Portfolio Optimization: Development and Comparative Analysis of Frank-Wolfe Algorithms on Real-World Data

Thomas Rosso id: **Francesco Gabriele id:**

Abstract

This paper presents a comprehensive study on the application of Frank-Wolfe algorithms and their variants, including the Away-Step Frank-Wolfe and Projected Gradient methods, for solving portfolio optimization problems. The research begins with an in-depth analysis of the theoretical foundations of these algorithms, followed by the development and implementation of efficient code to solve the portfolio optimization problem. The algorithms are tested on real-world portfolio datasets, and their performance is evaluated using various metrics.

1. Introduction to Portfolio Optimization

The Markowitz Portfolio Problem, introduced by Harry Markowitz in 1952, revolutionized the field of finance by providing a mathematical framework for portfolio selection. The primary objective of this problem is to find the optimal portfolio that minimizes risk for a given level of expected return or maximizes return for a given level of risk. This is achieved by balancing the trade-off between risk, represented by the variance of portfolio returns, and return, represented by the expected return of the portfolio.

1.1. Problem Formulation

Consider a portfolio consisting of n assets. Let x_i denote the proportion of the total budget invested in the i-th asset, and let r_i represent the expected return of the i-th asset. Assuming the returns $r \in \mathbb{R}^n$ of the assets follow a normal distribution with a mean vector $\bar{r}$ and a covariance matrix Σ, the Markowitz optimization problem can be formulated as:

$$\min_{x \in \mathbb{R}^n} \left(\eta \cdot x^\top \Sigma x - \bar{r}^\top x \right) \tag{1}$$

subject to:

$$\sum_{i=1}^{n} x_i = 1, \tag{2}$$

$$x_i \geq 0 \quad \forall i \in \{1, \ldots, n\}, \tag{3}$$

where $\eta > 0$ is the risk-aversion parameter, controlling the trade-off between minimizing risk and maximizing return.

The parameter η reflects the investor's risk preference:

- $0 < \eta < 1$: Characterizes a risk-seeking investor willing to take on higher risks for potentially greater returns.
- $\eta = 1$: Represents an investor following the classical mean-variance approach, aiming to optimize the balance between expected returns and risk.
- $\eta > 1$: Indicates a risk-averse investor, preferring to reduce risk even if it means accepting lower returns.

The goal is to determine the portfolio weights x that minimize the portfolio's risk, represented by $x^\top \Sigma x$, while achieving a balance with the expected return, represented by $\bar{r}^\top x$, as determined by the investor's risk aversion Rinaldi (2023).

1.2. Gradient of the Objective Function

The objective function in the Markowitz optimization problem is given by:

$$f(x) = \eta \cdot x^\top \Sigma x - \bar{r}^\top x$$

where x is the vector of portfolio weights, Σ is the covariance matrix of asset returns, $\bar{r}$ is the vector of expected returns, and η is the risk-aversion parameter.

The gradient of $f(x)$ with respect to the portfolio weights x is calculated as follows:

$$\nabla f(x) = \frac{\partial}{\partial x}\left(\eta \cdot x^\top \Sigma x - \bar{r}^\top x\right)$$

Breaking it down:

- The gradient of the quadratic term $\eta \cdot x^\top \Sigma x$ is:

$$\nabla_x \left(\eta \cdot x^\top \Sigma x\right) = 2\eta \Sigma x$$

- The gradient of the linear term $-\bar{r}^\top x$ is:

$$\nabla_x \left(-\bar{r}^\top x\right) = -\bar{r}$$

Thus, the gradient of the objective function is:

$$\nabla f(x) = 2\eta \Sigma x - \bar{r}$$

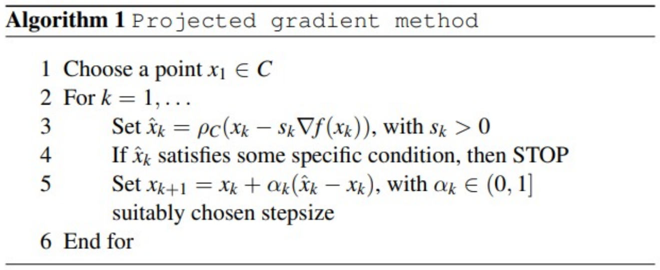
This gradient provides the direction in which the portfolio weights x should be adjusted to minimize the objective function. In optimization algorithms, the gradient guides the steps taken toward finding the optimal portfolio allocation.

2. Projected Gradient Algorithm

The Projected Gradient Algorithm is a widely-used optimization method, particularly suitable for problems involving constraints, such as the Markowitz portfolio optimization problem. The key idea behind this algorithm is to combine the gradient descent method with a projection step that ensures the iterates remain within the feasible set defined by the constraints. In the context of portfolio optimization, this ensures that the portfolio weights remain non-negative and sum to one, adhering to the budget constraint.

2.1. Algorithm Description

Given an objective function $f(x)$ that we wish to minimize over a feasible set C, the Projected Gradient Algorithm iteratively updates the current solution by moving in the direction opposite to the gradient of the function, followed by a projection back onto the feasible set.



Algorithm 1 Projected gradient method

```
1 Choose a point $x_1 \in C$
2 For $k = 1, \dots$
3     Set $\hat{x}_k = \rho_C(x_k - s_k \nabla f(x_k))$, with $s_k > 0$
4     If $\hat{x}_k$ satisfies some specific condition, then STOP
5     Set $x_{k+1} = x_k + \alpha_k(\hat{x}_k - x_k)$, with $\alpha_k \in (0, 1]$
      suitably chosen stepsize
6 End for
```

2.1.1. Algorithm Steps

1. **Initialization**: Start from an initial feasible point $x^{(0)} \in C$, typically chosen as an equally weighted portfolio (i.e., $x^{(0)} = \frac{1}{n}\mathbf{1}$, where n is the number of assets).

2. **Iterative Update**:

   (a) **Gradient Calculation**: Compute the gradient of the objective function at the current point $x^{(k)}$:

   $$\nabla f(x^{(k)}) = 2\eta \Sigma x^{(k)} - \bar{r}$$

   (b) **Projection Step**: Project the updated point onto the feasible set C. Depending on the selected projection method, this can be done using:

   - A standard simplex projection.
   - An optimized projection method, such as the one proposed by Condat (2016)Condat (2016), which efficiently projects onto the simplex or ℓ_1 ball.

   The execution time of each projection method was recorded to assess their relative performance.

Mathematically, this can be written as:

$$x^{(k+1)} := P_C\left(x^{(k)} - \alpha_k \nabla f(x^{(k)})\right)$$

where P_C is the projection operator.

(c) **Convergence Check**: Calculate the gradient mapping value $g^{(k)} = \|x^{(k)} - P_C\left(x^{(k)} - s \cdot \nabla f(x^{(k)})\right)\|$. If $g^{(k)} < \epsilon$, the algorithm is considered to have converged.

(d) **Line Search (Optional)**: Update the step size γ using a line search method to ensure the objective function is minimized efficiently.

3. **Stopping Criterion**: The algorithm continues until the gradient mapping norm is below a certain threshold ϵ or the maximum number of iterations is reached. In this analysis the chosen value for the threshold and the maximum number of iterations were respectively:

- ϵ = 1e-5
- Maximum number of iterations = $\frac{1}{\epsilon}$

2.2. Projection onto the Simplex

In the context of portfolio optimization, the feasible set C is often the unit simplex, defined as:

$$C = \left\{ x \in \mathbb{R}^n \mid \sum_{i=1}^{n} x_i = 1,\ x_i \geq 0 \, \forall i \right\}$$

The projection of a point y onto the simplex can be computed efficiently using various algorithms. One such method, as discussed by Condat (2016)Condat (2016), is the fast projection onto the simplex. This method involves sorting the elements of y and then determining the appropriate threshold τ such that the projected point $x = P_C(y)$ satisfies the constraints of the simplex.

The projection step can be formally written as:

$$x = P_C(y) = \arg\min_{x \in C} \|x - y\|_2^2$$

where $\|\cdot\|_2$ denotes the Euclidean norm.

2.3. Convergence Properties

The Projected Gradient Algorithm has well-established convergence properties, particularly when applied to convex optimization problems like portfolio optimization. For a convex objective function $f(x)$, the algorithm is guaranteed to converge to a stationary point where the gradient is orthogonal to the feasible set's boundary. This ensures that no further significant decrease in the objective function value is possible within the feasible region.

The convergence rate of the algorithm can be linear, especially when the objective function is strongly convex. This means that the distance to the optimal solution decreases exponentially with each iteration. The choice of step size s is crucial in determining the efficiency of convergence, with adaptive step sizes often providing faster convergence by aligning with the local curvature of the objective function. Convergence is assessed using a gradient mapping norm $g^{(k)} = \|x^{(k)} - P_C\left(x^{(k)} - s \cdot \nabla f(x^{(k)})\right)\|$. The algorithm stops when this norm falls below a pre-specified threshold ϵ, indicating that the solution has stabilized. A well-chosen ϵ balances computational efficiency with the desired accuracy of the solution.

