objective function, subject to a set of linear constraints.

A function is made quadratic by its composition of variables:

$$f(x) = q^T x + \frac{1}{2}x^t$$

This composition, where there can be exponentials of vectors / variables, is an example of a quadratic programme.

Problem Definition

Maximise:

$$\frac{1}{2}(x^2 + y^2) - 2x + 2y$$

Subject to:

$$6x + 4x \leq 24$$

$$x + 2y \leq 6$$

$$y - x \leq 1$$

$$0 \leq y \leq 2$$

$$x \geq 0$$

Steps to Solve the Problem

1. Identify the Feasible Region

The feasible region is the set of points that satisfy all the constraints. From these constraints, we can determine the boundary lines:

$$6x + 4x \leq 24 \text{ => } y = 6 - 1.5x$$

$$x + 2y \leq 6 \text{ => } y = 3 - 0.5x$$

$$y - x \leq 1 \text{ => } y = x + 1$$

$$0 \leq y \leq 2$$

$$x \geq 0$$

We express the inequalities as equalities to find the lines that form the boundaries of the feasible region. This step helps in visualising the feasible region on a graph, where the solution must lie.

2. Plot the Constraints

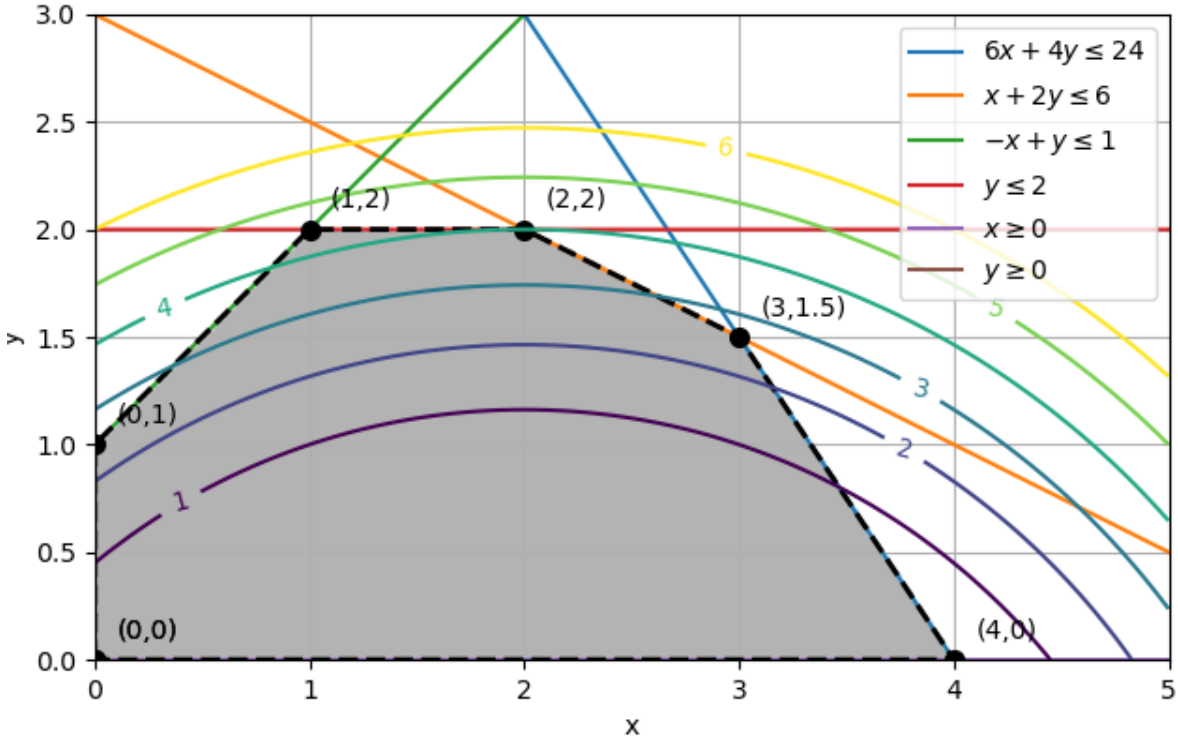


Figure 2: Feasible Region for QP, Interior-point Method: Python (Gekko, Matplotlib).

The intersection points of these lines form the vertices of the feasible region. From the provided graph, we can identify these points:

$(0, 1)$: Intersection of $x = 0$ and $y = 1$

$(1, 2)$: Intersection of $y = x + 1$ and $y = 2$

$(2, 2)$: Intersection of $y = 3 - 0.5x$ and $y = 2$

$(3, 1.5)$: Intersection of $y = 6 - 1.5x$ and $y = 3 - 0.5x$

$(4, 0)$: Intersection of $y = 4$ and $y = 0$

$(0, 0)$: Intersection of $y = 0$ and $y = 0$

Identifying these vertices is crucial because, according to linear programming principles, the maximum or minimum value of the objective function for a bounded feasible region occurs at one of the vertices.

3. Evaluate the Objective Function at the Vertices

The objective function is:

$$f(x, y) = 12(x2 + y2) - 2x + 2yf$$

By evaluating the objective function at each vertex, we determine the value of the function at all critical points. This step helps in identifying which vertex yields the maximum value for the objective function.

4. Determine the Optimal Solution

Compare the objective function values at each vertex to find the maximum:

$$f(0, 1) = 2.5$$

$$f(1, 2) = 4.5$$

$$f(2, 2) = 4$$

$$f(3, 1.5) = 2.625$$

$$f(4, 0) = 0$$

$$f(0, 0) = 0$$

4.5 > all other values.

Optimal Solutions

The optimal solution is at $f(1, 2)$ with an objective function value of 4.5. As shown by the graphical solution, *figure 2*.

This process shows how the feasible region is determined by the intersection of the constraint lines, and the objective function is evaluated at each vertex to find the maximum value. This can be visually confirmed using the provided graph, where the feasible region and vertices are plotted.

From this we can see that whilst quadratic programmes are intricate in the sense that they can involve more complex objective functions which can determine higher accuracy in results, they are more complicated to solve whether mathematically or computationally and this is what we'll explore and examine the trade-offs of this difference in an application of both methods.

