An Overview of Projection-Free Optimization Methods and Numerical Experiments

Giacomo Filippin
giacomo.filippin@studenti.unipd.it

January 30, 2025

Abstract

This paper explores two projection-free optimization methods: the vanilla Frank-Wolfe and the Pairwise Frank-Wolfe. After studying three key papers on these algorithms, we implement and evaluate them on the LASSO problem using a simulated dataset and three famous real-world datasets: Wine quality dataset, AMES housing dataset and YearPredictionMSD dataset. We then analyze the results, investigating the practical effectiveness of these methods in different scenarios.

Contents

1 Introduction

In data science, efficient optimization of large-scale problems is essential. Traditional optimization methods often rely on projection steps, which can be computationally heavy, especially in high-dimensional spaces. The Frank-Wolfe algorithm offers a projection-free approach to convex optimization. This makes it particularly useful for problems where projections are costly or difficult to compute.

This study expands on the Frank-Wolfe algorithm and one of its variants, the pairwise Frank-Wolfe, as they're presented in three key papers:

- *Revisiting Frank-Wolfe: Projection-Free Sparse Convex Optimization*
- *On the Global Linear Convergence of Frank-Wolfe Optimization Variants*
- *Frank–Wolfe and Friends: A Journey into Projection-Free First-Order Optimization Methods*

After understanding the fundamental concepts and results discussed in these papers, with particular attention to how the Frank-Wolfe algorithm works and why it is effective, we develop and implement the vanilla Frank-Wolfe and pairwise Frank-Wolfe algorithms, using techniques explained in the papers (es: line search over step sizes) to improve their convergence capabilities. We then apply these algorithms to the LASSO problem and test them in a selection of real-world scenarios, in order to evaluate the performance of the algorithms by analyzing the efficiency of the solutions.

2 Background

Convex optimization aims to minimize a convex function over a convex set. A function $f : \mathbb{R}^n \to \mathbb{R}$ is called convex if, for all $x, y \in \mathbb{R}^n$ and any $\theta \in [0, 1]$,

$$f\big(\theta x + (1-\theta)y\big) \;\le\; \theta\, f(x) \;+\; (1-\theta)\, f(y).$$

A set $\mathcal{D} \subset \mathbb{R}^n$ is convex if, whenever $x, y \in \mathcal{D}$, every convex combination $\theta x + (1-\theta)y$ (for $\theta \in [0, 1]$) stays inside $\mathcal{D}$.

Projection-free methods are algorithms that avoid directly projecting onto $\mathcal{D}$. This is helpful in high-dimensional problems or when $\mathcal{D}$ has a complicated shape, because projections can be expensive to compute.

The Frank-Wolfe algorithm (or conditional gradient method) is a classic example of a projection-free approach. At each step, it solves a simpler linear problem (the *linear minimization oracle*, or LMO) to find:

$$s_t = \arg\min_{s \in \mathcal{D}} \nabla f(x_t)^\top s.$$

Then it moves from the current point x_t toward s_t:

$$x_{t+1} = x_t + \gamma_t \big(s_t - x_t\big),$$

where $\gamma_t \in [0, 1]$ is a step size found by a line search or a predefined sequence.

Because the LMO can be easier to solve than a projection, Frank-Wolfe and similar methods can improve efficiency, especially for large-scale problems that benefit from sparse or structured solutions.

The Pairwise Frank-Wolfe algorithm is a variant that can speed up convergence by adding "away" steps. Along with moving towards s_t, it can also move away from an existing point v_t in the current solution, helping the algorithm adjust its position more flexibly inside $\mathcal{D}$.

3 Overview of Related Work

3.1 Revisiting Frank-Wolfe: Projection-Free Sparse Convex Optimization

This paper introduces a general framework for understanding Frank-Wolfe-type methods in sparse convex optimization. The authors focus on the algorithm's ability to avoid explicit projection steps by relying on a linear minimization oracle (LMO) to find a feasible direction at each iteration. This makes Frank-Wolfe particularly efficient for problems where projections are computationally expensive, such as those involving large-scale or structured data.

An important contribution of the paper is the use of the duality gap as both a convergence metric and a stopping criterion. The duality gap provides a practical way to measure how close the current solution is to optimality without requiring additional computationally expensive operations.

The authors demonstrate their framework's application to matrix factorization tasks like matrix completion, where Frank-Wolfe can achieve significant computational savings by avoiding projections. Overall, the paper highlights the efficiency and adaptability of Frank-Wolfe methods in handling large-scale optimization problems with structured constraints.

3.2 On the Global Linear Convergence of Frank-Wolfe Optimization Variants

This paper studies different versions of the Frank-Wolfe algorithm, including Away-Steps FW (AFW), Pairwise FW (PFW), and Fully Corrective FW (FCFW). The authors show that these variants can achieve global linear convergence under certain conditions, such as when the objective function is strongly convex. One key result is that these methods remain efficient even when the optimal solution lies on the boundary of the feasible region, which is often challenging for the standard Frank-Wolfe algorithm.

The paper introduces the concept of pyramidal width, a geometric property of the feasible region, which, along with the curvature of the objective function, explains how quickly the algorithms can converge. The authors also demonstrate that these guarantees hold even when the optimal solution is not unique or when it is located at the edge of the feasible set.

This work highlights how the variants improve on the standard Frank-Wolfe algorithm. For example, AFW reduces unnecessary points in the active set, while PFW adjusts weights between selected points to make progress more efficiently. These improvements make the variants particularly useful since they maintain sparsity while reducing computational costs.

3.3 Frank–Wolfe and Friends: A Journey into Projection-Free First-Order Optimization Methods

This paper reviews the Frank-Wolfe algorithm and how it has developed since it was introduced in 1956 for solving convex quadratic problems. It explains how the method avoids expensive projection steps by using simpler linear minimization subproblems instead, which makes it more efficient for large-scale tasks like LASSO and matrix completion.

This paper as well covers key variants of the algorithm, enforcing the conclusions on their improved convergence speed, especially when the problem is strongly convex. The authors also show how these methods are applied in real-world problems, including submodular optimization, matrix factorization, and machine learning tasks like adversarial attacks.

4 Methodology

In this section, we move from the overview and analysis of Frank-Wolfe optimization methods to their practical implementation and evaluation. While the previous sections explored the mathematical foundation and convergence properties of Frank-Wolfe algorithms, this one focuses on their practical application. Through numerical experiments, our aim is to evaluate their efficiency and validate the theoretical expectations seen in the papers.