3. Frank-Wolfe Algorithm

The Frank-Wolfe algorithm, also known as the Conditional Gradient method, is a classic iterative optimization algorithm particularly suited for solving constrained convex optimization problems. First introduced by Frank and Wolfe in 1956, this algorithm has gained renewed interest in recent years due to its projection-free nature, making it highly efficient for large-scale optimization problems where projection operations can be computationally expensive. Unlike other gradient-based methods that require a projection step to maintain feasibility, the Frank-Wolfe algorithm updates the solution by solving a linear optimization subproblem at each iteration, thus avoiding the need for expensive projections.

3.1. Algorithm Description

The Frank-Wolfe algorithm aims to minimize a convex objective function $f(x)$ over a compact convex set $\mathcal{D}$. The algorithm can be summarized in the following steps:

Algorithm 1 Frank-Wolfe (1956)

```
Let $\boldsymbol{x}^{(0)} \in \mathcal{D}$
for $k = 0 \dots K$ do
    Compute $\boldsymbol{s} := \arg\min_{\boldsymbol{s} \in \mathcal{D}} \langle \boldsymbol{s}, \nabla f(\boldsymbol{x}^{(k)}) \rangle$
    Update $\boldsymbol{x}^{(k+1)} := (1-\gamma)\boldsymbol{x}^{(k)} + \gamma \boldsymbol{s}$,   for $\gamma := \frac{2}{k+2}$
end for
```

In each iteration, the algorithm moves towards the linearized minimum of the objective function within the feasible region $\mathcal{D}$. The step size $\gamma^{(k)}$ is chosen to ensure convergence while balancing the trade-off between exploration and exploitation of the search space.

4. Away-Steps Frank-Wolfe

The Away-Steps Frank-Wolfe (AFW) algorithm was developed to address the slow convergence and zig-zagging behavior of the classical Frank-Wolfe (FW) algorithm, especially when the optimal solution lies on the boundary of the feasible region. AFW improves on the original method by introducing "away steps," which allow the algorithm to move away from atoms that contribute negatively to the objective function.

In each iteration, AFW computes both the standard Frank-Wolfe direction and an away direction, enabling it to move away from the worst atom in the current active set. This added flexibility helps correct the iterates' trajectory and significantly enhances convergence. Notably, under strong convexity, AFW achieves global linear convergence, a substantial improvement over the original FW algorithm's sublinear rate Lacoste-Julien and Jaggi (2015).

This chapter provides an in-depth discussion of the AFW algorithm as applied to portfolio optimization, where away steps improve performance near the boundary of the feasible set.

4.1. Algorithm Description

The AFW algorithm iteratively refines the portfolio weights by alternating between the standard Frank-Wolfe steps and away steps. This hybrid approach allows the algorithm to dynamically choose the most effective direction at each iteration, thereby improving convergence, especially when the iterates are close to the optimal solution.

Algorithm 1 Away-steps Frank-Wolfe algorithm: $\mathbf{AFW}(\boldsymbol{x}^{(0)}, \mathcal{A}, \epsilon)$

```
1:  Let $\boldsymbol{x}^{(0)} \in \mathcal{A}$, and $\mathcal{S}^{(0)} := \{\boldsymbol{x}^{(0)}\}$      (so that $\alpha_{\boldsymbol{v}}^{(0)} = 1$ for $\boldsymbol{v} = \boldsymbol{x}^{(0)}$ and 0 otherwise)
2:  for $t = 0 \dots T$ do
3:      Let $\boldsymbol{s}_t := \mathrm{LMO}_{\mathcal{A}}(\nabla f(\boldsymbol{x}^{(t)}))$ and $\boldsymbol{d}_t^{\mathrm{FW}} := \boldsymbol{s}_t - \boldsymbol{x}^{(t)}$      (the FW direction)
4:      Let $\boldsymbol{v}_t \in \arg\max_{\boldsymbol{v} \in \mathcal{S}^{(t)}} \langle \nabla f(\boldsymbol{x}^{(t)}), \boldsymbol{v} \rangle$ and $\boldsymbol{d}_t^{\mathrm{A}} := \boldsymbol{x}^{(t)} - \boldsymbol{v}_t$      (the away direction)
5:      if $g_t^{\mathrm{FW}} := \langle -\nabla f(\boldsymbol{x}^{(t)}), \boldsymbol{d}_t^{\mathrm{FW}} \rangle \leq \epsilon$ then return $\boldsymbol{x}^{(t)}$      (FW gap is small enough, so return)
6:      if $\langle -\nabla f(\boldsymbol{x}^{(t)}), \boldsymbol{d}_t^{\mathrm{FW}} \rangle \geq \langle -\nabla f(\boldsymbol{x}^{(t)}), \boldsymbol{d}_t^{\mathrm{A}} \rangle$ then
7:          $\boldsymbol{d}_t := \boldsymbol{d}_t^{\mathrm{FW}}$, and $\gamma_{\max} := 1$      (choose the FW direction)
8:      else
9:          $\boldsymbol{d}_t := \boldsymbol{d}_t^{\mathrm{A}}$, and $\gamma_{\max} := \alpha_{\boldsymbol{v}_t}/(1 - \alpha_{\boldsymbol{v}_t})$      (choose away direction; maximum feasible step-size)
10:     end if
11:     Line-search: $\gamma_t \in \arg\min_{\gamma \in [0, \gamma_{\max}]} f(\boldsymbol{x}^{(t)} + \gamma \boldsymbol{d}_t)$
12:     Update $\boldsymbol{x}^{(t+1)} := \boldsymbol{x}^{(t)} + \gamma_t \boldsymbol{d}_t$      (and accordingly for the weights $\boldsymbol{\alpha}^{(t+1)}$, see text)
13:     Update $\mathcal{S}^{(t+1)} := \{\boldsymbol{v} \in \mathcal{A}$ s.t. $\alpha_{\boldsymbol{v}}^{(t+1)} > 0\}$
14: end for
```

4.1.1. Algorithm Steps

The following steps outline the process for the Away-Steps Frank-Wolfe (AFW) algorithm, starting from initialization and continuing through the iterative process until convergence.

1. **Initialization**: Initialize the portfolio weights x_0 equally across all assets, ensuring the initial point satisfies the simplex constraint (i.e., the weights sum to one). The active set is initialized with this starting point, and the corresponding vertex weight is set to 1.

$$x_0 = \frac{1}{n}\mathbf{1}$$

$$\text{Active set} = \{x_0\}$$

$$\text{Vertex weight} = \{1.0\}$$

2. **Gradient Calculation**: At each iteration k, compute the gradient of the objective function $\nabla f(x_k)$ at the current portfolio weights x_k:

$$\nabla f(x_k) = 2\eta \Sigma x_k - \bar{r}$$

3. **Frank-Wolfe Step**: Solve the Linear Minimization Oracle (LMO) to find the vertex s that minimizes the linear approximation of the objective function. Compute the Frank-Wolfe direction:

$$d_{FW} = s - x_k$$

4. **Away Step**: Identify the vertex v_A from the active set that maximizes the dot product with the gradient. Compute the away direction:

$$d_A = x_k - v_A$$

5. **Duality Gap Calculation**: Calculate the duality gap, which serves as a measure of how close the current solution is to optimality:

$$g(x_k) = \langle x_k - s, \nabla f(x_k) \rangle$$

6. **Direction Selection**: Compare the directional derivatives of the Frank-Wolfe and Away directions. Choose the direction that offers the steepest descent:

$$\text{If } \langle d_{FW}, \nabla f(x_k) \rangle \leq \langle d_A, \nabla f(x_k) \rangle$$

choose the Frank-Wolfe step. Otherwise, choose the Away step.

7. **Step Size Determination**: Determine the step size γ_k using either a line search or a diminishing step size rule. The step size is crucial for balancing convergence speed and accuracy:

$$\gamma_k = \arg \min_{\gamma \in [0, \gamma_{max}]} f(x_k + \gamma d)$$

8. **Update Portfolio Weights**: Update the portfolio weights using the chosen direction and step size:

$$x_{k+1} = x_k + \gamma_k d$$

9. **Active Set and Vertex Weights Update**: After updating the portfolio weights, adjust the active set and the corresponding vertex weights. If the direction was a Frank-Wolfe step, the vertex s is either added or its weight is increased. If it was an Away step, the weight of the vertex v_A is reduced, and it may be removed from the active set if its weight drops to zero.

The algorithm iterates until the duality gap $g(x_k)$ falls below a predefined threshold ϵ, indicating convergence to the optimal solution.

5. Other Variants of the Frank-Wolfe Algorithm

While the classical Frank-Wolfe (FW) algorithm is well-regarded for its projection-free optimization, its slow convergence, especially near the boundary of the feasible region, has prompted the development of several variants. These variants aim to address the limitations of the original algorithm by improving its convergence properties, particularly by achieving linear convergence rates under certain conditions.

5.1. Pairwise Frank-Wolfe

The Pairwise Frank-Wolfe (PFW) variant further refines the concept introduced by AFW. Instead of merely taking away steps, PFW performs a swap between the weights of two atoms: one in the direction of improvement (like in standard FW) and one that is moving away. This pairwise adjustment allows the algorithm to more effectively reduce the error in each iteration, particularly in scenarios where sparse solutions are desired.