Application of Models: Portfolio Optimisation for the S&P 500

This is where we begin to analyse the differences in accuracy and performance for the linear and quadratic programmes in application for portfolio analysis specifically. For our data, we chose to fetch data from the S&P 500 using the *yfinance* Python library from 2000 – 2015, 3772 trading days. We also cleaned the data and dropped stocks that not only had missing data but many 0 values at the beginning of the timeframe as we faced issues with the linear programme before we processed the data like this. After the data cleansing there are 353 stocks left in the portfolio.

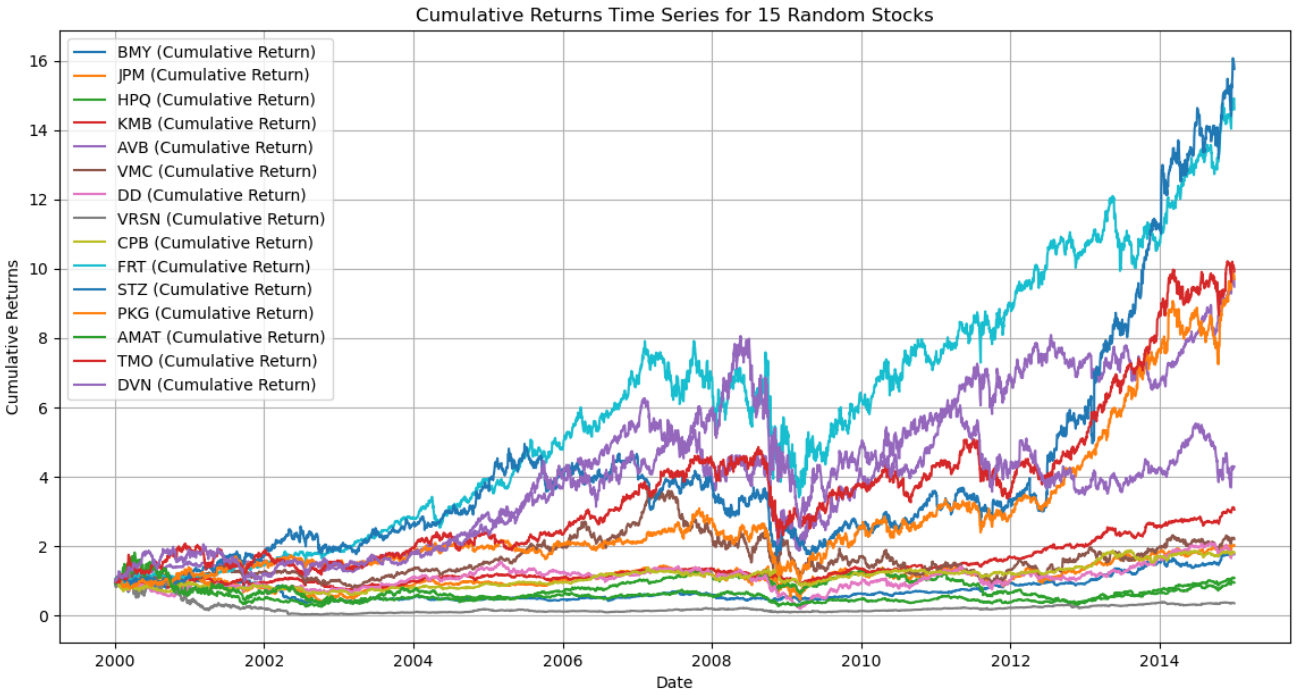


Figure 3: Time Series for S&P 500 Cumulative Returns (15 Stocks, 3772 Trading Days).

The objective of this portfolio of stocks is to optimise the portfolio so that the new portfolio can minimise the risk and maximise returns for both methods, linear and quadratic. Both optimising their own different portfolios. Given the metrics I wanted to analyse I had to determine how to make this comparison fair, meaning there are controls, for both models involved otherwise results can be seen as futile. Therefore, I chose to implement a linearisation of the quadratic equation for covariance within the linear programme's objective function, this was done by taking the mean of absolute deviations for each stock *i*.

Applied Linear Programming Model

To apply the set of 353 stocks' returns data to a linear programme, an algorithm using libraries in Python such as *cvxpy* for the linear programme itself into matrices suitable for multiplication.

Here is the provided linear problem we are solving:

Let:

- i represent each stock in the portfolio.
- j represents every interval in the portfolio, the interval is daily.
- Weights = W.
- Returns = R.
- Mean Returns = μ.
- α represents a balancing factor ranging from (0, 1), this determines whether the model will prefer to maximise returns or minimise variance. The higher α is, the higher the preference for minimising variance and vice versa.
- t represent the number of intervals for every stock, 3772 trading days.

Decision variable:

$$W_i$$

Objective Function:

$$\text{Minimise } \alpha * (W_i * \frac{\sum_j |R_{ij} - \mu_i|}{t}) - (1 - \alpha) * (W_i * \mu_i)$$

From now on let, $\frac{\sum_j |R_{ij} - \mu_i|}{t}$ *be denoted as* μD_i*, for mean absolute deviations for every stock.*

$$\text{Minimise } \alpha * (W_i * \mu D_i) - (1 - \alpha) * (W_i * \mu_i)$$

W_i, μD_i, μ_i are vectors and suitable for matrix multiplication. Hence why we took the *mean* absolute deviations for every stock over the span of t itervals.

Subject to:

$$\Sigma(W_i) = 1$$

$$W_i \geq 0$$

This problem tells us that we are minimising variance, as calculated by the mean absolute deviations, and maximising mean returns given our value of, α, our preference.

The weights of the portfolio can be represented as a vector with dimensions 353 x 1:

$$W_i = \begin{pmatrix} W_1 \\ \vdots \\ W_{353} \end{pmatrix}$$

The mean absolute deviations can be represented as a vector with dimensions 353 x 1:

$$\mu D_i = \begin{pmatrix} \mu D_1 \\ \vdots \\ \mu D_{353} \end{pmatrix}$$

These have been setup where a matrix multiplication is now feasible for the optimisation in Python. However, before this step, the algorithm solving the optimising the problem must occur to distribute the decision variable (W_i) values.

