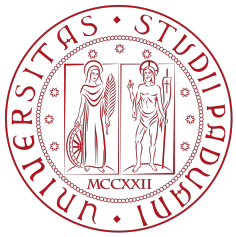

Università degli Studi di Padova

Department of Mathematics
Optimization for Data Science

FIRST ORDER OPTIMIZATION METHODS: a Comparative Analysis of the Projected Gradient Method and the Frank-Wolfe Algorithm on Portfolio Optimization

Group Components:
Lazzari Tommaso, Ludergnani Brenno, Movila Dumitru, Safa Nasser

September 2025

Contents

1 Introduction

Portfolio optimization is the process of selecting assets to achieve the best possible outcome in terms of risk and return. The foundational mean-variance model developed by Harry Markowitz frames this as a constrained, convex quadratic programming problem, making it an ideal test case for constrained first-order optimization methods. This study evaluates and compares the performance of the Frank-Wolfe (FW) algorithm, its two variants—the Away-Step Frank-Wolfe (AFW) and Pairwise Frank-Wolfe (PFW)—and the Projected Gradient method specifically on this problem.

The Frank-Wolfe algorithm is particularly well-suited for this task due to its ability to handle structured constraints, its good scalability, and its crucial property of producing sparse solutions. However, the classical FW method is known for its slow (sublinear) convergence rate when the optimal solution lies at the boundary of the feasible set. To address this limitation, this analysis also explores the AFW and PFW variants, which are highlighted for their enhanced convergence properties and robustness. The ultimate aim is to compare the performance of these algorithms through both theoretical analysis and empirical studies using four different datasets.

2 Markowitz Formulation

Assume to have n available assets. We call x_i the amount of money invested in the i-th asset during the period of time considered, while r_i are the returns on the assets.

On the money quantity, we have two types of constraint: the budget constraint and the non-negativity constraint. The budget constraint regards the total amount of money that can be invested:

$$\sum_{i=1}^{n} x_i = B$$

this can be simply set equal to 1. On the other hand, the non-negativity constraint involves the fact that short selling is not allowed in this formulation; thus we have:

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

While the vector $r \in \mathbb{R}^n$, the entries representing the returns r_i, is randomly generated by a stochastic model characterized by mean $\bar{r}$ and covariance matrix Σ. Thus the expected return is modeled as:

$$\bar{r}^t x$$

while the variance is modeled as:

$$x^t \Sigma x$$

The classic portfolio problem described by Markowitz is:

$$\begin{gathered}\max_{x\in\mathbb{R}^n} \eta\bar{r}^t x - x^t\Sigma x \\ \sum_{i=1}^n x_i = 1 \\ x_i \geq 0, \forall i\end{gathered}$$

where $\eta \in [0, +\infty)$ is the risk tolerance parameter. Which can be modeled as a minimization problem:

$$\begin{gathered}\min_{x\in\mathbb{R}^n} x^t\Sigma x - \eta\bar{r}^t x \\ \sum_{i=1}^n x_i = 1 \\ x_i \geq 0, \forall i\end{gathered}$$

Where $\eta \to \infty$ represent a higher risk tolerance favoring higher returns, while a lower $\eta \approx 0$ favors a more cautious investment strategy, prone to diversification.

The goal is to find the set of assets that minimizes the risk connected to a given portfolio (variance), while maximizing the expected return with respect to the constraints. This can be done because of the particular structure of the objective function, which consists of a linear part and a quadratic part. The linear part

$$\eta\bar{r}^t x$$

is a linear function. The quadratic part:

$$x^t\Sigma x$$

is constituted by the covariance matrix Σ, which is inherently positive semi-definite. Hence $x^t\Sigma x$ is a convex function. Since the sum of convex functions is still convex, the objective function:

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

is convex. This guarantees the existence and uniqueness of its global minimum. Also any local minima found by an optimization algorithm is indeed the global minimum.

2.1 Optimization Problem

By the description of the Portfolio Optimization problem reported previously, we derive the feasible set as:

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

this geometric object is called unit simplex also known as standard simplex. The unit simplex is a convex set and it's also closed and bounded. This implies C is a compact set. Particularly C can be represented as the convex hull of the standard basis vectors of $\mathbb{R}^n$. Which vertices represent the cases in which all the budget is invested on a single asset.

Under these hypotheses, the linear minimization oracle will always choose the canonical basis vector with non-negative entry corresponding the most negative entry of the objective function's gradient in the current solution. Which means:

$$s = LMO_C(x) = \arg\min_{z \in C} \nabla f(x)^t z$$

hence $s = e_j$ such that $j = \arg\min \nabla_i f(x)$. Where in the case of portfolio optimization, the gradient of the objective function is computed as:

$$\nabla f(x) = 2\Sigma x - \eta\mu$$

where $\mu := \bar{r}$ and η is the previously defined risk tolerance constant. The LMO selects the index $j \in \{1, \ldots, n\}$ associated with the smallest entry of the gradient. This means the LMO is efficient, since finding the minimum entry requires $O(n)$ operations at each call and does not need any projection process. These characteristics favor the use of Frank-Wolfe iteration method.