Pairwise Frank-Wolfe is particularly useful in high-dimensional problems where maintaining sparsity in the solution is crucial. It also enjoys global linear convergence, similar to the AFW, though with some differences in practical performance depending on the specific problem structure Lacoste-Julien and Jaggi (2015).

5.2. Fully Corrective Frank-Wolfe

The Fully Corrective Frank-Wolfe (FCFW) algorithm extends the AFW and PFW approaches by performing a full re-optimization over the convex hull of all previously selected atoms (the active set) at each iteration. This full correction ensures that the solution remains as close as possible to the optimal point within the feasible set after every step.

While this approach can be computationally more expensive per iteration due to the full re-optimization, it offers significant improvements in convergence speed, especially in cases where the cost of the linear optimization subproblem is high. The FCFW variant is particularly effective when a high degree of accuracy is required, as it ensures that the active set is continuously refined and optimized Lacoste-Julien and Jaggi (2015).

5.3. Wolfe's Minimum Norm Point Algorithm

Wolfe's Minimum Norm Point (MNP) algorithm is closely related to the Fully Corrective Frank-Wolfe but differs in its correction step. Instead of re-optimizing over the convex hull of all atoms, MNP performs a sequence of affine projections to minimize the norm of the point within the polytope. This makes it particularly efficient in applications where the objective function is the Euclidean norm.

The MNP algorithm is often used in scenarios where the feasible region is a polytope, and the goal is to find the point of minimum norm. Its convergence properties are similar to those of FCFW, with the added benefit of efficiency in certain practical applications, such as submodular function optimization Lacoste-Julien and Jaggi (2015).

6. Step Size Strategies

We tested two step size strategies to observe their influence on convergence and performance across three algorithms:

1. **Projected Gradient**
   - **Diminishing Step Size**: Set as $\gamma_k = \frac{2}{k+2}$, ensuring convergence but slowing down as iterations increase.
   - **Line Search**: Dynamically finds the step size by minimizing the objective function along the search direction, typically yielding faster convergence at a higher computational cost.
2. **Frank-Wolfe Algorithm**
   - **Diminishing Step Size**: Set as $\gamma_k = \frac{2}{k+2}$, ensuring convergence but with slower progress in later iterations.
   - **Line Search**: The optimal γ_k is determined by solving $\min_{\gamma \in [0,1]} f(x_k + \gamma d_k)$, typically resulting in better early convergence.
3. **Away-Steps Frank-Wolfe Algorithm**
   - **Line Search**: In the Away-Steps variant, the step size is dynamically determined using line search. The optimal step size is calculated for both Frank-Wolfe and Away directions, maximizing algorithm efficiency and accelerating convergence, especially near the boundary of the feasible set.

7. Projection Methods

In our implementation of the Projected Gradient algorithm, we used two projection methods for projecting the portfolio weights onto the simplex:

- **Standard Simplex Projection**: A classical approach that involves sorting the vector elements and applying thresholding. Its computational complexity is $O(N \log N)$, where N is the number of assets.

- **Optimized Simplex Projection**: Condat's (2016) optimized projection method uses a Gauss-Seidel-like algorithm to dynamically update the threshold during projection, offering faster performance with a practical complexity closer to $O(N)$, making it suitable for large-scale problems.

We evaluated the performance of both projection methods by comparing their execution times across multiple test cases.

8. Key Properties

The following properties apply to all variants of the Frank-Wolfe (FW) algorithm considered in this work.

8.1. Duality Gap

In constrained convex optimization, the duality gap $g(x)$ quantifies how close the current solution is to being optimal and can be expressed as:

$$g(x) := \max_{s\in D}\langle x - s, \nabla f(x)\rangle.$$

The linearization $f(x) + \langle s - x, \nabla f(x)\rangle$ lies below the graph of the function, due to the convexity of f. As a result, it follows that $g(x) \geq f(x) - f(x^*)$, which means the duality gap can be interpreted as a certificate of the current approximation quality.

8.2. Curvature

The curvature constant C_f of a convex and differentiable function $f : \mathbb{R}^n \to \mathbb{R}$, over a compact domain D, is defined as:

$$C_f := \sup_{x,s\in D,\ \gamma\in[0,1]} \frac{1}{\gamma^2}\left(f\left(y\right) - f\left(x\right) - \langle y - x, \nabla f(x)\rangle\right)$$

where $y = x + \gamma(s - x)$.

This constant measures the non-linearity of the objective function f. It captures how much the function deviates from its first-order approximation (i.e., its linearization) when moving from point x in the direction of s.

A higher curvature constant implies that the function deviates more from linearity, which can slow down the convergence of the FW algorithm. Conversely, a lower curvature constant indicates that the function behaves more linearly, leading to faster convergence.

8.3. Primal Convergence Theorem

The primal convergence theorem states that for each $K \geq 1$, the iterates $x^{(k)}$ of the aforementioned algorithms satisfy:

$$f(x^{(k)}) - f(x^*) \leq \frac{2C_f}{k+2}(1+\delta)$$

where $x^* \in D$ is an optimal solution of the problem, and $\delta \geq 0$ is the accuracy to which the internal linear subproblems are solved.

8.4. Convergence in Primal Error

From the previous theorem, it follows that the iterate $x^{(k)}$ of any of the FW variants mentioned before satisfies:

$$f(x^{(k)}) \leq f(x^*) + \epsilon$$

where ϵ is the desired accuracy. This holds after $O\left(\frac{1}{\epsilon}\right)$ iterations.

8.5. Small Duality Gap is Guaranteed

For all variants of the FW algorithm, a small duality gap is guaranteed. Specifically, after $O\left(\frac{1}{\epsilon}\right)$ iterations over an arbitrary bounded domain $D \subseteq X$, the following holds:

$$g(x^{(k)}) \leq \epsilon.$$

This remains true even if the subproblems are solved approximately.

8.6. Primal-Dual Convergence Theorem

If the aforementioned algorithms are executed for $K \geq 2$ iterations, there exists an iterate $x^{(\hat{k})}$ for some $1 \leq \hat{k} \leq K$, where the duality gap is bounded by:

$$g(x^{(\hat{k})}) \leq \frac{2\beta C_f}{K+2}(1+\delta),$$

where $\beta = \frac{27}{8} = 3.375$, and $\delta \geq 0$ is the accuracy to which the internal linear subproblems are solved.

9. Datasets

For the evaluation of our algorithms, we employed three real-world portfolios, each representing different asset classes and market conditions. These datasets provide a realistic basis for testing the performance and robustness of the algorithms in portfolio optimization tasks. By using real-world data, we aim to assess how well the algorithms handle the complexities and constraints typical in financial markets.

9.1. FTSE MIB

This dataset consists of 34 assets from the FTSE MIB Market Index, covering the period from January 2007 to May 2013, with a weekly frequency. The FTSE MIB is a key index representing the performance of 40 of the largest companies listed on the Italian stock exchange. Data for this portfolio were sourced from Yahoo Finance.

9.2. EURO STOXX 50

The EURO STOXX 50 dataset includes 32 assets from the EURO STOXX 50 Index, which tracks the top 50 blue-chip stocks from 11 Eurozone countries. This dataset spans the period from January 2007 to May 2013, with a weekly frequency, and was obtained from Yahoo Finance.

9.3. FTSE 100

This dataset contains 63 assets from the FTSE 100 Index, one of the most widely followed stock market indices in the UK, representing the 100 largest companies listed on the London Stock Exchange. The data cover the period from January 2007 to May 2013, with a weekly frequency, and were retrieved from Yahoo Finance.

All datasets were sourced from the publicly available data repository at Francesco Cesarone's data sets collection.

10. Performance Analysis of Optimization Methods Across Different Datasets

In this section, we will analyze the performance of various optimization methods across three datasets: FTSE100, EUROSTOXX50, and FTSE_MIB. We compare the methods based on their average execution times and the number of iterations required for convergence. The methods include the Projected Gradient algorithm (with both simplex and optimized simplex projection), Frank-Wolfe, and the Away-Step Frank-Wolfe, tested with diminishing and line search step size strategies. For the following section, the following parameter are set:

- Risk aversion parameter = 1
- Stopping criteria threshold (ϵ)= 1e-5
- Maximum number of iterations = 1 / ϵ

10.1. FTSE100 Dataset

In the FTSE100 dataset, the choice of step size strategy significantly impacts the Projected Gradient method's efficiency. Using a diminishing step size results in nearly 100,000 iterations, leading to long execution times due to overly conservative steps as iterations progress. In contrast, line search dynamically adjusts step sizes, reducing execution time to about 0.07 seconds by better aligning with the objective function's landscape. The optimized simplex projection consistently outperforms the standard projection due to its lower computational complexity, which becomes advantageous in larger datasets. The Frank-Wolfe algorithm, especially with line search, shows excellent performance by converging in just 10 iterations, as it balances exploration and exploitation effectively. The graphs (Figures 35 and 36) demonstrate rapid convergence in both the objective function and duality gap, highlighting the efficiency of the projection-free Frank-Wolfe approach. The Away-Step Frank-Wolfe variant further accelerates convergence by correcting trajectory zigzags near the feasible region boundary, reaching the solution in only 5 iterations, as shown in Figures 37 and 38.