This is important as this will now determine from the objective function what the portfolio of stocks will be based on values of summed absolute deviations each stock has.

Portfolio Optimisation

The linear nature of the model inherently limits its ability to fully account for the complexities of stock correlations and market conditions compared to more sophisticated quadratic programming models. Hence, we can see the linear programme invest all its assets into 1 stock, '*ED'*, with no diversification at $\alpha = 0.5$, *see appendix [1].*

Through testing, we determined this to be caused by the imbalanced scaling of variables within the objective function:

$$\text{Minimise } \alpha * (W_i * \mu D_i) - (1 - \alpha) * (W_i * \mu_i)$$

Where $(W_i * \mu D_i)$ is substantially larger than $(W_i * \mu_i)$with μD_i being larger than μ. This is because of the linear programme not being able to identify, with any given value of α, what preference we have for our portfolio. And this causes the linear portfolio to be heavily bias to one stock unless we alter α to absurd values for diversification to begin. This shows a limitation with the linear programme's capability to linearise a complex relationship between stocks like covariance and utilise it in the same manner as the quadratic programme.

Furthermore, even without considering the scaling limitations, we can still infer that the linear programme will always be bias regardless of the values of alpha we provide since it is in the nature of the linear programme to invest into 1 stock as the 'optimal' solution.

Computational Efficiency and Optimisation

Despite this, one of the standout features of this linear programming model is its computational efficiency. The optimisation process was completed on average in 0.13 seconds, underscoring the model's capacity to handle extensive datasets swiftly. This efficiency is crucial in real-world financial applications, where timely decision-making can significantly impact investment outcomes. However, this is important to consider as a trade-off, especially when we analyse the Quadratic model, because the trade-off here is accuracy and time taken (depending on the problem at hand).

Applied Quadratic Programming Model

To apply the set of 353 stocks' returns data to a quadratic programme, an algorithm using libraries in Python such as *cvxpy* for the linear programme itself into matrices suitable for operation.

Here is the provided linear problem we are solving:

The covariance matrix Σ represents the covariance between pairs of assets in the portfolio. It is a square matrix where the element at the ith row and jth column, σ_{ik}, represents the covariance between the returns of the ith and kth assets.

Formula for covariance:

$$\sigma_{ik} = \frac{1}{N-1}\sum_{j=1}^{N}(R_{ij} - \mu_i)(R_{kj} - \mu_j)$$

Let:

- R_{ij} *is the return of asset* i *on day* j*,*
- μ_i *is the mean return of asset* i*,*
- *N is the total number of observations.*

Decision variable:

$$W_i$$

The entire covariance matrix Σ is: 353 x 353

$$\Sigma = \begin{bmatrix} \sigma_{11} & \cdots & \sigma_{1k} \\ \vdots & \ddots & \vdots \\ \sigma_{i1} & \cdots & \sigma_{ik} \end{bmatrix}$$

Objective Function:

$$\text{Minimise } \alpha * W_i^T \Sigma W_i - (1 - \alpha) * (W_i * \mu_i)$$

Here, i represents all stocks, W_i is the vector of portfolio weights, and W_i^T is the transpose of W. The expression $W^T \Sigma W$ computes the portfolio variance based on the asset weights and the covariances between them.

Subject to:

$$\Sigma(W_i) = 1$$

$$W_i \geq 0$$

The quadratic programming model optimises the allocation of weights across the assets to minimise the overall portfolio risk and maximise returns given α, while adhering to the specified constraints. The covariance matrix Σ plays a crucial role in this optimisation, as it quantifies the relationships between the asset returns, allowing the model to consider not just individual asset risks but also how assets interact with each other, collinearity.

This is something that can particularly be tested for with Econometric methods to analyse and test for multicollinearity between variables, however this is something we will not test for and will only consider its presence as significant based on results between the linear and quadratic portfolios.

Portfolio Optimisation

The quadratic nature of the model inherently had the ability to fully account for the complexities of stock correlations and market conditions compared to more to the previous linear model. Hence, we can see the quadratic programme invest diversely into these stocks, *'DECK', 'DVA', 'GILD', 'KMX', 'ODFL', 'REGN', 'TSCO', 'TYL', 'WDC'. See appendix [2].*

This is expected of a quadratic programme to do especially when considering what was discussed regarding the capturing of collinearity between stocks. However, this still does not tell us why this model worked in balancing the two objectives, minimising variance and maximising returns, better than the linear programme.

Through testing, we determined this to be caused by the balanced scaling of variables within the objective function:

$$\text{Minimise } \alpha * W_i^T \Sigma W_i - (1 - \alpha) * (W_i * \mu_i)$$

Where $W_i^T \Sigma W_i$ is relatively equivalent to $(W_i * \mu_i)$ thus having comparable scales. I found that this is due to the nature of variance in the quadratic equation not having to sum absolute values, which in turn did not inflate the value of the sum of variance like the linear programme did. Keeping the value in scale with the sum of mean returns.

This is means that for any value of α we used, the programme was able to identify our preferences between minimising variance and maximising returns and optimised accordingly. For example, when I provided $\alpha = 1$, this meant that my preference was 100% bias to minimising variance and as a result the programme diversified into majority of the 353 stocks. The opposite happened with $\alpha = 0$, the programme invested into 1 stock with the highest mean returns out of all the stocks to maximise returns.

Computational Efficiency and Optimisation

The optimisation process was completed on average in 0.4 seconds, which is almost 3x slower than the linear programme. Though we must keep in mind that there again is the importance of the trade-off, accuracy and time taken. In this case, the optimisation process took 3x longer than the linear programme, though it was quick, this can be exponentially longer with larger datasets and more complex processes. With this problem we solved, it seems that the Quadratic programme seems to fit much better given its nature to have balanced scaling between the variables we are optimising for. Had this been another problem, a simpler one, the linear programme can be seen as a more valuable method especially regarding computational efficiency.