4.1 Numerical Experiment Setup

The experiments are run on a personal computer with an Intel Core i7 processor, 32 GB of RAM, and no GPU acceleration. The algorithms are written in Python 3.10, using libraries like NumPy, Pandas, SciPy, and sklearn.

The code is split into two files to keep it organized and easy to follow:

1. **functions.py**: This file contains all the functions for processing data, running the optimization, and evaluating the results.

2. **script.py**: This is the main script where the functions are imported, and the experiments are performed.

This structure makes the code simpler to read, with a logical separation between the functions and the experiments, hence why it was chosen.

4.2 Algorithms Implementation

This section provides an overview of the implementation of both the vanilla Frank-Wolfe variant and the pairwise Frank-Wolfe variant.

Specifically, we apply them to the LASSO problem in its constrained formulation, as described in Section 3.3 of *Frank–Wolfe and Friends: A Journey into Projection-Free First-Order Optimization Methods*.

4.2.1 Frank-Wolfe Algorithm for LASSO

We want to solve the LASSO problem

$$\min_{x \in \mathcal{D}} \frac{1}{2} \|Ax - y\|^2,$$

where $\mathcal{D}$ is the set of vectors that satisfy the ℓ_1 constraint through a regularization parameter λ. Below is a basic outline of the Frank-Wolfe steps (as seen in the provided code):

1. **Gradient Computation:** Calculate the gradient of the objective at the current point x_{k-1}:

$$g_k = A^\top (Ax_{k-1} - y).$$

2. **Frank-Wolfe Oracle:** Use a function (`fwOracle`) that looks at g_k and picks the coordinate i with the largest absolute value. It then sets

$$s_k[i] = -\lambda \, \text{sign}(g_k[i]),$$

and all other entries of s_k to zero. This step chooses the direction that should change the most.

3. **Direction and Line Search:** Compute the direction along which we move with our step:

$$d_k = s_k - x_{k-1}.$$

Then, find the best step size ρ_k (clamped between 0 and 1) by minimizing the objective in the direction d_k. In our implementation, this is done by:

$$\rho_k = \max\big(0, \min\big(1, -\frac{(Ax_{k-1} - y)^\top (A\, d_k)}{(A\, d_k)^\top (A\, d_k)}\big)\big).$$

4. **Update x_k:** Move along d_k using the chosen step size:

$$x_k = (1 - \rho_k)\, x_{k-1} \; + \; \rho_k\, s_k.$$

5. **Duality Gap Check:** Calculate the duality gap:

$$g_{\text{dual}} = \langle g_k, x_{k-1} - s_k \rangle.$$

   If $\left| g_{\text{dual}} \right|$ is small enough (below ϵ), we stop because the solution is likely good.

6. **Track the Objective Value:** Finally, we record the LASSO objective

$$f_{\text{LASSO}}(x_k) = \frac{1}{2}\|Ax_k - y\|^2$$

   to keep an eye on how the algorithm progresses.

This loop repeats until the duality gap is under ϵ or until the maximum number of iterations K is reached.

4.2.2 Pairwise Frank-Wolfe Algorithm for LASSO

The Pairwise Frank-Wolfe algorithm builds on the standard Frank-Wolfe approach by allowing "away" steps, which can remove parts of the current solution if we are on the boundary of the ℓ_1 ball. In the code, we apply it to the LASSO problem

$$\min_{x \in \mathcal{D}} \frac{1}{2}\|Ax - y\|^2,$$

where $\mathcal{D} = \{x : \|x\|_1 \leq \lambda\}$. Below is an outline of the main steps:

1. **Compute the Gradient:** We compute the gradient of the objective at x_{k-1}:

$$g_k = A^\top (Ax_{k-1} - y).$$

2. **Pairwise Frank-Wolfe Oracle:** A function (`pairwise_fwOracle`) returns two points:

$$s_k, \quad v_k.$$

   - s_k is found by picking the coordinate of the gradient with the largest absolute value and setting that coordinate to $-\lambda \operatorname{sign}(g_k)$. - v_k (the "away" point) is chosen by looking for the coordinate of x_{k-1} that best reduces the objective if we move away from it.

3. **Direction:** We define the pairwise direction:

$$d_k = s_k \; - \; v_k.$$

   This allows the algorithm to add a new coordinate (via s_k) and possibly remove another (via v_k) at the same time.

4. **Duality Gap Check:** We compute the duality gap as

$$g_{\text{dual}} = -\, g_k^\top d_k,$$

and stop if its absolute value is under ϵ.

5. **Line Search:** Next, we find the step size $\rho_k \in [0, 1]$ by a quadratic approximation of the objective. If simply moving by ρ_k would exceed the ℓ_1 bound ($\|x\|_1 \leq \lambda$), the code uses bisection to shrink the step until feasibility is restored.

6. **Update x_k:** Once we have ρ_k,

$$x_k = x_{k-1} + \rho_k\, d_k.$$

7. **Record the Objective:** We track

$$f_{\text{LASSO}}(x_k) = \frac{1}{2}\|Ax_k - y\|^2$$

to see how the algorithm progresses.

8. **Repeat:** If the duality gap is above ϵ and we have not reached the maximum number of iterations K, we continue.

Thanks to these away steps and the pairwise direction, this method can converge faster when $\|x\|_1$ is near λ. It also keeps the solution sparse, which is helpful in high-dimensional settings.

4.2.3 Comparison, Fairness and Key Optimizations

In our implementation, both the standard and pairwise Frank-Wolfe algorithms start from the same initial solution $\mathbf{x}_0 = \mathbf{0}$, use the same tolerance ε for the duality gap, and share the same maximum number of iterations. Moreover, they rely on the same gradient function (`gradient`), objective function (`fLASSO`), and design matrix $\mathbf{A}$. This ensures that any performance differences arise from how each method updates $\mathbf{x}_k$, rather than from inconsistent parameters.

One key difference is in the *direction-finding step*.

- **Standard Frank-Wolfe:** It calls `fwOracle` to identify a single direction $\mathbf{s}_k$, then computes $\mathbf{d}_k = \mathbf{s}_k - \mathbf{x}_{k-1}$, and applies a line search. The chosen step size is clamped to $[0, 1]$, but it does not need extra checks to remain in the ℓ_1-ball.