10.2. EUROSTOXX50 Dataset

In the EUROSTOXX50 dataset, the Projected Gradient method again struggles with a diminishing step size, resulting in execution times of about 5.85 seconds. The line search strategy, however, significantly reduces this time to 0.04 seconds for simplex projection and 0.06 seconds for the optimized version, due to its adaptive nature. The performance difference between standard and optimized simplex projections is less pronounced here, suggesting that optimized projections are more impactful in larger, more complex datasets. The Frank-Wolfe algorithm, particularly with line search, maintains strong performance, converging in just 5 iterations, while the Away-Step variant matches this efficiency. Graphs show the Away-Step variant's rapid convergence and reduction in duality gap, underscoring its advantage in managing boundary constraints dynamically, as further demonstrated in Figures 23 and 24.

10.3. FTSE_MIB Dataset

For the FTSE MIB dataset, the diminishing step size in the Projected Gradient method leads to slow convergence, with execution times nearing 6 seconds for standard simplex projection and over 10 seconds for optimized projection. This sluggish performance is due to diminishing step sizes that become too small to drive efficient progress. Line search, however, improves execution times to around 0.03 seconds for both projection methods by maintaining appropriately large step sizes. The optimized simplex projection remains slightly more efficient than the standard projection, especially as the dataset size increases. The Frank-Wolfe algorithm continues to excel, with the line search variant converging in just 2 iterations, as depicted in Figures 7 and 8. The Away-Step Frank-Wolfe variant performs best, converging in only 2 iterations with an execution time of 0.0022 seconds. Graphs (Figures 9 and 10) illustrate that the Away-Step variant achieves superior efficiency in reducing the duality gap and improving objective function convergence.

Overall, these observations highlight that diminishing step sizes hinder convergence by becoming overly conservative, while line search strategies adapt better to the optimization landscape, enhancing progress. Optimized simplex projections reduce computational overhead, benefiting larger datasets with more constraints. The Frank-Wolfe algorithm's efficiency stems from its projection-free nature, and the Away-Step variant excels by dynamically correcting its path near boundaries, making these methods particularly suitable for portfolio optimization with complex or boundary-heavy constraints.

Table 1: Average Execution Times for Different Methods and Step Sizes

Dataset	**Method**	**Step Size**	**Average Time**	**Average Iteration**
FTSE100	Projected Gradient (Simplex)	diminishing	6.117898	99998.0
FTSE100	Projected Gradient (Simplex)	line_search	0.077245	39.0
FTSE100	Projected Gradient (Optimized Simplex Projection)	diminishing	13.427553	99998.0
FTSE100	Projected Gradient (Optimized Simplex Projection)	line_search	0.068309	39.0
FTSE100	Frank-Wolfe	line_search	0.013057	10.0
FTSE100	Frank-Wolfe	diminishing	0.001992	77.0
FTSE100	Away-Step Frank-Wolfe	line_search	0.004766	5.0
EUROSTOXX50	Projected Gradient (Simplex)	diminishing	5.846412	99998.0
EUROSTOXX50	Projected Gradient (Simplex)	line_search	0.038211	42.0
EUROSTOXX50	Projected Gradient (Optimized Simplex Projection)	diminishing	9.710399	99998.0
EUROSTOXX50	Projected Gradient (Optimized Simplex Projection)	line_search	0.062249	42.0
EUROSTOXX50	Frank-Wolfe	line_search	0.008895	5.0
EUROSTOXX50	Frank-Wolfe	diminishing	0.001327	24.9
EUROSTOXX50	Away-Step Frank-Wolfe	line_search	0.007481	5.0
FTSE_MIB	Projected Gradient (Simplex)	diminishing	5.858660	99998.0
FTSE_MIB	Projected Gradient (Simplex)	line_search	0.033824	37.0
FTSE_MIB	Projected Gradient (Optimized Simplex Projection)	diminishing	10.446620	99998.0
FTSE_MIB	Projected Gradient (Optimized Simplex Projection)	line_search	0.034548	37.0
FTSE_MIB	Frank-Wolfe	line_search	0.003123	2.0
FTSE_MIB	Frank-Wolfe	diminishing	0.000451	13.0
FTSE_MIB	Away-Step Frank-Wolfe	line_search	0.002156	2.0

11. FTSE MIB

11.1. Projected Gradient

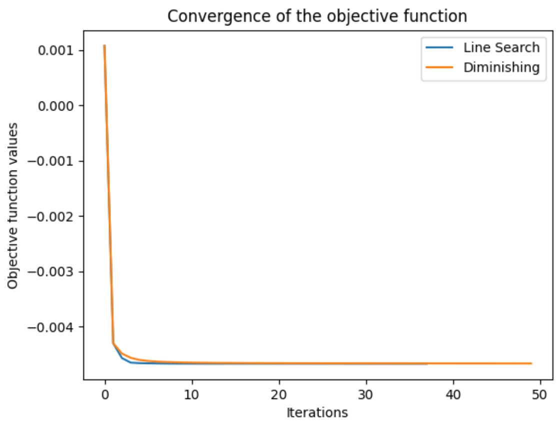


Figure 1: Convergence of the objective function with Simplex Projection Method

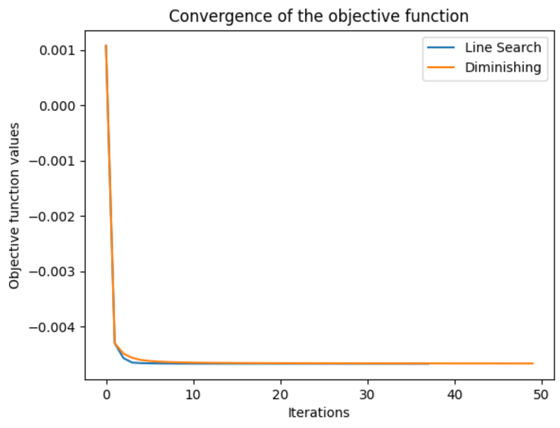


Figure 2: Convergence of the objective function with Optimized Projection Method

Figure 3: Convergence of the objective function with different Projection Method

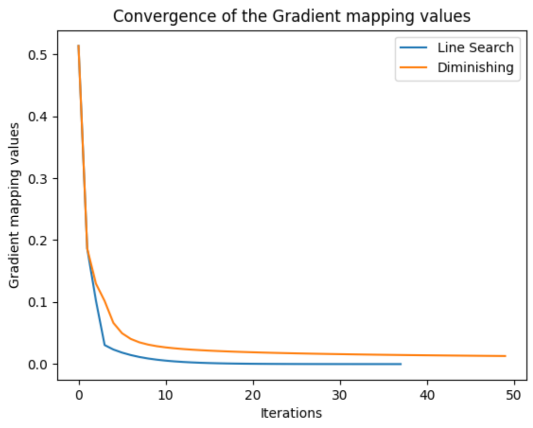


Figure 4: Convergence of the gradient mapping values with Simplex Projection Method

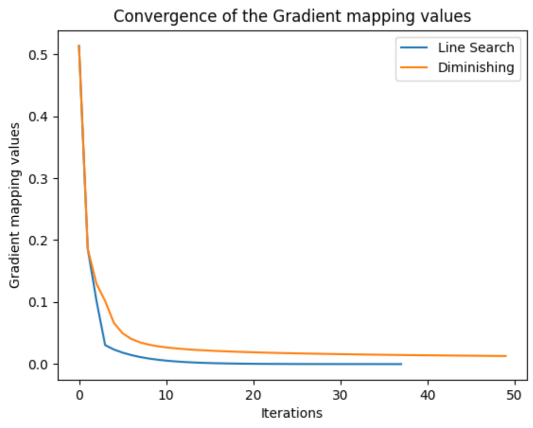


Figure 5: Convergence of the gradient mapping values with Optimized Projection Method

Figure 6: Convergence of the gradient mapping values with different Projection Method