3 Frank-Wolfe Iteration Method

Consider a general constrained convex optimization problem of the form:

$$\min_{x \in C} f(x)$$

we assume that the objective function $f : \mathbb{R}^n \to \mathbb{R}$ is convex and continuously differentiable, and that the domain C is a compact, convex subset of any vector space.

For such optimization problems one of the simplest and earliest known iterative optimizer is the Conditional Gradient Method also called Classical Frank-Wolfe Method (FW).

It was first introduced by Marguerite Frank and Philip Wolfe in 1956. The algorithm iteratively leverages a linear approximation of the objective function to generate a sequence of feasible iterates, avoiding costly projections while maintaining convergence guarantees. The "projection-free" peculiarity has renewed interest in the method, especially in large-scale data science applications where projection operations can be computationally expensive.

3.1 Algorithm Description

At each iteration k, the algorithm solves the sub-problem:

$$s^{(k)} = \arg\min_{s \in C} \langle \nabla f(x^{(k)}), s \rangle$$

finding the new vertex s that minimizes the inner product with the gradient.

The new iterate is computed as:

$$x^{(k+1)} = (1 - \gamma_k)x^{(k)} + \gamma_k s^{(k)}$$

This is a convex combination of the two points, since the stepsize $\gamma_k \in [0, 1]$. The latter can be set adopting different strategies, such as the exact line search or predefined schemes such as the diminishing steps strategy.

The structure of the Classical Frank-Wolfe algorithm is the following:

1. **Initialization**: choose the starting point $x^{(0)} \in C$.
2. **Direction-Finding Sub-Problem**: solve the linear sub-problem to identify the FW atom $s^{(k)}$ and the descent direction $d^{(k)} = x^{(k)} - s^{(k)}$.
3. **Step-Size Determination**: find the step-size γ_k suitable for the iteration k.
4. **Iterate Update.**
5. **Termination**: check the convergence criteria and terminate if satisfied; otherwise, repeat from step 2.

Algorithm 1 Classical Frank-Wolfe Algorithm.

Input: Convex function f, compact convex domain C.
Initialize: $x^{(0)} \in C$.
for $i = 1$ to $\ldots$ **do**
  Compute the linear subproblem: $s^{(k)} = \arg\min_{s \in C} \langle \nabla f(x^{(k)}), s \rangle$
  Choose the step size γ_k.
  Update the solution $x^{(k+1)} = (1 - \gamma_k)x^{(k)} + \gamma_k s^{(k)}$
end for

3.2 Unit Simplex Case

In the context of the portfolio optimization problem, we consider the special case in which the domain C is represented by the unit simplex $\Delta_{n-1} \subseteq \mathbb{R}^n$. In this case the linear sub-problem is computationally cheap, since $\hat{x}_k = e_{i_k}$ where $i_k = \arg\min_i \nabla_i f(x^{(k)})$.

When considering the unit simplex as feasible set:

$$C = \left\{ x \in \mathbb{R}^n : e^t x = 1, x \geq 0 \right\}$$

such that this is the convex hull of the vector of the canonical basis, we have that:

3.3 Frank-Wolfe Convergence Analysis

The Frank-Wolfe method is guaranteed to converge under general convexity assumptions, achieving a rate of $O(1/k)$ when the objective function is smooth and convex. This baseline rate can be enhanced under stronger assumptions, such as strong convexity or when the optimum lies in the relative interior of

Algorithm 2 Classical Frank-Wolfe Algorithm. Unit simplex case.

1: **Input**: Convex function f, compact convex domain C.
2: **Initialize**: $x^{(0)} \in C$.
3: **for** $i = 1$ to $\ldots$ **do**
4: Compute the linear subproblem: $s^{(k)} = e_{i_k} = \arg\min_{s \in C} \nabla_i f(x^{(k)})$
5: Choose the step size γ_k.
6: Update the solution $x^{(k+1)} = (1 - \gamma_k)x^{(k)} + \gamma_k s^{(k)}$
7: **end for**