- **Pairwise Frank-Wolfe:** It calls `pairwise_fwOracle`, which returns two points $\mathbf{s}_k$ and $\mathbf{v}_k$. The direction is then $\mathbf{d}_k = \mathbf{s}_k - \mathbf{v}_k$. Because this step can move both toward a new coordinate and away from a previously chosen coordinate, we must ensure that after updating $\mathbf{x}_k$, we still satisfy $\|\mathbf{x}_k\|_1 \leq \lambda$. For this reason, a bisection step is performed if the naive line search produces an $\mathbf{x}_k$ that violates the ℓ_1-constraint.

By using the same stopping condition (the duality gap) and the same initial setup, we make the comparison as fair as possible. The main optimization that both codes employ is a *line search* at every iteration. In particular, each algorithm computes a step size based on $\mathbf{r}_k = \mathbf{A}\mathbf{x}_{k-1} - \mathbf{y}$ and $\mathbf{v}_k = \mathbf{A}\mathbf{d}_k$. This approach typically converges faster than fixing a step size in advance. Moreover,

by efficiently identifying the coordinate with the largest (or smallest) gradient entry in `fwOracle` and `pairwise_fwOracle`, we keep overhead low and preserve sparsity. Consequently, any difference in number of operations or iteration needed reflects the inherent advantage (or cost) of handling away steps in the pairwise version, rather than inconsistencies in other parts of the code.

4.3 Datasets and Parameters

To evaluate the performance of the Frank-Wolfe and pairwise Frank-Wolfe algorithms, we selected three datasets: a simulated dataset and three real-world datasets. The simulated one allows controlled experimentation in a high-dimensional, sparse, and predictable environment. The real-world ones instead provide insight into how these algorithms perform on practical regression problems with varying levels of complexity and noise. This combination offers a balanced and comprehensive approach to testing the implementation, as well as the strengths and weaknesses of projection-free optimization methods in different situations.

For all datasets, we designed and implemented a pre-processing pipeline. This pipeline ensures that each dataset is handled appropriately, including necessary steps such as normalization, encoding of categorical variables, handling of missing data and train-test split to maximize the applicability and efficiency of the Frank-Wolfe and pairwise Frank-Wolfe algorithms. This preparation allows us to use the datasets effectively in testing the algorithms' performance in both synthetic and real-world contexts.

4.3.1 LARS Simulated Dataset

For the simulated dataset, we used the LARS (Least Angle Regression) model to generate data with the following parameters: 4,000 samples, 200 features, 20 informative features, and a noise level of 0.5. The primary goal of using this simulated dataset is to test the performance of Frank-Wolfe algorithms in a controlled high-dimensional environment where the sparsity structure and the noise level are clearly defined.

This allows us to create scenarios that are particularly suited for sparse optimization tasks, such as the one targeted by the Frank-Wolfe algorithm. Additionally, by tuning parameters such as noise, we can observe how robust the algorithms are under different conditions.

However, there are also drawbacks to relying on simulated data. Although it provides a clean testing ground for algorithmic performance, it may not fully represent the complexities of real-world data, such as nonlinear relationships and unstructured noise. The artificial nature of the data set might lead to overestimating the efficiency of the algorithms when applied to practical problems. Therefore, it must be complemented with real-world datasets to get a more holistic view of their applicability.

4.3.2 Wine Quality Dataset

The Wine Quality dataset contains 4,898 samples and 11 features, representing physicochemical properties of wine, such as acidity, sugar content, and alcohol, along with a target variable indicating the quality rating of the wine. This data set presents a real-world regression problem in which the goal is to predict wine quality based on measurable attributes. Unlike the controlled structure of the simulated data set, the Wine Quality data set includes inherent noise and variability typical of

real-world data, making it a valuable benchmark for the robustness of the Frank-Wolfe algorithms in practice.

The moderate size of this dataset allows us to assess how well the algorithms handle regression tasks where feature relationships are less structured and may be influenced by factors such as measurement noise.

However, one drawback of this dataset is that it may not be sparse or high-dimensional enough to fully exploit the strengths of projection-free methods like Frank-Wolfe, which are particularly effective in dealing with sparse optimization problems. Additionally, the relatively small feature set may limit the scalability insights we gain from testing on larger and more complex datasets.

It is also important to note that this data set has an ordered categorical target and that we are performing a linear regression as if it were a continuous quantitative one. Although performing linear regression on an ordered categorical target is not ideal, our primary focus is on evaluating the convergence capabilities of the Frank-Wolfe algorithms rather than optimizing predictive performance, so we are purposely doing a bad regression choice, given that the focus is on the optimization algorithm and not on the model's performance.

4.3.3 AMES Housing Dataset

The AMES Housing dataset consists of 2,930 samples and 79 features, representing various attributes of residential homes in Ames, Iowa, including numerical and categorical variables such as lot size, neighborhood, and overall quality. The target variable in this dataset is the sale price of the homes, making it a typical high-dimensional regression problem. This dataset is well-suited for testing the scalability and flexibility of the Frank-Wolfe and Pairwise Frank-Wolfe algorithms, especially when dealing with a larger number of features.

The high dimensionality and diversity of features, including both numerical and categorical data, provide a real-world challenge for optimization algorithms. The AMES Housing dataset allows us to observe how well the algorithms can manage a more complex feature space and handle the mix of continuous and discrete variables. Furthermore, the larger size of the dataset makes it a good benchmark for testing the scalability of the Frank-Wolf algorithm.

As for the Wine Quality dataset, However, the lack of sparsity in the features may not fully align with the strengths of Frank-Wolfe methods, which are designed to excel in sparse optimization tasks. Additionally, this dataset needs a heavier preprocessing pipeline, which adds additional complexity when applying these algorithms.

4.3.4 YearPredictionMSD Dataset

The YearPredictionMSD dataset consists of 515,345 samples and 90 features, representing various audio attributes of songs, such as tempo, loudness, and timbre. The target variable is the release year of the song, making it a large-scale regression problem. This dataset is well-suited for testing the scalability and efficiency of the Frank-Wolfe and Pairwise Frank-Wolfe algorithms, especially in high-dimensional settings with a substantial number of samples and features.

The high dimensionality and extensive size of the YearPredictionMSD dataset provide a robust benchmark for evaluating how effectively the algorithms manage computational resources and converge in large-scale optimization tasks. However, similar to the AMES Housing dataset, the lack of inherent sparsity in the features may not fully leverage the strengths of Frank-Wolfe methods.