11.2. Frank-Wolfe algorithm

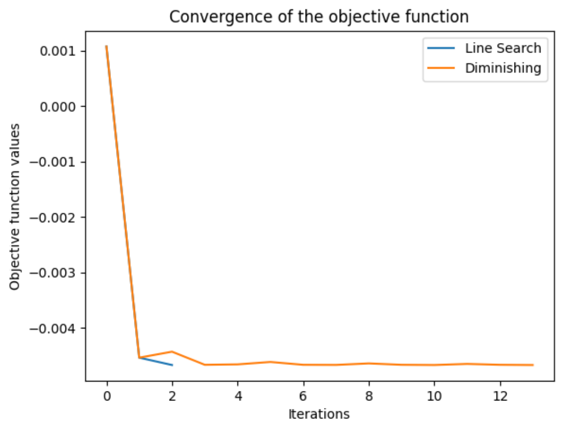


Figure 7: Convergence of the Objective function

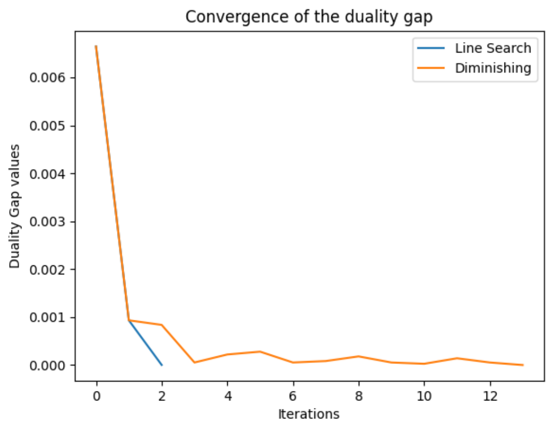


Figure 8: Convergence of the Duality Gap

11.3. Away-steps Frank-Wolfe algorithm

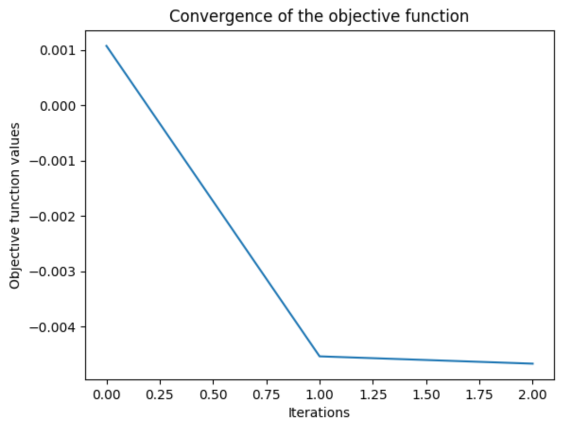


Figure 9: Convergence of the Objective function

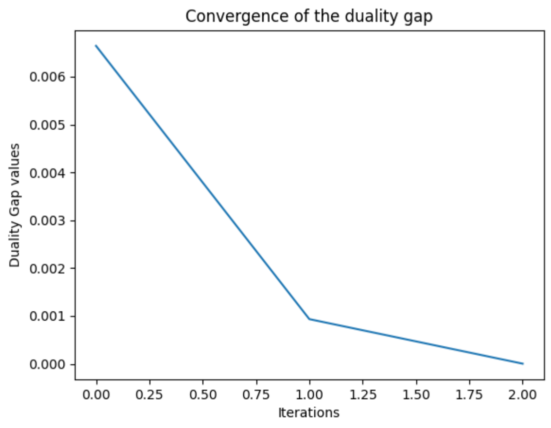


Figure 10: Convergence of the Duality Gap

11.4. Asset allocation after convergence

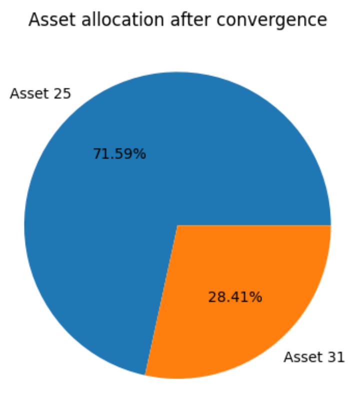


Figure 11: Projected Gradient

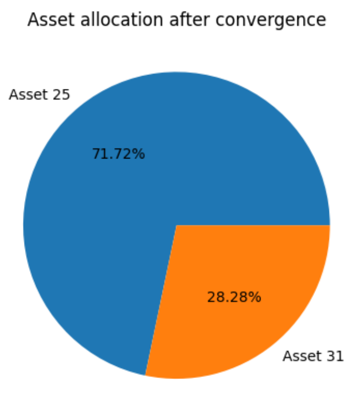


Figure 12: Frank-Wolfe

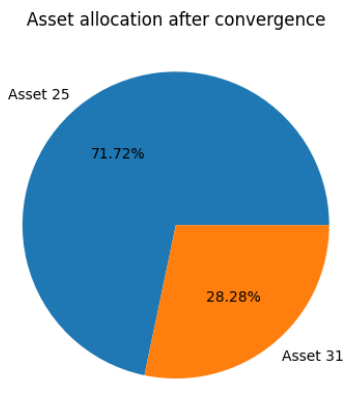


Figure 13: Away-Steps Frank-Wolfe

11.5. Final Comparison

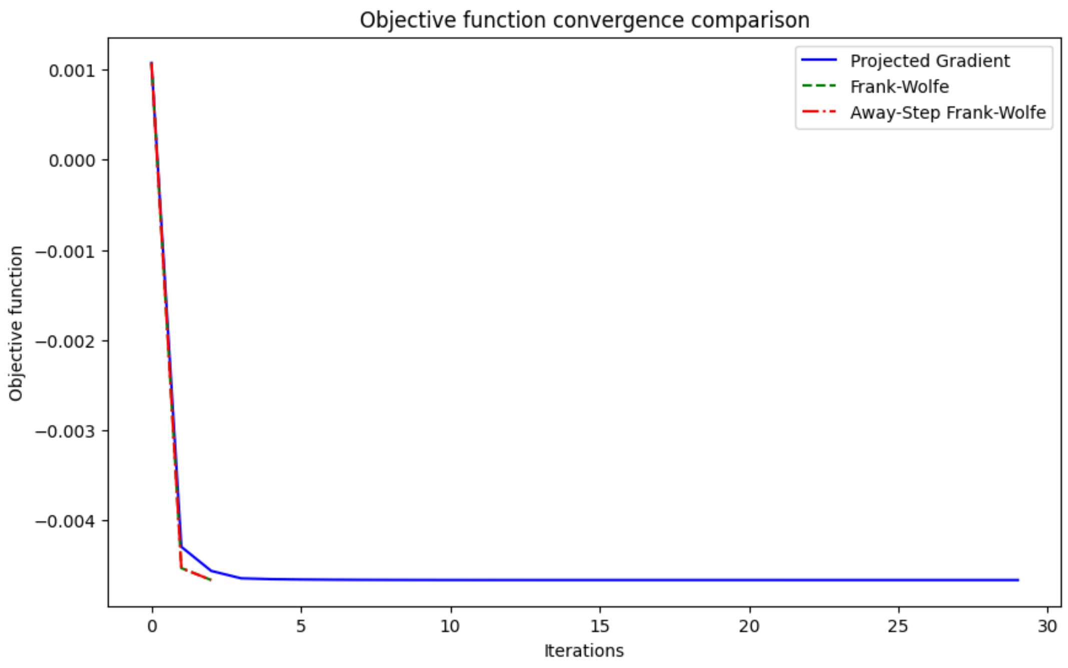


Figure 14: Objective function convergence comparison

12. EUROSTOXX50

12.1. Projected Gradient

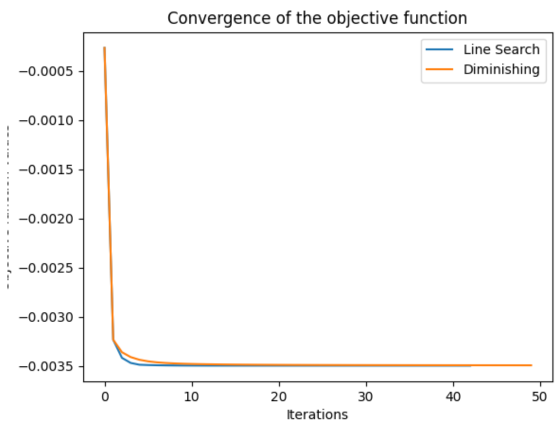


Figure 15: Convergence of the objective function with Simplex Projection Method

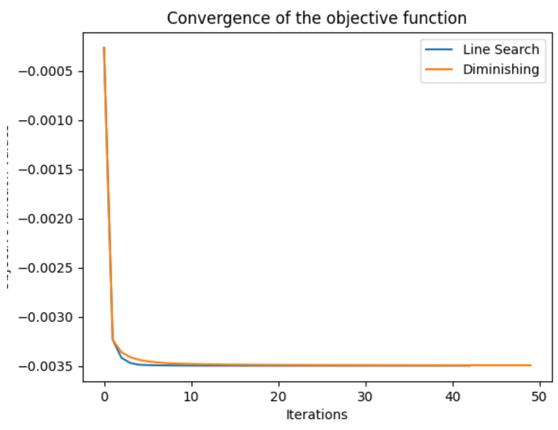


Figure 16: Convergence of the objective function with Optimized Projection Method

Figure 17: Convergence of the objective function with different Projection Method

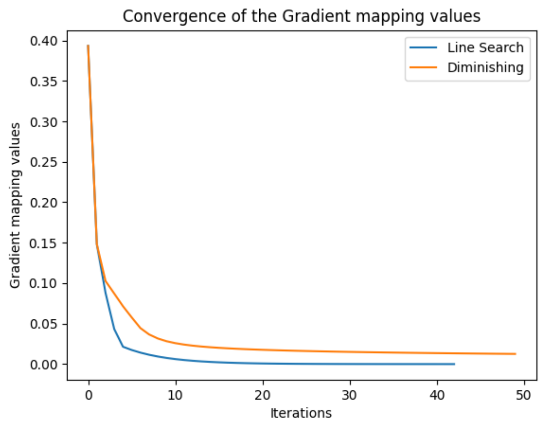


Figure 18: Convergence of the gradient mapping values with Simplex Projection Method