Long Run Analysis of Optimised Portfolios and Real-World

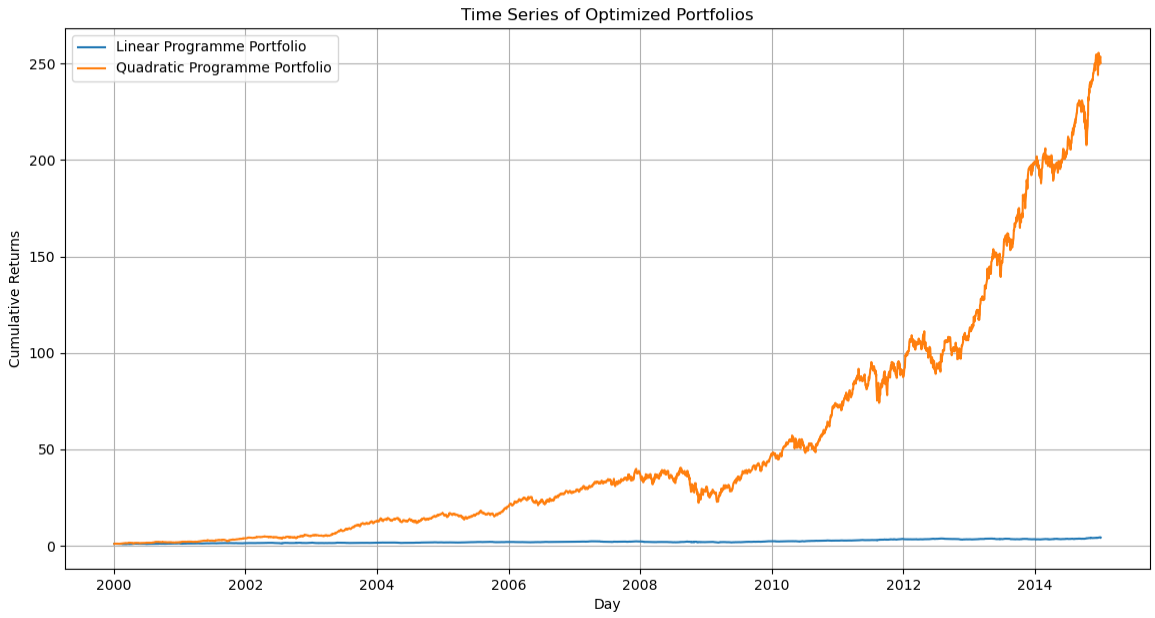


Figure 4: Time Series of Portfolio Cumulative Returns for Linear and Quadratic Portfolios, $\alpha = 0.5$.

See appendix [3] for individual time series of linear programmes cumulative returns.

Performance Analysis

The comparative analysis of the linear and quadratic programming portfolios demonstrates distinct differences in performance, attributable to the intrinsic characteristics of the two models. The time series of cumulative returns reveals that the quadratic portfolio outperformed the linear, with mean returns of approximately 0.0016370732 compared to 0.0004511053 Exhibiting a 263% increase in returns. This superior performance is due to the quadratic model's tendency to adhere to values of α and balance the objective function to maximise returns and minimise variance, which inherently allows the portfolio to pick the best of both worlds. And we can attribute this proper allocation of α to the fact that

Risk and Variance Considerations

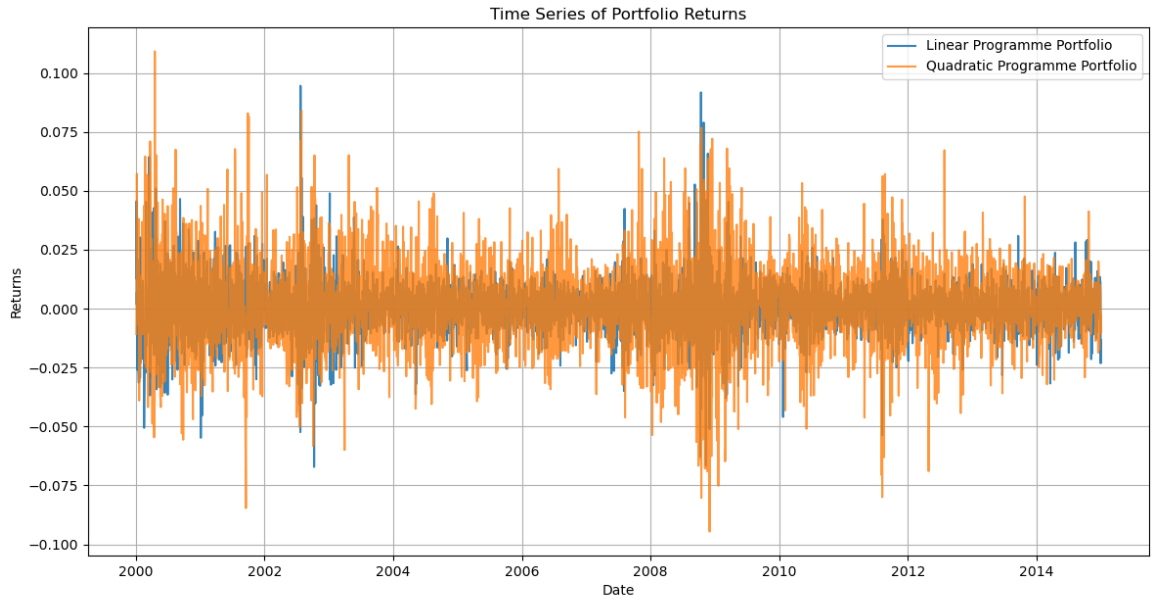


Figure 5: Time Series of Long Run Portfolio Returns for Linear and Quadratic Portfolios, $\alpha = 0.5$.

Furthermore, the variance of the linear portfolio conveys less risk at a variance of 0.0001370330 compared to 0.0003444271, exhibiting an approximate 60% decrease in volatility compared to the quadratic model. This is quite interesting when because the linear programme is not diversified whatsoever, and the quadratic programme is. However, we know these results occur due to the fact that the linear programme is highly bias to stocks with low volatility due to the issues discussed about the objective function. Interestingly, we find obscure results when setting $\alpha = 0$ for both programmes.