the feasible region. In fact, for strongly convex problems, the method can even attain linear convergence given suitable conditions. The algorithm is also robust in the presence of inexact oracles, where the linear subproblems are solved only approximately. Even in this setting, convergence guarantees are preserved, which makes the method practical in applications where computing exact solutions is too costly. A central strength of the FW procedure lies in the fact that it preserves feasibility at every step. This feature is especially advantageous in structured optimization problems, where enforcing feasibility is often difficult. Another important property is its affine invariance: the algorithm's performance does not depend on affine transformations of the feasible domain. This ensures that scaling or translating the problem has no negative effect on its efficiency. The convergence of the FW algorithm is formally established in **Theorems 1 and 2**, where the rate depends on the curvature constant C_f and on the precision δ used in solving the linear subproblems. The method guarantees both primal and primal-dual convergence, which underlines its reliability for constrained convex optimization. Specifically, the primal convergence theorem bounds the primal error, ensuring its decrease at a rate of $O(1/k)$, while the primal-dual convergence theorem provides an equivalent rate for the duality gap. These results are particularly significant in scenarios where exact projections are computationally demanding or impractical, highlighting the FW algorithm's usefulness in large-scale optimization tasks.

Theorem 1. ***(Primal Convergence)*** *For each $k \geq 1$, the iterate $x^{(k)}$ of Frank-Wolfe algorithm and its variants, satisfies:*

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

where $x^* \in C$ is an optimal point, $x^{(k)}$ is the current solution at step k, $\delta \geq 0$ is the accuracy to which the internal linear subproblems are solved and C_f is the curvature constant of the objective function f.

For any constrained convex optimization problem of the form:

$$\min_{x \in C} f(x)$$

and a feasible point $x \in C$ we define the following simple surrogate of duality gap:

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

Convexity of f implies that the linearization $f(x)+\langle s-x, \nabla f(x)\rangle$ always lies below the graphic of the function f. This immediately gives a crucial property of the duality gap, as being the certificate of the current approximation quality i.e. $g(x) \geq f(x) - f(x^*)$.

Theorem 2. ***(Primal-Dual Convergence)*** *If Frank-Wolfe algorithm or any of its variants is run for $K \geq 2$ iterations, then the algorithm has an iterate $x^{(\hat{k})}$ with $1 \leq \hat{h} \leq K$ with duality gap bounded by:*

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

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

3.4 Frank-Wolfe Variants

To improve its efficiency and broaden its applicability to diverse problem settings, many extensions of the FW method have been introduced:

- **Diminishing Step Variant**: A predefined decreasing sequence step size choosing rule, which usually is of the form

  $$\gamma_k = \frac{2}{k+2}$$

  This schedule ensures that $\gamma_k \to 0$ as $k \to \infty$, while still being large enough in early iterations.

- **Line-Search Variant**: Performs a line search to find the optimal step size that minimizes:$f(x^{(k)} + \gamma_k(s - x^{(k)}))$. So that:

  $$\gamma_k = \arg\min_{\gamma \in [0,1]} f(x^{(k)} + \gamma(s - x^{(k)}))$$

  Selecting the most suitable γ_k at every iteration ensures faster convergence and more efficient steps.

- **Away-Step Frank-Wolfe (AFW)**:This variant introduces "away step", thus the name, which allows the method to step away from an active vertex of the feasible set and mitigate the zig-zagging effect that often arises near the boundary of the feasible region. By doing so, it improves convergence behavior, particularly in problems where the feasible domain contains sharp vertices.

- **Pairwise Frank-Wolfe (PFW)**: This variant improves descent directions by combining pairs of feasible points, which leads to faster convergence through more refined updates. Instead of relying solely on the current iterate and a new vertex, it performs pairwise moves between the present point and previously selected vertices, offering an effective strategy to reduce the zig-zagging phenomenon.

4 Away-Step Frank-Wolfe

The Away Step Frank-Wolfe (AFW) is a variant of the Classical Frank-Wolfe algorithm aiming to enhance convergence rate in optimization problems over polytopes. It was specifically designed to address the zig-zagging behavior of Classical Frank-Wolfe when the optimal solution of the problem is close to the boundary of the feasible set. In the Away Step Frank-Wolfe the algorithm keeps track of the visited vertices by storing them in the active set $S^{(k)}$, which represents the subset of the set of vertices, though which we can represent the current solution $x^{(k)}$ as a convex combination of extremal points.

In the AFW algorithm, not only we compute the classical descent direction using the output of the LMO to produce $d_{FW}^{(k)} = s^{(k)} - x^{(k)}$, but we also compute the away step direction $d_A^{(k)} = x^{(k)} - v^{(k)}$ where $v^{(k)}$ is the "worst vertex" belonging to the active set, obtained as:

$$v_A^{(k)} = \arg\max_{v \in S^{(k)}} \langle \nabla f(x^{(k)}), v \rangle$$

this is the vertex that is most aligned with the gradient of the objective function.

The Away-Step FW can decide either to use the classical descent direction $d_{FW}^{(k)}$ or to use the away direction $d_A^{(k)}$, based on which direction would bring the best improvement in the objective function with respect to $\nabla f(x^{(k)})$.