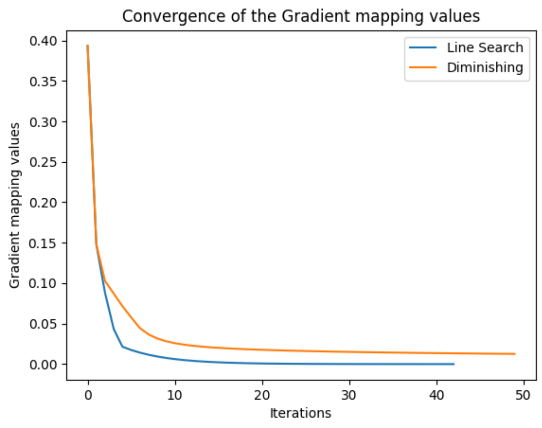


Figure 19: Convergence of the gradient mapping values with Optimized Projection Method

Figure 20: Convergence of the gradient mapping values with different Projection Method

12.2. Frank-Wolfe algorithm

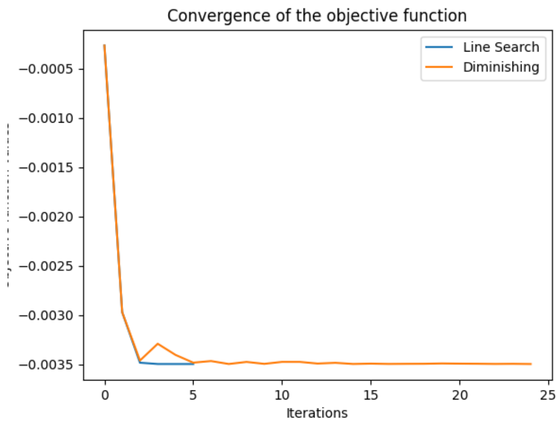


Figure 21: Convergence of the Objective function

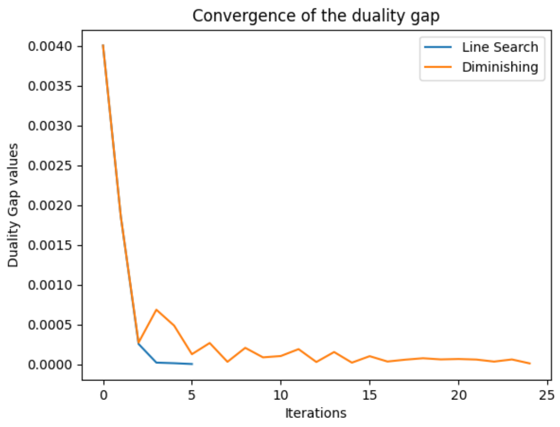


Figure 22: Convergence of the Duality Gap

12.3. Away-steps Frank-Wolfe algorithm

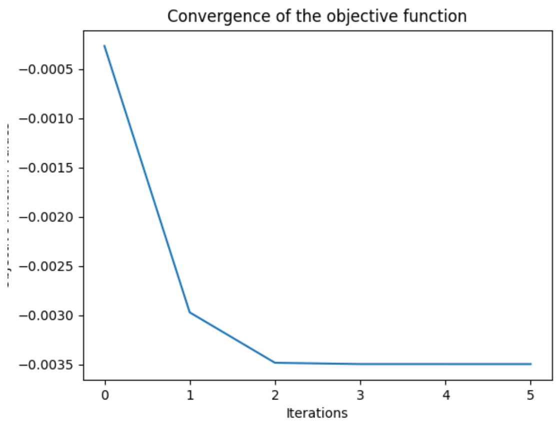


Figure 23: Convergence of the Objective function

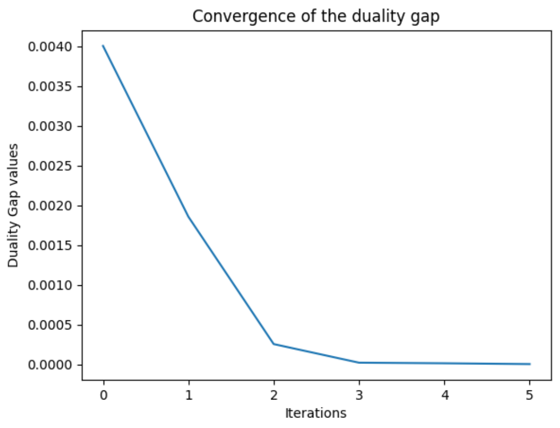


Figure 24: Convergence of the Duality Gap

12.4. Asset allocation after convergence

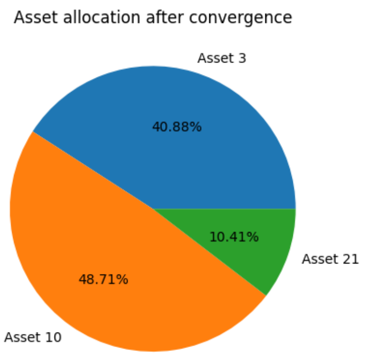


Figure 25: Projected Gradient

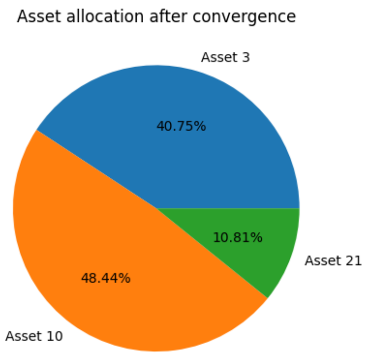


Figure 26: Frank-Wolfe

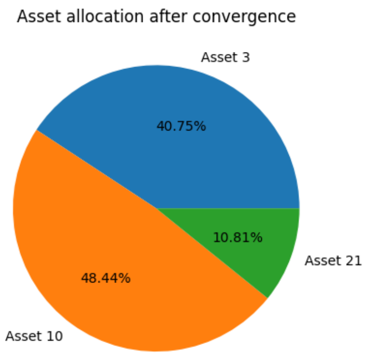


Figure 27: Away-Steps Frank-Wolfe

12.5. Final Comparison

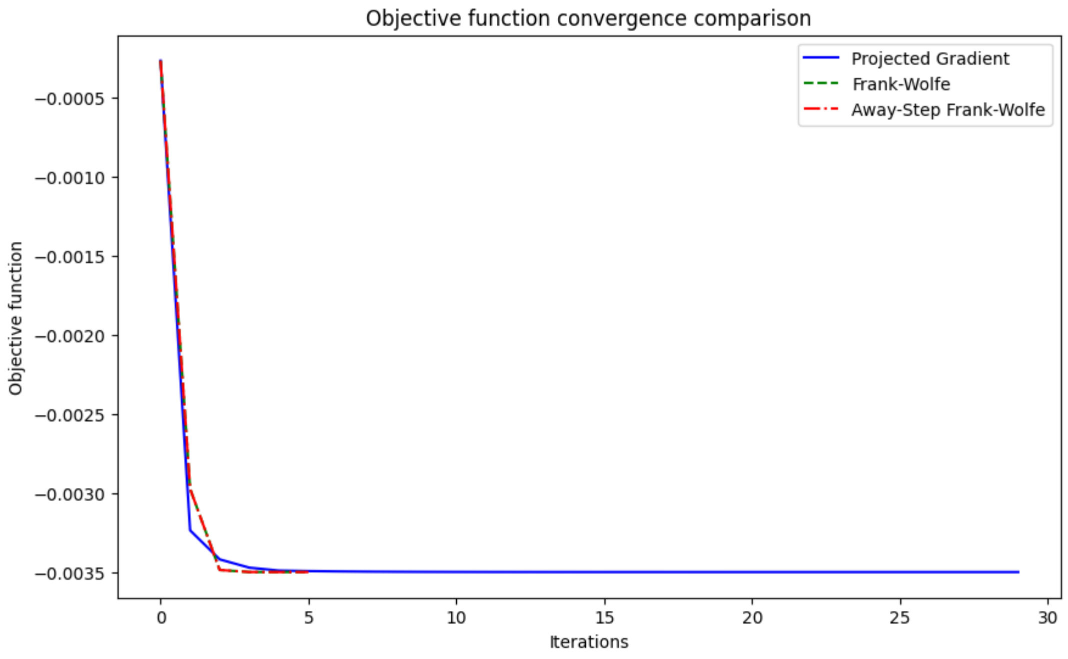


Figure 28: Objective function convergence comparison

13. FTSE100

13.1. Projected Gradient

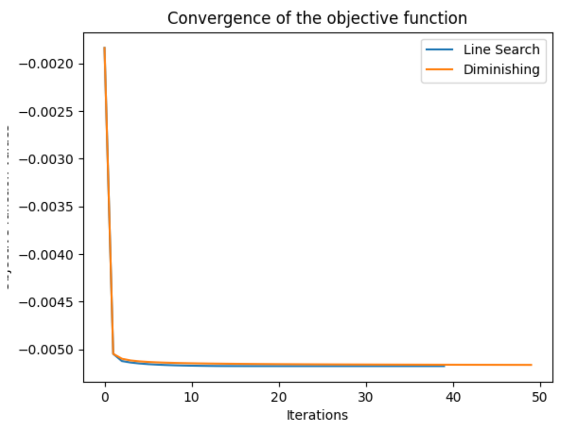


Figure 29: Convergence of the objective function with Simplex Projection Method

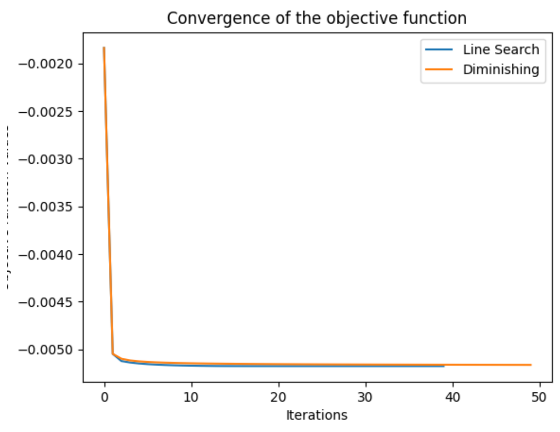


Figure 30: Convergence of the objective function with Optimized Projection Method