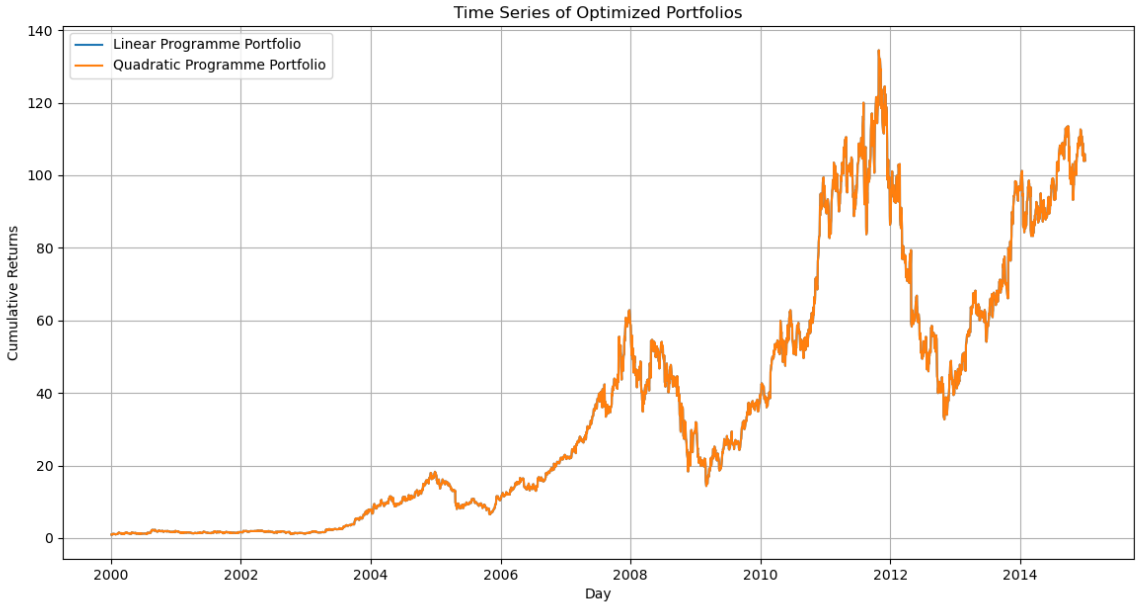


Figure 6: Time Series of Long Run Portfolio Returns for Linear and Quadratic Portfolios, $\alpha = 0$.

Here, we see that both portfolios invest into the same stock, '*DECK'*, this is because this stock has the highest returns of any stock in the portfolio of 353 stocks in the S&P and when we set $\alpha = 0$, it will make sense for both programme to prioritise maximising returns fully as $1 - 0 = 1$ (100%), thus a 100% bias in both objective functions. However, it is interesting to see the linear programme behave in this according manner even though it has scaling issues with the preferences in the objective function especially since for any other value α, between 0 – 1, the linear programme does not behave accordingly. As shown in the previous time series, *see figure [4]*.

Short Run (1-Year) Analysis of Optimised Portfolios and Real-World

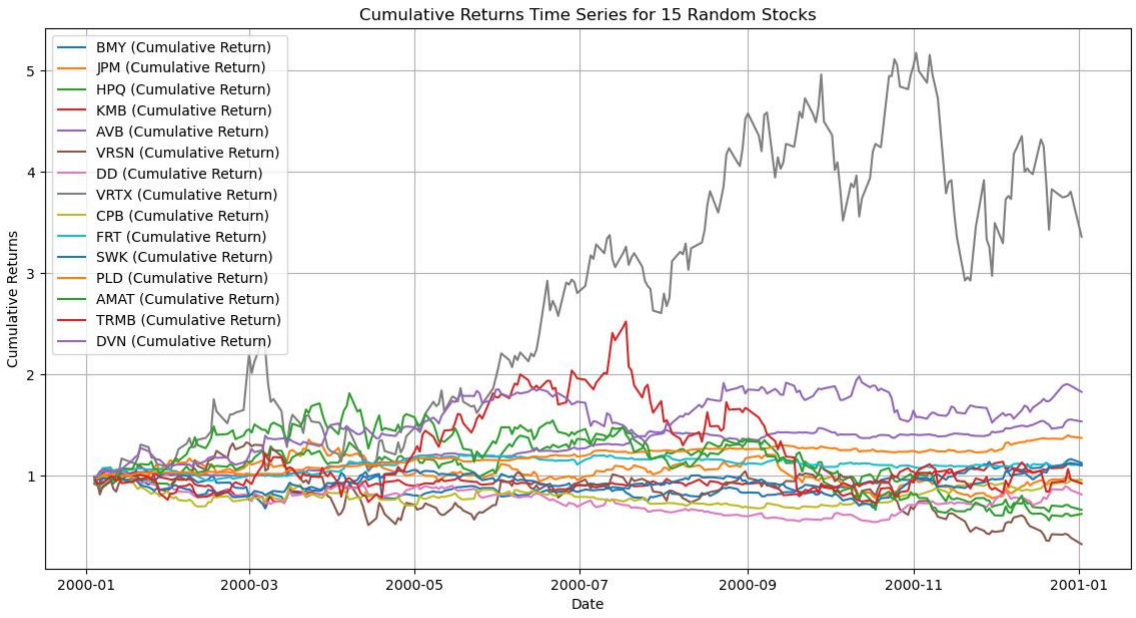


Figure 7: Time Series for S&P 500 Cumulative Returns (15 Stocks, 252 Trading Days).

These results were based on the entire timeframe that the stocks' data were collected from, 2000 – 2015, which is a long time. Thus, we also tested for performance in the short run to determine the capability of both models in shorter intervals where metrics like risk have more influence and more important to mitigate given the fact that in the long run, risk is almost always mitigated.