5 Results

5.1 Performance Metrics

We measure the quality and efficiency of the two optimization methods using four metrics:

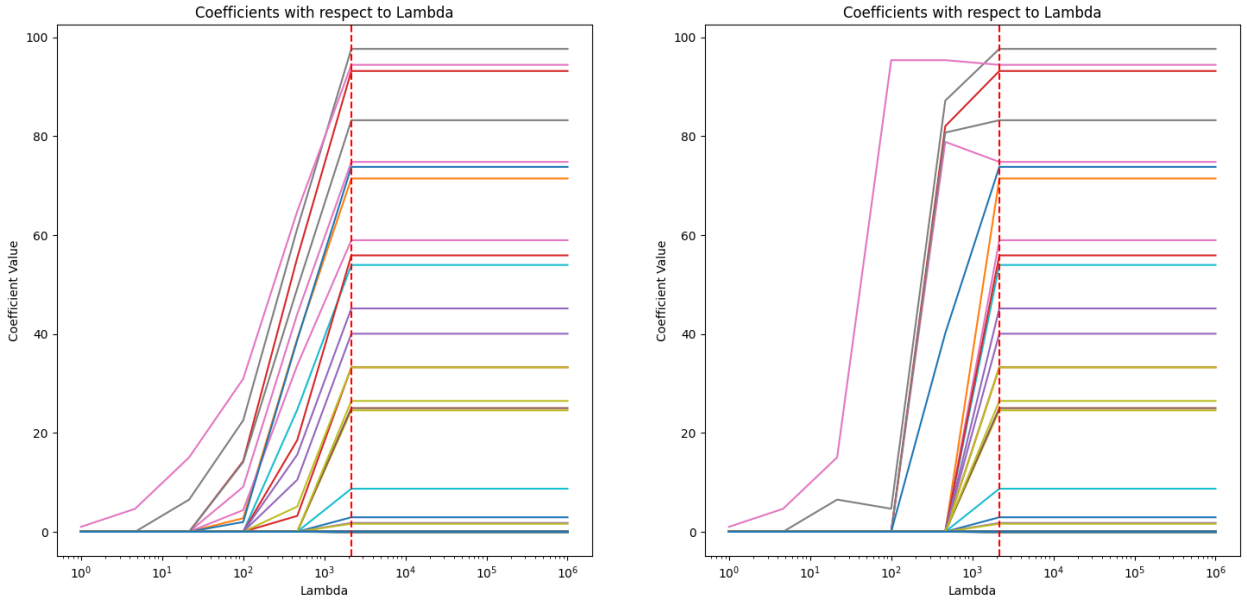
- **Duality Gap**: Indicates how close the current solution is to the theoretical optimum. A smaller gap means better convergence. Since duality gap in this implementation is used as a stopping criterion, we will not use it directly to compare algorithms, but rather to ensure a fair comparison between them.
- **Number of Iterations**: While still useful to know the number of iterations needed for convergence, it would be incorrect to compare algorithms based only on this number, since different algorithms could require less, but heavier iterations. This would result in an unfair comparison.
- **Number of Operations**: This metric reflects the computational cost of the algorithm's convergence. Fewer operations to reach a given accuracy imply a faster, more efficient algorithm. The number of operations is more informative than the number of iterations, making this is the main metric we're going to use to compare the algorithms. Regardless of the hardware used and the number of iterations, less operations equals more efficient convergence.
- **MSE (Mean Squared Error) and R squared**: These metrics evaluate the predictive accuracy of the final model. Although the main focus is on comparing the optimization procedures, this metrics are monitored to confirm that the solutions found are reasonable and aligned. It is also important to note that these metrics are not to be looked at alone, but should be compared to a "baseline" standard optimization algorithm in order to verify that the limits of the regressions performed are not caused by the optimization algorithms, but rather by the environment and by other steps of the modelling phase. The reason for the choice of these metrics is that while MSE depends on the scale of the target, and thus is not comparable across different datasets, the R squared gives an idea of how much information the model was able to discern in a more "absolute" way.

5.2 Experimental Results

In this section, we present the results of our experiments. For each dataset, and for both standard and pairwise variants, we first perform a grid search over lambda to identify the optimal regularization parameter. Then, we evaluate the resulting models using the metrics described in the "Performance Metrics" section. The goal is to better understand if and when the pairwise Frank-Wolf shows better convergence capabilities than the standard Frank-Wolf in real-life scenarios.

5.2.1 LARS Dataset

The main reason to use a simulated dataset like LARS is to have an idea of how much the efficiency can ideally change between variants. By fine-tuning dimensionality, sparsity of the solution and noise in the observations, we can then see how the difference between variants change for real-word dataset with respect to this initial comparison. The parameters chosen to generate the dataset are 4,000 samples, 200 features, 20 informative features, and a noise level of 0.5.

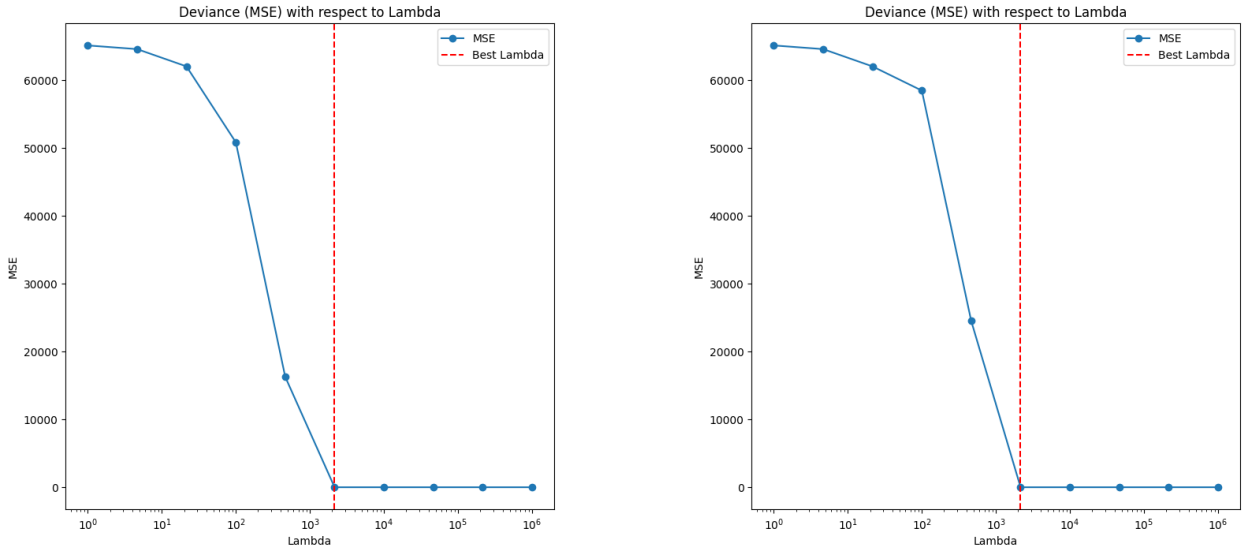


(a) standard Frank-Wolfe

(b) pairwise Frank-Wolfe

Figure 1: Grid Search results: Coefficients for different regularization parameters for LARS dataset

As shown in Figure 1 and 2, the grid-search for the best regularization parameter gives the same result (2154.435) for both algorithms. Fixing this hyperparameter, the two variants converge to a nearly identical solution with a mean squared error of 0.266 and an adjusted R squared of 0.999 for both variants.