We observe that during each classical FW steps a vertex may or may not be added to the active set, this means $|S^{(k+1)}| \geq |S^{(k)}|$.

While during each away-step the size of the active set can either decreases or remain the same, $|S^{(k+1)}| \leq |S^{(k)}|$. Assume $d_A^{(k)}$ provides the best improvement in the objective function, than the next iterate is defined as:

$$x^{(k+1)} = x^{(k)} + \alpha_k d_A^{(k)} = x^{(k)} + \alpha_k (x^{(k)} - v^{(k)}) = \left[(1 + \alpha_k) \sum_{v \in S^{(k)}} \gamma_v v \right] - \alpha_k v^{(k)}$$

where α_k is found through the line-search method and all γ_v form a convex combination. Since $v^{(k)} \in S^{(k)}$, this vertex is included in the convex combination describing the current solution. Then if:

$$(1 + \alpha_k)\gamma_{v^{(k)}} - \alpha_k = 0 \iff \alpha_k = \frac{\gamma_{v^{(k)}}}{1 - \gamma_{v^{(k)}}}$$

the vertex $v^{(k)}$ is eliminated from the description of $x^{(k)}$, hence from the active set. Thus we would obtain $|S^{(k+1)}| < |S^{(k)}|$. We would call this a drop-step.

Otherwise, the weight of $v^{(k)}$ in the convex combination would be modified by the factor $-\alpha_k$ while the active set remains the same $|S^{(k+1)}| = |S^{(k)}|$.

4.1 Algorithm Description

The Away-Step Frank-Wolfe Algorithm is characterized by the following steps:

1. **Initialization**: Start with an initial point $x^{(0)} \in C$ and initialize the active set $S^{(0)} = \{x^{(0)}\}$.
2. **Direction-Finding Subproblem**: Solve the linear subproblem to find the FW direction $d_{FW}^{(k)}$.
3. **Away-Step Direction**: Solve the linear subproblem to find the away vertex $v^{(k)}$ and define $d_A^{(k)} = x^{(k)} - v^{(k)}$.
4. **Direction choice**: choose the direction leading to the best improvement between $d_{FW}^{(k)}$ and $d_A^{(k)}$.
5. **Step-Size Determination**: Compute the step-size γ_k
6. **Iterate update.**
7. **Termination**: check the convergence criteria and stop if satisfied. Otherwise repeat from step 2.

Algorithm 3 Away-Step Frank-Wolfe Algorithm.

Input: Convex function f, compact convex domain C.
Initialize: $x^{(0)} \in C$ and $S^{(0)} = \{x^{(0)}\}$.
for $i = 1$ to $\ldots$ **do**
  Compute the linear subproblem: $\hat{x}_{FW}^{(k)} = \arg\min_{s \in C} \langle \nabla f(x^{(k)}), s \rangle$
  if $\hat{x}_{FW}^{(k)}$ satisfies some conditions **then**
    Stop.
  else
    Set $\hat{x}_A^{(k)} = \arg\max_{x \in s^{(k)}} \nabla f(x^{(k)})^t (x - x^{(k)})$
    Set $d_{FW}^{(k)} = \hat{x}_{FW}^{(k)} - x^{(k)}$ and $d_A^{(k)} = \hat{x}_A^{(k)} - x^{(k)}$
    if $\nabla f(x^{(k)})^t d_{FW}^{(k)} \leq \nabla f(x^{(k)})^t d_A^{(k)}$ **then**
      set $d^{(k)} = d_{FW}^{(k)}$ and $\bar{\alpha} = 1$
    else
      set $d^{(k)} = d_A^{(k)}$ and $\bar{\alpha} = \max_\beta \left\{ x^{(k)} + \beta d_A^{(k)} \right\} \in C$
      set $x^{(k+1)} = x^{(k)} + \alpha_k d^{(k)}$ with $\alpha_k \in (0, \bar{\alpha})$
      Compute $s^{(k+1)}$ set of the current used vertices.
    end if
  end if
end for

4.2 Away-Step Frank-Wolfe Convergence analysis

The AFW algorithm enjoys stronger convergence guarantees compared to the classical FW method, particularly when the solution lies on the boundary of the feasible set. The introduction of away steps enables the method to achieve linear convergence under suitable conditions. Specifically, when the objective function

is strongly convex and the feasible domain is a polytope, AFW ensures that the primal suboptimality decreases at a geometric rate. This behavior is explained through the notion of a geometric strong convexity constant, which couples the condition number of the objective with a polytope-dependent constant capturing the geometry of the feasible set. Formally, under strong convexity one obtains a bound of the form:

$$f(x^{(k)}) - f(x^*) \leq \Big(f(x^{(0)}) - f(x^*)\Big) \exp(-\rho k)$$

with $\rho > 0$, k number of iterations, x^* optimal point. Particularly ρ is determined by the curvature of the function and the characteristics of the feasible set.