We ran the programmes at $\alpha = 0.5$

Once again at, when running the linear optimisation model, there was a high bias to one stock and the optimised portfolio is again only a singular stock with no diversification for any value of α, *see appendix [3]*. The cements the issue with the linear programme being the linearisation of a complex equation such as covariance into absolute deviations as this does not allow any scalability for a balance of preferences in the objective function.

The quadratic programme invested into a diverse portfolio of stocks, *see appendix [4]*.

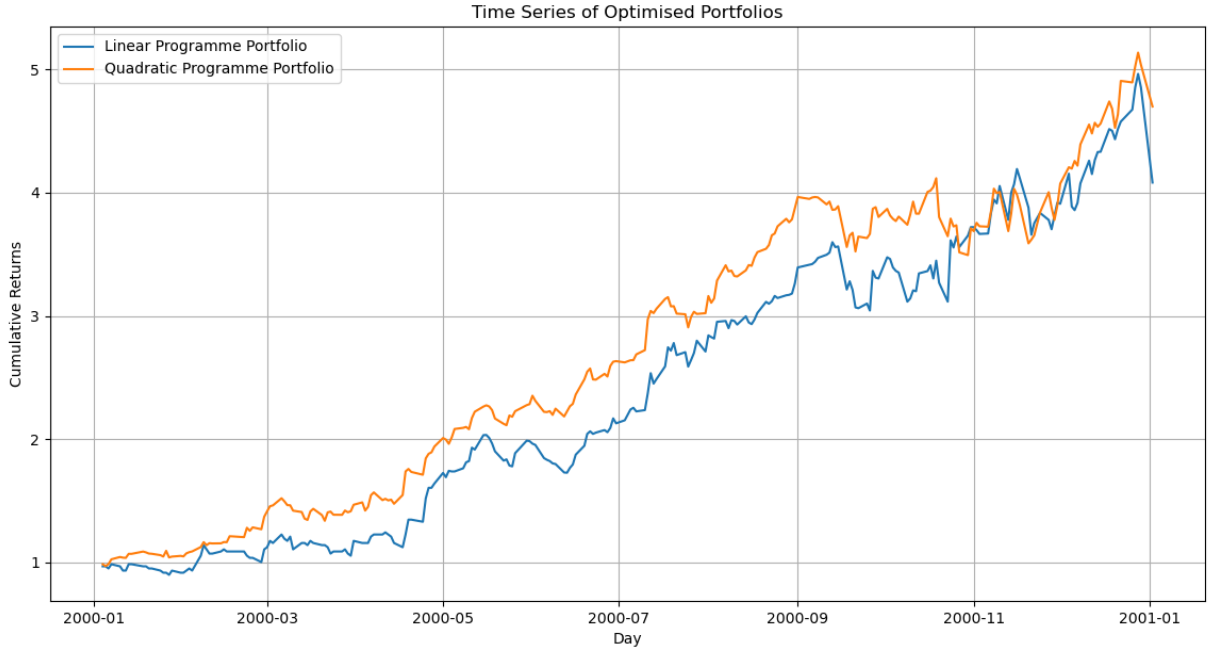


Figure 8: Time Series of Short Run Portfolio Cumulative Returns for Linear and Quadratic Portfolios.

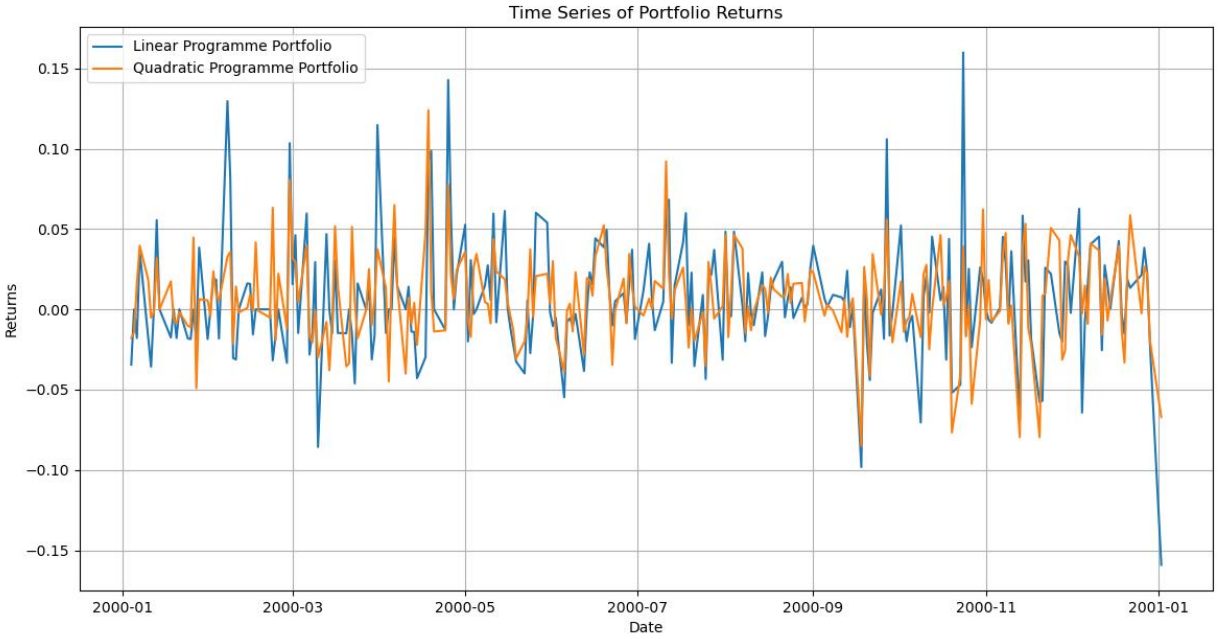


Figure 9: Time Series of Portfolio Returns for Linear and Quadratic Portfolios.