(a) standard Frank-Wolfe

(b) pairwise Frank-Wolfe

Figure 2: Grid Search results: MSE for different regularization parameters for LARS dataset

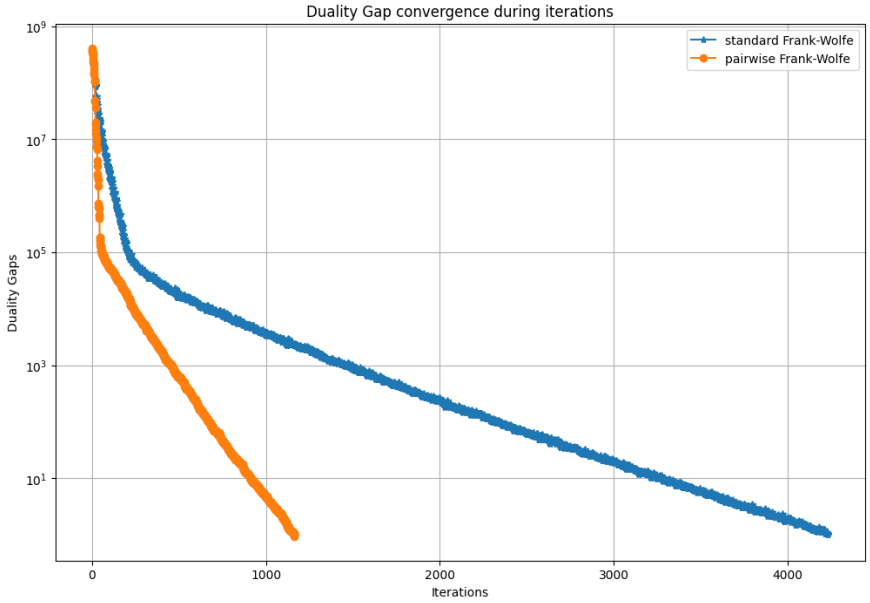
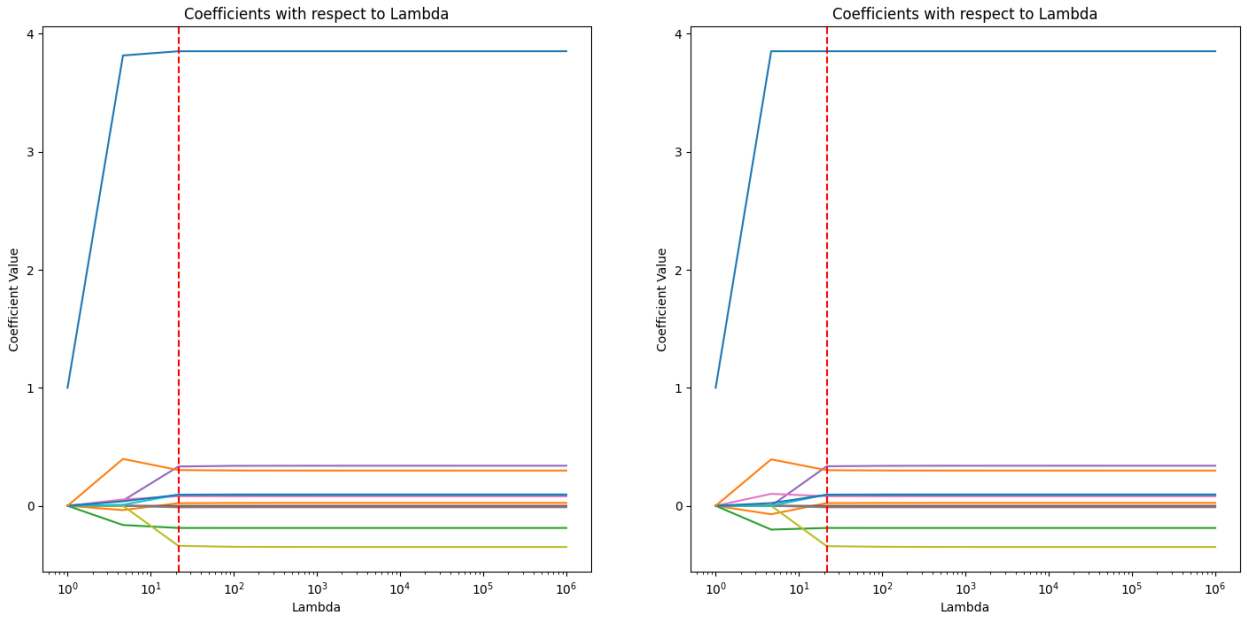


Figure 3: duality gaps for standard and pairwise Frank-Wolfe and fixed best lambda (2154.435) for LARS dataset

What is relevant, however, is the convergence of the duality gap between iterations for the two variants. As shown in Figure 3, the Pairwise variant takes 1156 iterations against the 4251 needed by the vanilla Frank-Wolfe. This results in a substantial reduction in the number of operations needed to reach convergence, with ”only” 464712000 operations needed instead of 1306902000 (around 0.64% less). Having now seen the improvement in convergence that the pairwise variant can bring on a simulated dataset, we verify this increased efficiency on our real-world datasets.

5.2.2 Wine Quality Dataset

Given the small and simple nature of the Wine Quality dataset, we intuitively expect a smaller impact of the pairwise variant on the efficiency of the solution compared to the simulated dataset and larger and more complex real-world ones.