A crucial element of this analysis is the subdivision between "good steps" that guarantee a sufficient decrease in the objective function and "bad steps" represented by drop-steps in which we just eliminate a "bad vertex" from the current solution description.

Since the number of bad steps performed until step k is upper bounded by $k/2$, progresses are consistently made and AFW still recovers $O(1/k)$ convergence rate.

These results highlight AFW as a robust and theoretically well-grounded variant, especially effective for structured optimization tasks where the solution lies on the boundary of the feasible region.

5 Pairwise Frank-Wolfe

The Pairwise Frank-Wolfe (PFW) algorithm is a refinement of the classical Frank-Wolfe method designed to overcome some of its inherent drawbacks. The PFW variant improves convergence by incorporating more sophisticated update steps, thereby increasing both the efficiency and the practical applicability of FW. PFW exploits the geometry of the feasible set by working with convex combinations of the vertices involved. Unlike the classical approach, which only moves towards a new vertex, PFW considers both the current iterate and atoms from the active set, striking a balance between exploring new directions and correcting past choices. This strategy not only accelerates convergence but also promotes sparsity in the active set, which is particularly beneficial for problems where sparse solutions are desired.

5.1 Algorithm Description

The PFW method extends the classical FW algorithm by introducing pairwise moves. Instead of advancing solely toward the current linear minimizer, it also allows movement away from vertices already included in the active set. This mechanism reduces the zig-zagging effect and leads to faster convergence. The Pairwise Frank-Wolfe Algorithm is characterized by the following steps:

1. **Initialization**: Start with an initial point $x^{(0)} \in C$ and initialize the active set $S^{(0)} = \{x^{(0)}\}$.

2. **Direction-Finding Subproblem**: Solve the linear subproblem to find the atom $s^{(k)}$.

3. **Away-Step Direction**: Solve the linear subproblem to find the away vertex $v^{(k)}$.

4. **Pairwise Direction**: Compute the pairwise direction as $d_{PFW}^{(k)} = s^{(k)} - v^{(k)}$.

5. **Step-Size Determination**: Compute the step-size γ_k

6. **Iterate update.**

7. **Termination**: check the convergence criteria and stop if satisfied. Otherwise, repeat from step 2.

Algorithm 4 Pairwise Frank-Wolfe Algorithm.

Input: Convex function f, compact convex domain C.
Initialize: $x^{(0)} \in C$ and $S = \{x^{(0)}\}$.
for $i = 1$ to $\ldots$ **do**
  Solve for $s^{(k)}$ the linear sub-problem:

$$s^{(k)} = \arg\min_{s \in C} \langle \nabla f(x^{(k)}), s \rangle$$

  Identify $v^{(k)} \in S^{(k)}$ such that:

$$v^{(k)} = \arg\max_{v \in S^{(k)}} \langle \nabla f(x^{(k)}), v \rangle$$

  .
  Compute $d^{(k)} = s^{(k)} - v^{(k)}$.
  Choose γ_k that minimizes $f(x^{(k)} + \gamma_k d^{(k)})$
  Update $x^{(k)}$ and $S^{(k)}$
end for

5.2 Pairwise Frank-Wolfe Convergence Analysis

The PFW method offers improved convergence behavior compared to the classical FW algorithm. By using pairwise steps, it can move from suboptimal vertices more effectively, thereby advancing towards the optimal solution in fewer iterations. Under general convexity assumptions, the PFW algorithm maintains the $O(1/k)$ convergence rate of the original FW method, but in practice it often converges more quickly thanks to more effective direction updates. In the case of strongly convex functions, the PFW algorithm achieves linear convergence, which substantially enhances its efficiency relative to the classical approach. This faster rate is particularly valuable in scenarios that demand rapid convergence, such as real-time applications. Moreover, the PFW method is resilient

when working with inexact oracles: it preserves theoretical convergence guarantees even if the linear subproblems are solved only approximately. This property is especially important in settings with limited computational resources, where exact solutions are too costly to obtain.

6 Projected Gradient

The Projected Gradient Method (PG) is a fundamental tool in convex optimization, particularly effective for solving constrained problems. The well-known Gradient Method cannot be applied to constrained optimization problems, in fact, its iterates might not belong to the feasible set. Therefore, PG method at each iteration, performs a gradient descent update followed by a projection onto the feasible region, guaranteeing that all iterates satisfy the imposed constraints.

6.1 Algorithm Description

The PG approach is especially useful when the search space is confined to a feasible region $C \subseteq \mathbb{R}^n$.