Here we can see that once again, though significantly closer in results, the linear programme is beat by the quadratic with higher mean returns and lower variance. However, after determining that this discrepancy in results is mainly since the linear programme cannot operate properly in the first place. This can be seen as an unfair comparison to make. Despite this, it's important to note the differences in performance these programmes can have as we can determine trade-offs between them if we were to iterate on the linear programme for a different problem where the objective function could potentially work similarly to the quadratic. It also seems that the best performing stock in the portfolio in this time frame is the one that both portfolios invested mostly into, 'LH'. It is quite interesting to see how similar the results can be, even though the linear programme has its own discrepancies, in the short run for any value of α.

Conclusion

The exploration of convex optimisation [1], particularly through linear and quadratic programming, has profound implications in the field of finance and beyond. These mathematical tools offer structured methodologies for decision-making in complex environments, where balancing multiple objectives, such as risk and return, is essential. This study's comparative analysis of linear and quadratic programming models in portfolio optimisation illuminates key aspects of convex optimisation's practical applications and limitations.

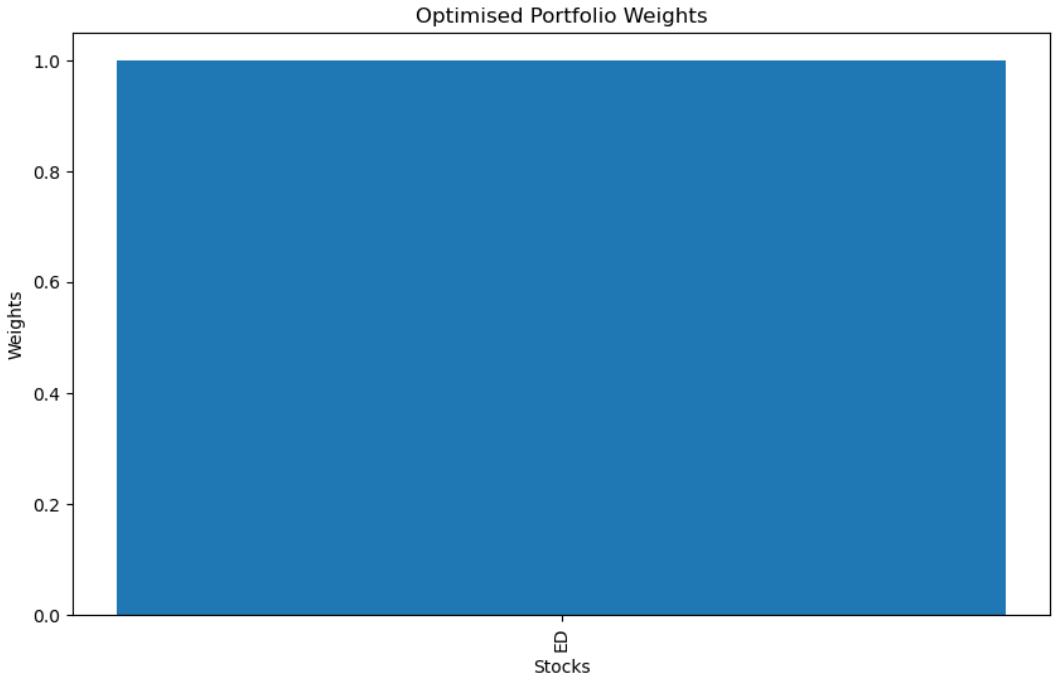
The broader implications of this study highlight the critical importance of understanding the underlying mathematical frameworks when applying convex optimisation techniques in practical scenarios. The choice between linear and quadratic programming should be informed by the specific context of the application, including the desired balance between computational efficiency and risk management. In finance, this balance is particularly crucial, as it directly impacts investment strategies and outcomes. As well as the objective itself and the timeframe of which the data exists, as we saw from results methods may perform similarly in the short run but highly differ in the long run.

Furthermore, the study reinforces the value of incorporating constraints and practical considerations into optimisation models to align them more closely with real-world scenarios. This can involve the balancing of preferences to promote have desired outcomes and constraints used to avoid certain thresholds or characteristics in a model, as shown by the example models.

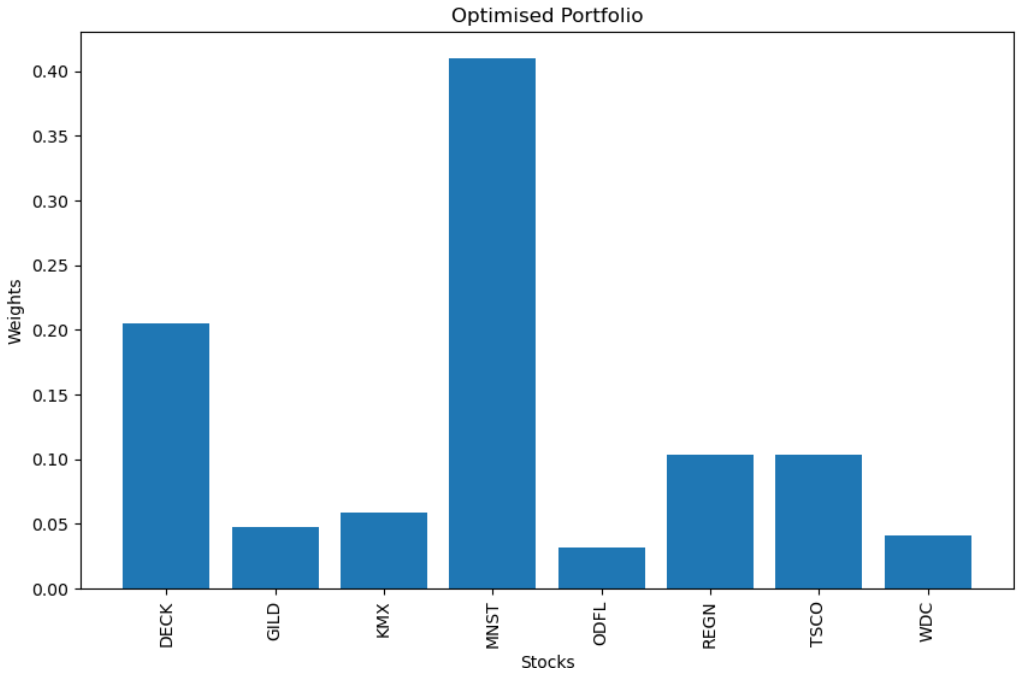
In summary, convex optimisation, exemplified through linear and quadratic programming, provides powerful tools for tackling complex optimisation problems across various domains. This study's findings underscore the importance of a deep understanding of these methods, as well as the necessity of tailoring optimisation strategies to specific objectives and constraints, such as in portfolio optimisation. As technology and computational methods advance, the applications of convex optimisation will continue to expand, offering new opportunities and challenges in fields ranging from finance to logistics and beyond.