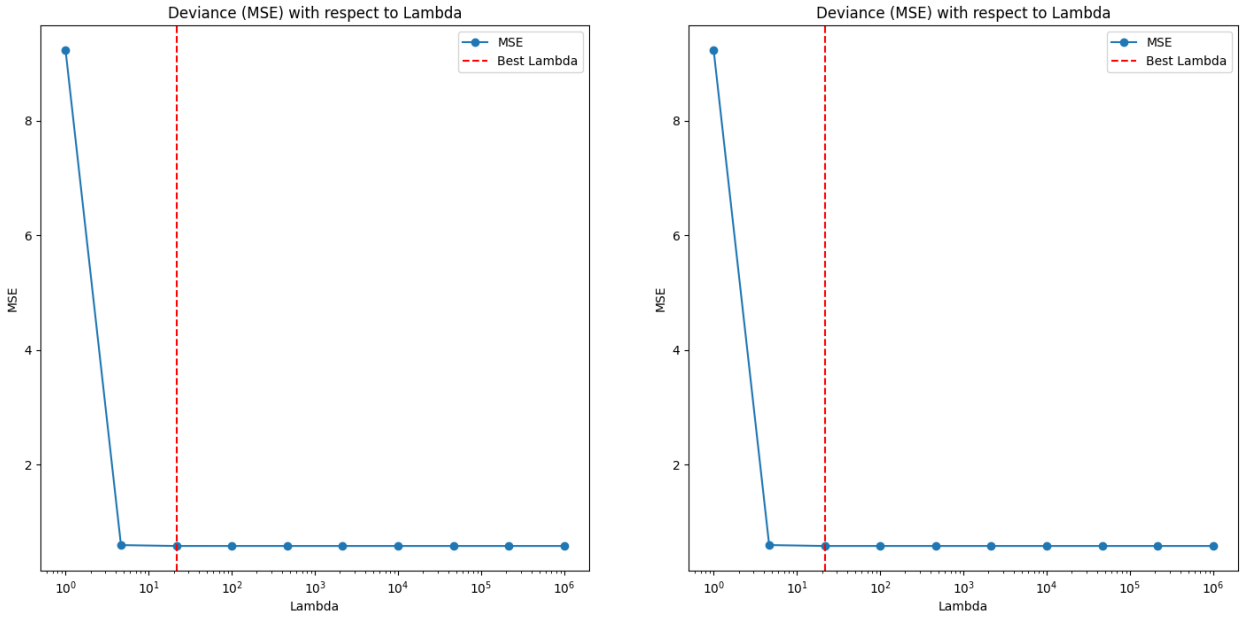


(a) standard Frank-Wolfe

(b) pairwise Frank-Wolfe

Figure 4: Coefficients for different regularization parameters for WINE dataset

The grid-search finds the lower value of 21.544 as the best regularization parameter for both vanilla and pairwise variants. As shown in Figures 4 and 5, for both variants, the effect of increasing the regularization value appears very similar on both coefficients and mean squared errors. The solution found when fixing the best regularization hyperparameter is approximately the same, with mean squared error 0.575 and adjusted R squared 0.26. It is important to take into account, for the low value of the adj. R squared, that we are performing a linear regression on a categorical output. Again, this is not relevant for our comparison as long as the low adj. R squared is the same for both solutions.



(a) standard Frank-Wolfe (b) pairwise Frank-Wolfe

Figure 5: MSE for different regularization parameters for WINE dataset

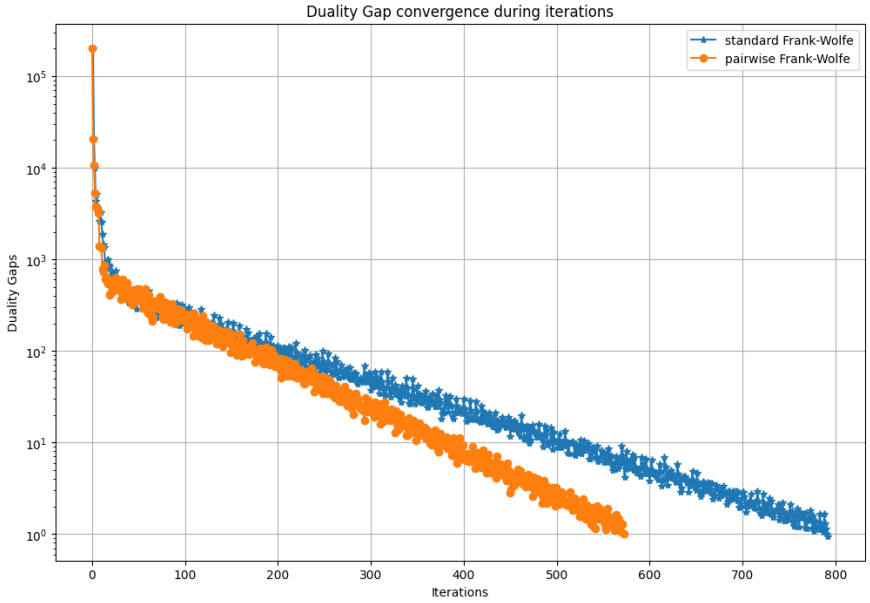
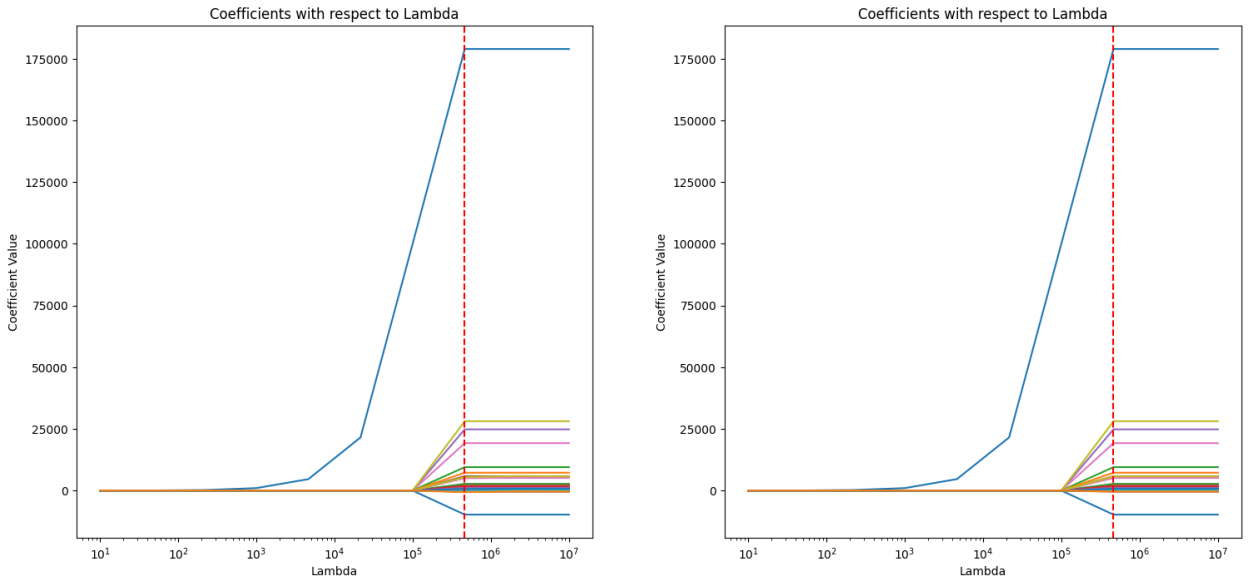