Its objective is to minimize a differentiable function $f(x)$ over the convex set C. To achieve this, the algorithm repeatedly takes a gradient descent step and then projects the result back onto C, ensuring that the constraints are satisfied. The iterative update rule is given by:

$$x^{(k+1)} = \rho_C(x^{(k)} - \gamma_k \nabla f(x^{(k)}))$$

where ρ_C denotes the projection operator onto C and γ_k is the step-size. The projection guarantees that each new iterate $x^{(k+1)}$ remains within the feasible region, which is essential to maintain the validity of the solution under the constraints of the problem.

Algorithm 5 Projected Gradient Algorithm.

1: **Input**: Convex function f, compact convex domain C.
2: **Initialize**: $x^{(0)} \in C$.
3: **for** $i = 1$ to $\ldots$ **do**
4: Compute the gradient descent step:

$$y^{(k+1)} = x^{(k)} - \gamma_k \nabla f(x^{(k)})$$

5: Project onto the feasible set:

$$x^{(k+1)} = \rho_C(y^{(k+1)})$$

6: Check the stopping criteria.
7: **end for**

6.2 Projected Gradient Convergence Analysis

The Projected Gradient (PG) method has a strong theoretical foundation ensuring its convergence. For convex objective functions, its convergence rate is comparable to standard gradient descent, with the rate primarily determined by assumptions such as strong convexity and Lipschitz continuity of the gradient.

Specifically, for a function that is L-smooth and μ-strongly convex, the PG method achieves a linear rate of convergence. This means that the distance from the current iterate to the optimal solution decreases at a geometric rate with each iteration. This can be expressed formally by the following inequality:

$$\|x^{(k+1)} - x^*\| \leq \rho \|x^{(k)} - x^*\|$$

where x^* is the optimal solution and $\rho \in (0, 1)$ is the convergence factor.

Observe that ρ is determined by the properties of the function.

7 Data

The different algorithms have been tested on four different real world portfolios, downloaded from Real-world datasets for portfolio selection and solutions of some stochastic dominance portfolio models - ScienceDirect.. The datasets contain weekly return values of the FTSE100, Dow Jones, NASDAQ100 and SP500 indexes, adjusted for dividends and for stock splits, which are cleaned from errors as much as possible.

8 Results

Across the four datasets, the results reported in **Table 1** (FTSE100), **Table 2** (NASDAQ100), **Table 3** (DOW JONES), and **Table 4** (SP500) highlight consistent trends in terms of efficiency and accuracy among the algorithms, which are further confirmed by the convergence plots (**Graphs 1A-4B**). In all cases, the final objective values achieved by the different methods are very close (e.g., around – 0.005240 for FTSE100 and –0.007470 for NASDAQ100), indicating that all algorithms converge to nearly identical solutions. However, the final duality gap values show that variants like Pairwise and Away-Step FW achieve comparable or slightly better accuracy with fewer iterations. For example, in **Table 3**, Pairwise FW and Away-Step FW converge in only 40 and 39 iterations, respectively, compared to 205 iterations for standard FW with line search, while maintaining a final objective around – 0.004183. Similarly, in **Table 2**, Away-Step FW converges in just 44 iterations versus 402 for standard FW, with almost identical accuracy (final gap $\approx 1e-5$). Projected Gradient consistently requires the maximum number of iterations (1000) in **Table 1** and **Table 3** and exhibits relatively larger duality gaps (e.g., 2.54e–4 in **Table 3** and 3.4e–5 in **Table 1**), indicating slower convergence. Runtime analysis further emphasizes the advantage of Pairwise and Away-Step FW, which often achieve convergence in less

than 0.01 seconds (**Table 1** and **Table 2**), whereas Projected Gradient can take over 1 second (1.89s in **Table 4**). The Frank-Wolfe variant with diminishing steps also displays improved efficiency over the classical line search approach, as shown in **Table 1** where it requires only 185 iterations and 0.0058 seconds compared to 397 iterations and 0.518 seconds for FW with line search. Overall, while all methods reach comparable final objectives, Pairwise and Away-Step FW consistently demonstrate superior computational efficiency, combining low iteration counts and runtimes without compromising accuracy. Regarding solution sparsity, the advanced FW variants (Pairwise and AwayStep) tend to produce portfolios with smaller support size (e.g., 4 assets on FTSE100 in **Table 1**, 5 assets on NASDAQ100 in **Table 2**), compared to the standard FW which retains $5 - 6$ assets. These differences are visible in the portfolio allocations of **Graphs 1C-4C**, where sparse but well-diversified solutions emerge. The convergence plots (**Graphs 1A-4A** and **Graphs 1B-4B**) further emphasizes the superior speed of Pairwise and Away-Step Frank-Wolfe algorithms, which rapidly reduce both duality gap and objective function, while FW with diminishing steps converges much more slowly. Overall, these findings suggest that the Pairwise and Away-Step variants consistently outperform the other methods in both computational efficiency and solution sparsity, while retaining objective values comparable to the best-performing methods (e.g. -0.007494 for both Pairwise and Away-Step FW on SP500 in **Table 4**).