Figure 31: Convergence of the objective function with different Projection Method

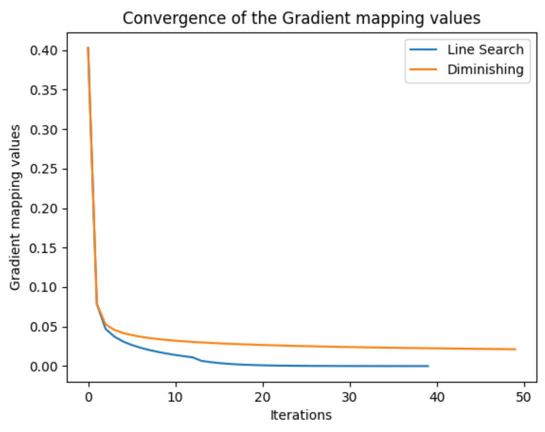


Figure 32: Convergence of the gradient mapping values with Simplex Projection Method

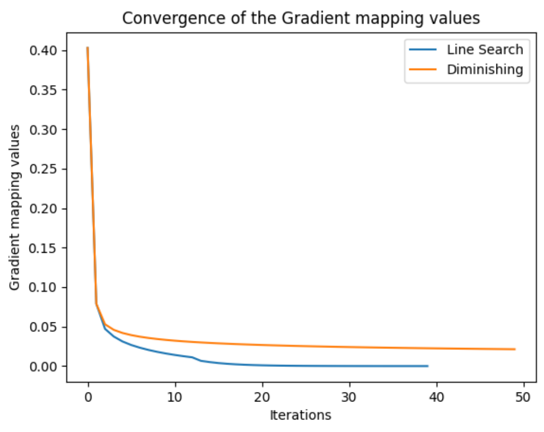


Figure 33: Convergence of the gradient mapping values with Optimized Projection Method

Figure 34: Convergence of the gradient mapping values with different Projection Method