Figure 6: duality gaps for standard and pairwise Frank-Wolfe and fixed best lambda (21.544) for WINE dataset

Figure 6 confirms what we expected from this dataset: convergence is easier and thus faster than before, and this reduces the efficiency gain of the pairwise variant. The iterations needed for convergence are 573 for the pairwise against 792 of the vanilla Frank-Wolfe, resulting in 16846200 operations instead of 23284800 (”just” 27.7% less).

5.2.3 AMES Housing Dataset

This is a high-dimensional problem. The increased dimension of both observations and features allows us to verify how the two variants scale to larger and more complex problems. Theoretically, we would expect the efficiency advantage of the pairwise variant to increase with respect to the simpler WINE regression problem.

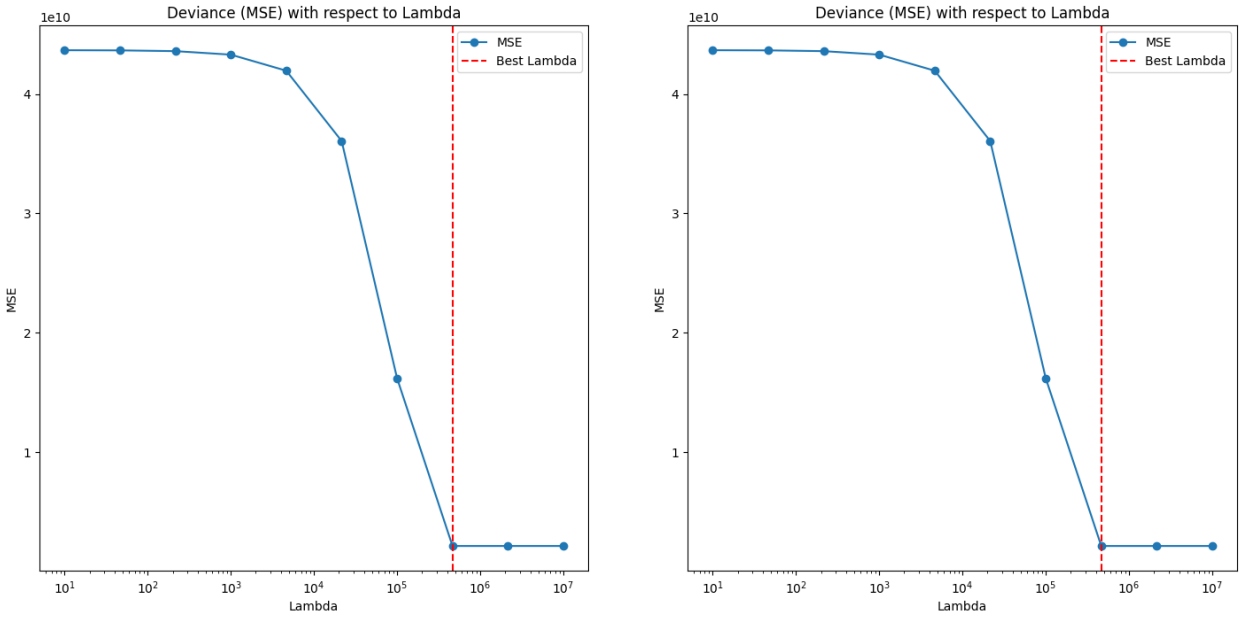


(a) standard Frank-Wolfe

(b) pairwise Frank-Wolfe

Figure 7: Coefficients for different regularization parameters

Likely due to the bigger scale of the problem, the grid-search highlights a higher value of 464158.883 as the best regularization parameter for both variants. The search results, in Figures 7 and 8, confirm once again similar regularization effects for both variants. The solution found with the best regularization parameter is once again approximately the same, with mean squared error 2114677529 and adjusted R squared 0.736. Note that it is important to take into account the scale of the target feature when looking at the MSE. The target is not scaled ad has a way bigger mean and variance with respect to previous ones, thus giving a bigger MSE. This is confirmed by the adjusted R squared, which is around .5 higher than the one for the WINE dataset, even if the MSE is 2114677529 instead of 0.575. Regression quality (and consequently the adj. R squared) has also increased due to the dataset being more suited for a linear regression thanks to its continuous quantitative target.