References

[1] I. M. Bromze, F. Rinaldi, and D. Zeffiro. Frank–Wolfe and friends: A journey into projection-free first-order optimization methods. *Optimization Letters*, 2021.

[2] S. Lacoste-Julien and M. Jaggi. On the global linear convergence of Frank-Wolfe optimization variants. *Advances in Neural Information Processing Systems (NeurIPS)*, 2015.

[3] M. Jaggi. Revisiting Frank-Wolfe: Projection-free sparse convex optimization. *Proceedings of the 30th International Conference on Machine Learning*, 2013.

[4] R. Bruni, F. Cesarone, A. Scozzari, and F. Tardella. Real-world datasets for portfolio selection and solutions of some stochastic dominance portfolio models. *Omega*, 2015.

[5] M. Frank and P. Wolfe. An algorithm for quadratic programming. *Naval Research Logistics Quarterly*, 3(1-2):95–110, 1956.

[6] H. Markowitz. Portfolio Selection. *The Journal of Finance*, 7(1):77–91, 1952.

9 Appendix

In the following tables are displayed the observed metrics of the performances of five different tested algorithms on the four real world financial indexes. In each table, for each algorithm we report:

- the final objective function value;
- the final duality gap value;
- the support size of the solution;
- the CPU runtime required to converge (in seconds);
- and the number of iterations required to converge (note that 1000 is the maximum number of iterations set for the experiments)

With **(ls)** we indicate that the line-search method has been implemented, while **(dim)** refers to the diminishing stepsize strategy.

9.1 Tables of Results

Table 1: Algorithms' Performance Results on FTSE100 dataset

	Final Objective	**Final Gap**	**Support Size**	**Runtime (s)**	**Iterations**
Frank-Wolfe (ls)	-0.005234	0.000012	5	0.513618	397
Pairwise FW (ls)	-0.005240	0.000026	4	0.007890	45
Away-Step FW (ls)	-0.005240	0.000012	4	0.007021	43
Projected Gradient (ls)	-0.005240	0.000012	4	0.701789	1000
Frank-Wolfe (dim)	-0.005240	0.000015	4	0.006097	185

Table 2: Algorithms' Performance Results on NASDAQ100 dataset

	Final Objective	**Final Gap**	**Support Size**	**Runtime (s)**	**Iterations**
Frank-Wolfe (ls)	-0.007463	0.000017	6	0.135628	402
Pairwise FW (ls)	-0.007470	0.000016	5	0.008014	46
Away-Step FW (ls)	-0.007470	0.000010	5	0.007194	44
Projected Gradient (ls)	-0.007470	0.000010	5	0.492818	682
Frank-Wolfe (dim)	-0.007470	0.000037	5	0.007914	238

Table 3: Algorithms' Performance Results on DOW JONES dataset

Algorithm	Final Objective	Final Gap	Support Size	Runtime (s)	Iterations
Frank-Wolfe (ls)	-0.004179	0.000011	5	0.060055	205
Pairwise FW (ls)	-0.004183	0.000011	4	0.005978	40
Away-Step FW (ls)	-0.004183	0.000017	4	0.005692	39
Projected Gradient (ls)	-0.004174	0.000024	4	0.665030	1000
Frank-Wolfe (dim)	-0.004183	0.000072	4	0.002165	73

Table 4: Algorithms' Performance Results on SP500 dataset

	Final Objective	Final Gap	Support Size	Runtime (s)	Iterations
Frank-Wolfe (ls)	-0.007487	0.000016	6	0.525722	402
Pairwise FW (ls)	-0.007494	0.000017	5	0.027570	46
Away-Step FW (ls)	-0.007494	0.000059	5	0.024642	43
Projected Gradient (ls)	-0.007494	0.000010	5	1.770175	686
Frank-Wolfe (dim)	-0.007494	0.000043	5	0.060711	233

9.2 Graphical Representation

In the following graphs, for each one of the four different datasets we show:

1. A plot of the duality gap for the tested algorithms, where the y-axis is on a logarithmic scale and represents the value of the duality gap g_k, while on the x-axis we find the iteration number.

2. A convergence plot of the objective function. The function's value is reported on the y-axis, while the iteration number is on the x-axis.