Appendices

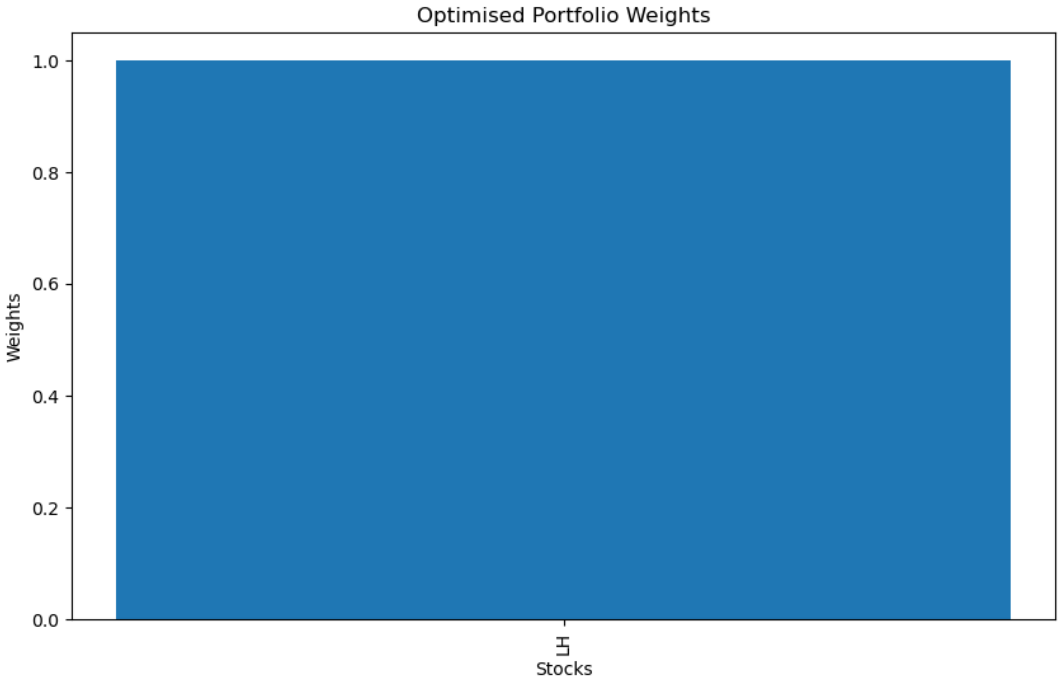
Appendix 1: Linear Programme, Long Run Optimised Portfolio.



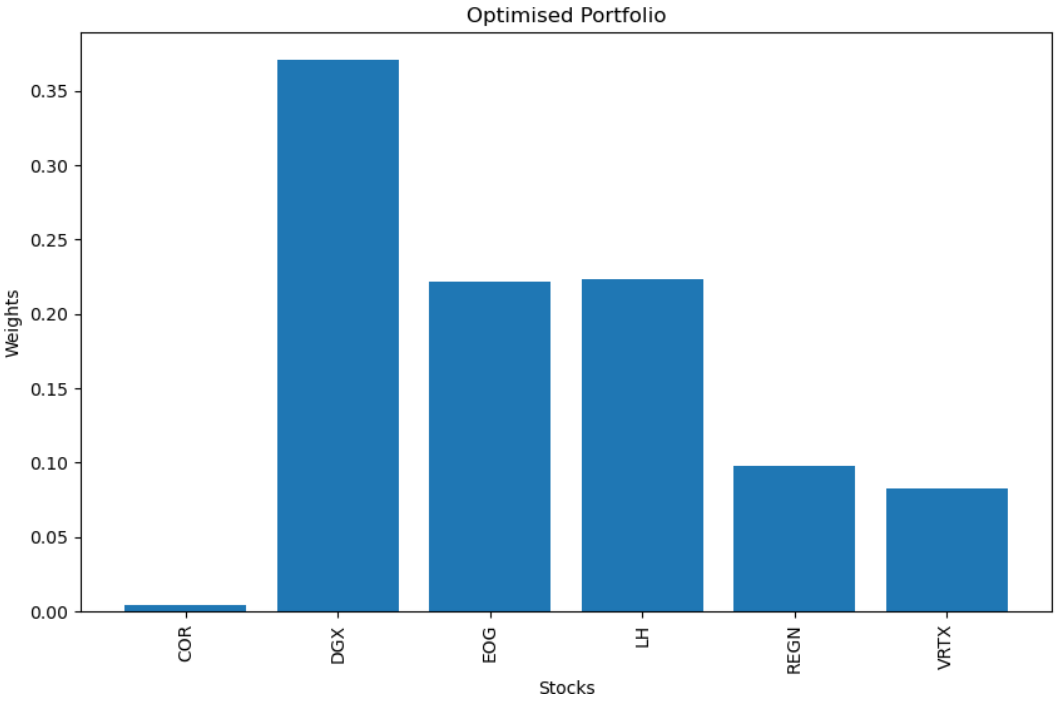
Appendix 2: Quadratic Programme, Long Run Optimised Portfolio.



Appendix 3: Linear Programme, Short Run Optimised Portfolio.



Appendix 4: Quadratic Programme, Short Run Optimised Portfolio.



References

[1] S. Boyd and L. Vandenberghe, Convex Optimization, Cambridge University Press (2004).

[2] F. Cesarone, A. Scozzari, and F. Tardella, "Portfolio selection problems in practice: comparison between linear and quadratic optimization models," Università degli Studi Roma Tre - Dipartimento di Economia (2010).