# ICBS Computational Finance with C++

# Coursework

# Name: Ronald Leung

# CID: 02422598

## Software Structure

A diagram of functions and functions

Description automatically generated

Figure 1 Class diagram

There are 5 classes and 3 functions in this program, as illustrated in figure 1.

|  |  |
| --- | --- |
| **Class** | ParameterEstimation |
|  | * Calculate mean return and covariance matrix |
| **Class** | TargetReturns |
|  | * Create a vector of target return (from 0% to 10%) |
| **Class** | MatrixCaculator |
|  | * Implement basic matrix operations |
| **Sub Class** | ConjugateGradientMethod, a sub class of MatrixCalculator |
|  | * Conjugate Gradient Method |
| **Class** | Quadratic |
|  | * Prepare Q matrix, b matrix, initial guess x0 vector * Select a specific range of the return matrix * Calculate OOS returns |

|  |  |
| --- | --- |
| **Function** | readData |
|  | * Read CSV files to get stock returns, original given function |
| **Function** | String\_to\_double |
|  | * Convert string to doble |
| **Function** | MatrixWriter |
|  | * Write results to CSV files |

A screenshot of a computer program

Description automatically generated

Figure 2 main.cpp structure

Figure 2 illustrates the structure of main.cpp. The main.cpp can be divided into 4 sessions with a structure similar to the coursework instructions.

The main idea is to make C++ code in the main.cpp file feels like Python. With the classes and functions that are built previously in figure1, the modular code in main.cpp could be clean and easy to debug.

## Evaluation

A graph with many colored dots

Description automatically generated

Figure 3 Actual Return vs Variance for each iteration

Figure 3 plots the actual return across different level of variance. Each point represents a portfolio at a specific iteration with a given target return.

A diagram of a graph

Description automatically generated with medium confidence

Figure 4 Efficient Frontier vs OOS Actual Return, Expected Volatility (standard deviation) as x-axis

Figure 4 compares the in-sample efficient frontier with the out-of-sample performance across multiple iterations.

The blue curve represents the efficient frontier calculated using all in-sample data. This curve shows the optimal portfolios that provide the maximum expected return for a given level of risk (expected volatility) based on historical data.

Coloured points represent the out-of-sample performance of portfolios across different iterations. The points are scattered around the in-sample efficient frontier (blue curve), indicating variability in performance when the model is tested using out-of-sample data. The variability can be attributed to the limitation of the Markowitz model, changing market conditions, and model overfitting to the sub-set of data.

Even though the out-of-sample points are scattered around the efficient frontier, it is still able to observe the risk-return trade-off that is inherent in portfolio optimisation. The efficient frontier represents the theoretical best trade-off, while the out-of-sample points show the real-world deviations.

The scatter of out-of-sample points emphasizes the importance of robust back-testing. It shows that relying solely on in-sample data for portfolio optimization can be misleading. Continuous back-testing with rolling windows helps in assessing the stability and reliability of the optimization model, providing a more realistic expectation of future performance.A graph of a diagram

Description automatically generated with medium confidenceA graph of a number of blue dots

Description automatically generated with medium confidenceA chart with blue and green dots

Description automatically generatedA graph of blue dots

Description automatically generated

Figure 5 Out of sample actual returns vs target returns, compare with all iterations

The **3D scatter plot** shows the relationship between the iterations, target returns, and the difference between the target return and the actual return. Each point represents a portfolio at a specific iteration with a given target return, coloured by the magnitude of the difference.

From the **top view**, it indicates that the effectiveness of the portfolio optimization process varies over time (across iterations). It is suggested that market conditions could change from time to time, and it is important to periodically reassess and rebalance the portfolio to adapt to changing environments.

From the **side view**, it suggests that portfolios with higher target returns tend to underperform more compared to their targets. It highlights that aiming for higher returns using the Markowitz model might lead to larger deviations from the target. This could be due to an increased risk-return trade-off, a higher level of risk associated with a high target return can lead to greater deviations from the target.

The **Front view and Top view** show trends and cycles in the portfolio’s performance. Negative differences (target return – actual return) occur periodically with temporal patterns. There could be some time-varying factors that are impacting the portfolios’ performance which require further investigations.

## Appendix

C++ source code and python plots are zipped for your further reference.