3. The obtained composition of the optimal portfolio.

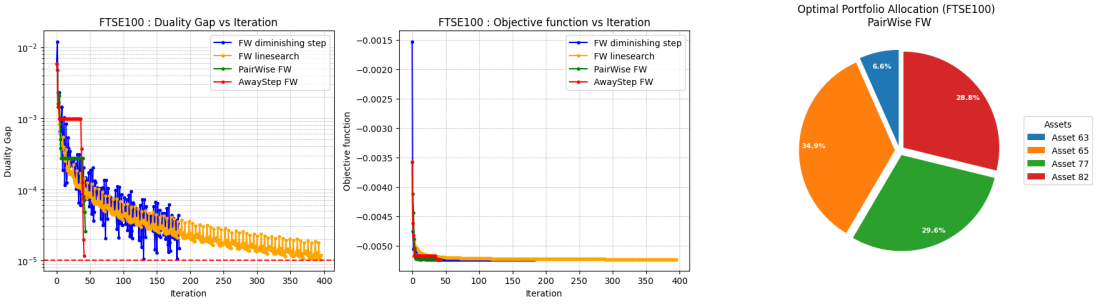

Figure 1: **Graphs 1A, 1B, 1C**:Algorithms' Convergence Analysis on FTSE100 dataset

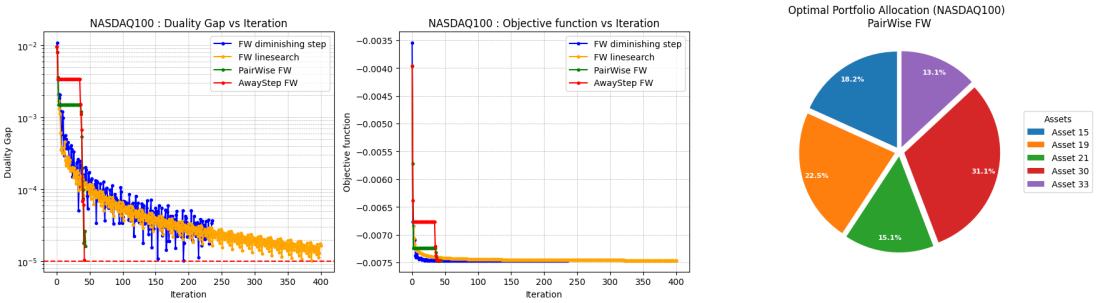

Figure 2: **Graphs 2A, 2B, 2C**: Algorithms' Convergence Analysis on NASDAQ100 dataset.

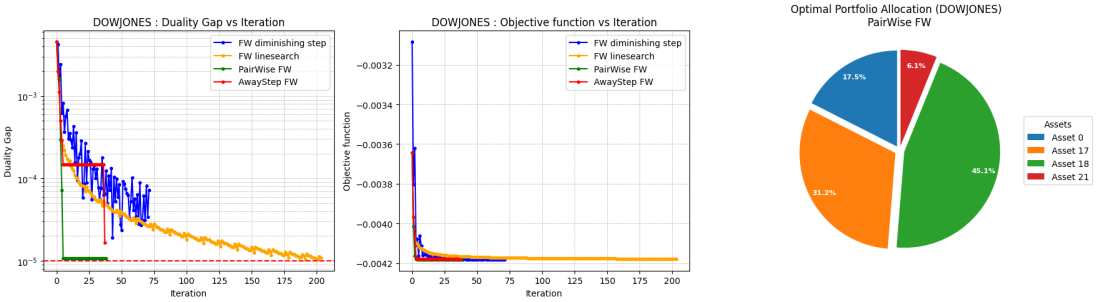

Figure 3: **Graphs 3A, 3B, 3C**: Algorithms' Convergence Analysis on DOW JONES dataset.

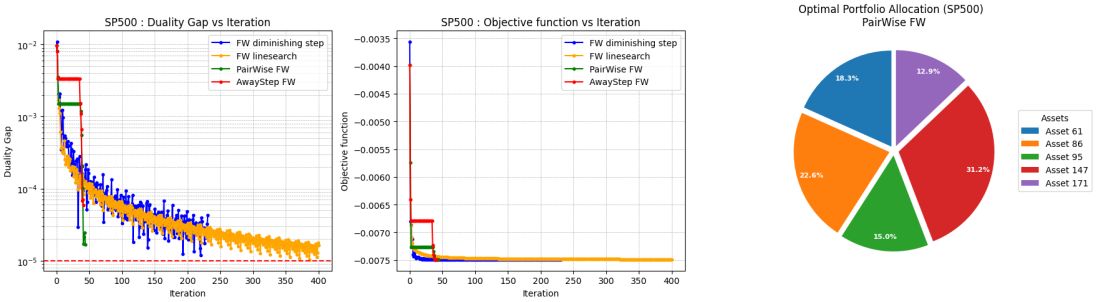

Figure 4: **Graphs 4A, 4B, 4C**: Algorithms' Convergence Analysis on SP500 dataset.