13.2. Frank-Wolfe algorithm

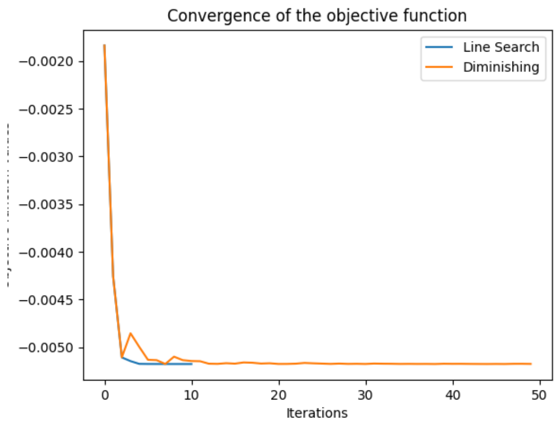


Figure 35: Convergence of the Objective function

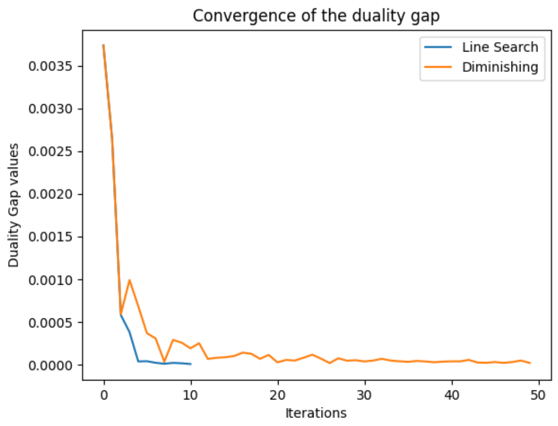


Figure 36: Convergence of the Duality Gap

13.3. Away-steps Frank-Wolfe algorithm

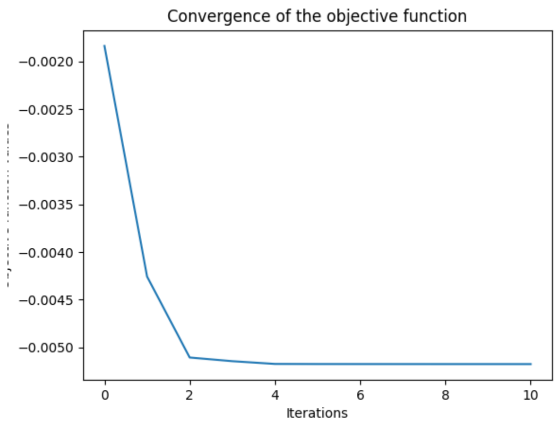


Figure 37: Convergence of the Objective function

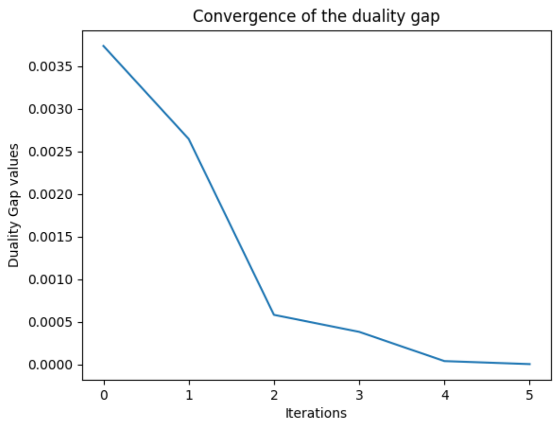


Figure 38: Convergence of the Duality Gap

13.4. Asset allocation after convergence

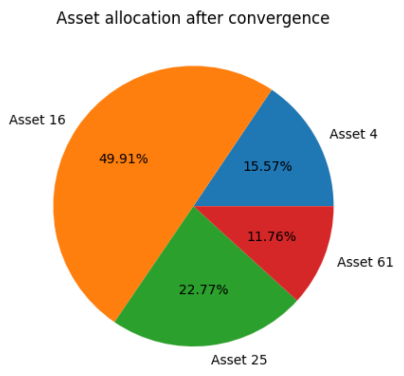


Figure 39: Projected Gradient

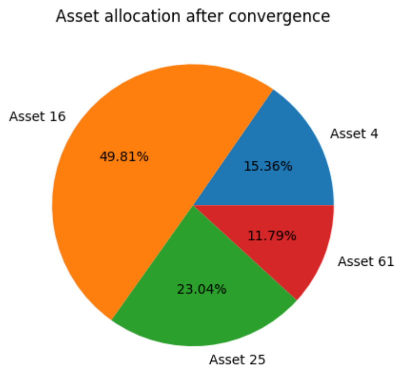


Figure 40: Frank-Wolfe

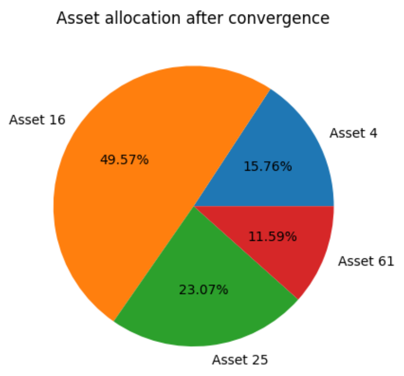


Figure 41: Away-Steps Frank-Wolfe

13.5. Final Comparison

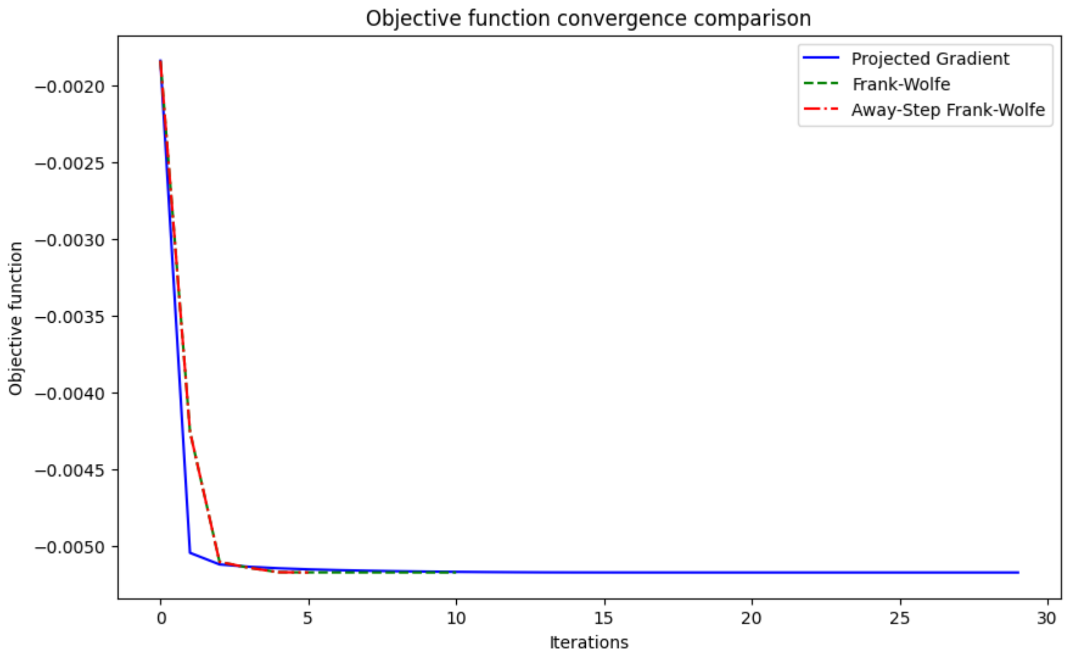


Figure 42: Objective function convergence comparison

14. Effect of the Parameter η

In this section, we analyze the impact of the parameter η, which represents the degree of risk aversion in the portfolio optimization process. As said in previous section, the parameter η controls the trade-off between maximizing expected returns and minimizing risk (measured as the portfolio variance). A higher value of η indicates a more risk-averse behavior, placing greater emphasis on minimizing risk, whereas a lower η suggests a preference for higher returns, even at the cost of taking on more risk. By varying η, we can observe how the portfolio asset weights adjust to reflect different levels of risk tolerance. In particular, we expect that as η increases, the portfolio will become more concentrated in lower-risk assets, while for lower values of η, the portfolio will allocate more weight to assets with higher expected returns, despite their associated volatility. The analysis in this section provides insights into how sensitive the portfolio allocation is to changes in the investor's risk preference, and it highlights the importance of selecting an appropriate η based on individual risk tolerance.

14.1. Pie chart with change in the parameter ETA

The experiment will be run in the FTSE 100 Dataset.

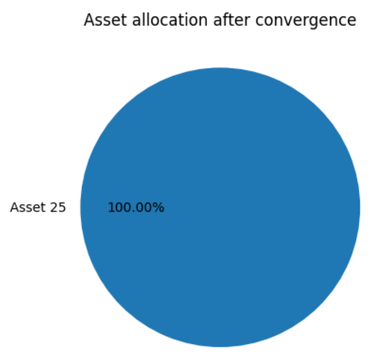


Figure 43: ETA = 0.1

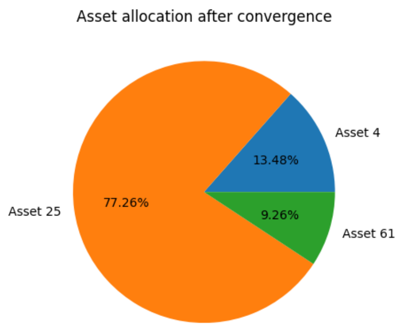


Figure 44: ETA = 0.25

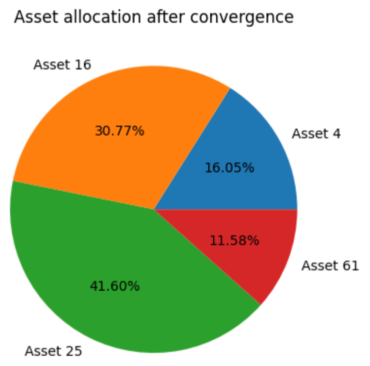


Figure 45: ETA = 0.5

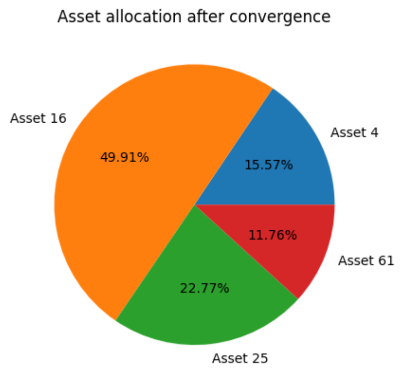


Figure 46: ETA = 1

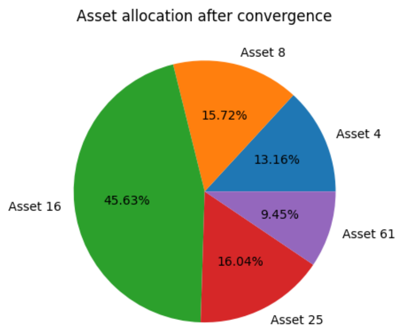


Figure 47: ETA = 1.5

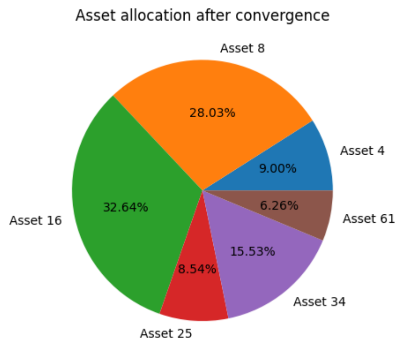


Figure 48: ETA = 3

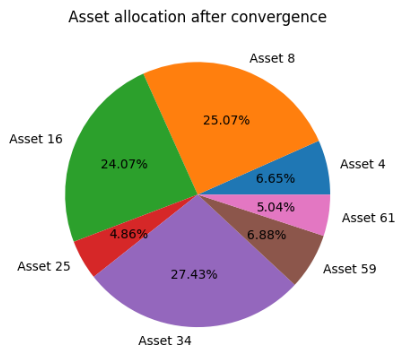


Figure 49: ETA = 5

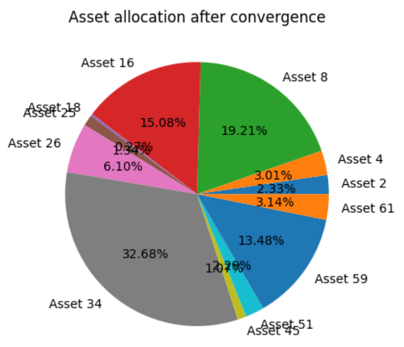


Figure 50: ETA = 10

14.2. Comment

- **Low η (Aggressive Portfolio)**:
  For very low values of η (e.g., $\eta = 0.1$ and $\eta = 0.25$), the portfolio is almost entirely concentrated in one or two assets. These assets typically offer higher expected returns, although they may come with higher volatility. This behavior aligns with the expectation that a low η allows the algorithm to prioritize return maximization over risk minimization.

- **Moderate η (Balanced Risk and Return)**:
  As η increases to moderate levels (e.g., $\eta = 0.5$, $\eta = 1$, and $\eta = 1.5$), the asset allocation becomes more diversified. At these values, the optimization process begins to balance return with a focus on reducing risk. The allocation is spread across several assets, reflecting a more balanced portfolio that takes both risk and return into account.

- **High η (Conservative Portfolio)**:
  For higher values of η (e.g., $\eta = 3$, $\eta = 5$, $\eta = 10$), the portfolio becomes increasingly conservative. The weights shift towards assets with lower variance, reducing exposure to riskier assets, and the allocation becomes more evenly distributed across a broader range of assets. This reflects a strong preference for risk minimization, with the algorithm prioritizing stability over the pursuit of higher returns.

15. Conclusion

This study examined the performance of the Projected Gradient, Frank-Wolfe, and Away-Step Frank-Wolfe algorithms on three datasets: FTSE100, EUROSTOXX50, and FTSE_MIB. The comparison focused on execution time, iteration count, and the effect of the risk-aversion parameter η on portfolio composition.

Across all datasets, the Frank-Wolfe and Away-Step Frank-Wolfe algorithms demonstrated superior performance, particularly when employing a line search strategy. These methods significantly reduced the number of iterations and computational time compared to the Projected Gradient method, which exhibited slow convergence, especially when using a diminishing step size. Switching the Projected Gradient method to a line search step size improved its performance, but it still lagged behind the Frank-Wolfe variants in terms of speed and efficiency.

In addition to algorithm performance, the analysis of the risk-aversion parameter η revealed that it plays a pivotal role in shaping the portfolio's asset allocation.

In conclusion, the Frank-Wolfe and Away-Step Frank-Wolfe algorithms, particularly with line search step sizes, are more efficient for the optimization problems considered in this study. The choice of risk-aversion parameter η significantly affects the portfolio's risk-return profile, with higher values promoting conservative allocations. These findings highlight the importance of selecting the appropriate algorithm and step size strategy, as well as fine-tuning η, based on the desired risk and return trade-offs.

References

Laurent Condat. Fast projection onto the simplex and the ℓ_1 ball. *Mathematical Programming, Series A*, 158:575–585, 2016. doi: 10.1007/s10107-015-0946-6.

Simon Lacoste-Julien and Martin Jaggi. On the global linear convergence of frank-wolfe optimization variants. *Advances in Neural Information Processing Systems*, pages 496–504, 2015. doi: 10.5555/2969239.2969301.

Francesco Rinaldi. Lecture slides on 'constrained problems in data science', 2023. URL `https://stem.elearning.unipd.it/course/view.php?id=5040`. Lecture Notes, Lecture 15, 2022/2023.