(a) standard Frank-Wolfe (b) pairwise Frank-Wolfe

Figure 8: MSE for different regularization parameters

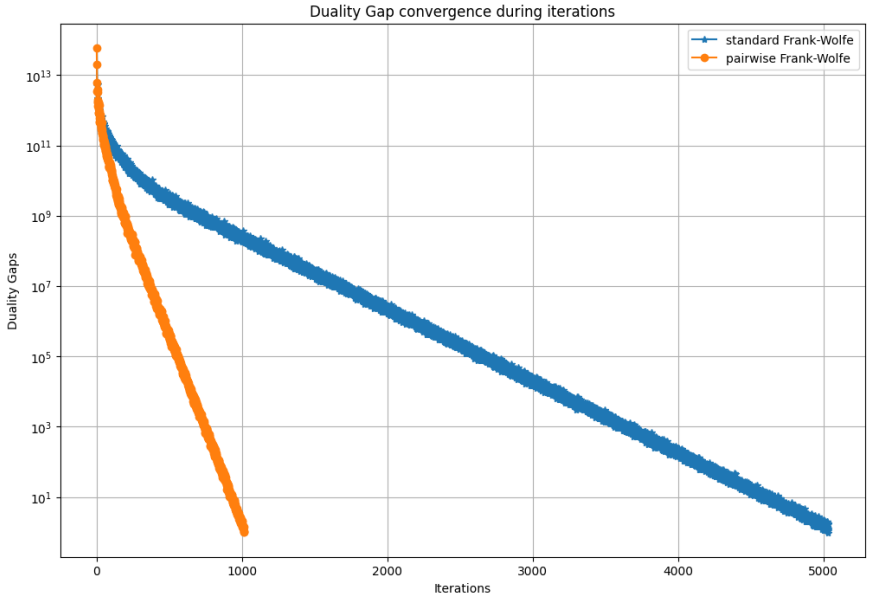
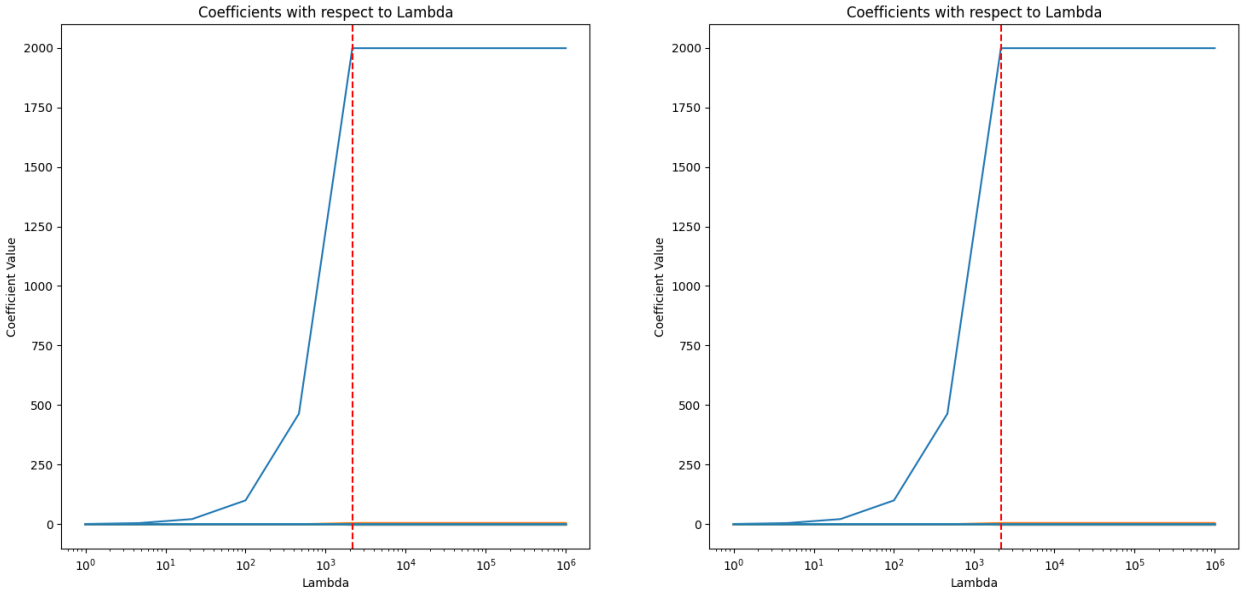


Figure 9: duality gaps for standard and pairwise Frank-Wolfe and fixed best lambda (464158.883)

Figure 9 confirms what we expected from this dataset: convergence is harder than before. This increases the efficiency gain of the pairwise variant with respect to the vanilla one. The iterations needed for convergence are 1014 for the pairwise against 5033 of the vanilla Frank-Wolfe, resulting in 16284840 operations instead of 80829980 (around 80% less). This is the biggest efficiency improvement so far, beating the simulated LARS dataset as well, which showed a reduction of around 64%.

5.2.4 Year Prediction MSD Dataset

For our last real-world problem, we try our variants on a significantly larger and more complex dataset. Intuitively, we'd expect the pairwise variant to show the biggest efficiency improvement, given the trend that seems to correlate the difference between variants in number of operations needed with the complexity and the dimensionality of the problem.

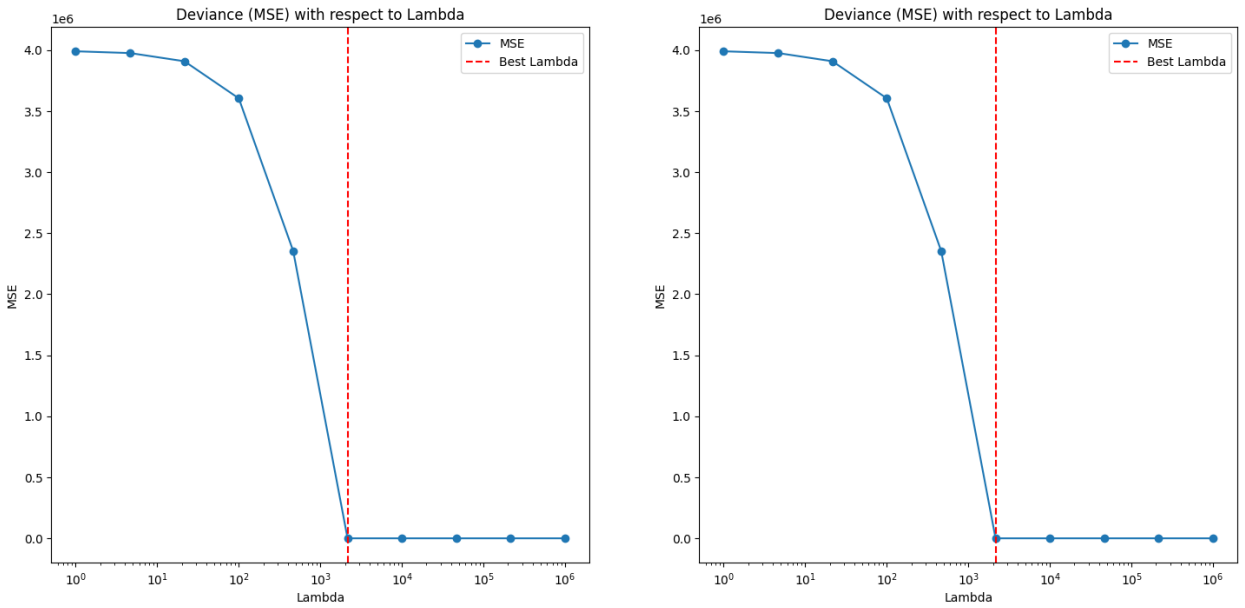


(a) standard Frank-Wolfe

(b) pairwise Frank-Wolfe

Figure 10: Coefficients for different regularization parameters

The grid-search shows 2154.435 as the best regularization parameter for both vanilla and pairwise variants. Once again, Figures 10 and 11 shows, for both variants, a similar effect in increasing the regularization value. The diffence between best solutions when fixing the best regularization parameter is still negligible, and shows a pretty bad performance, with mean squared error 90.866 and adjusted R squared 0.234, likely due to the complexity of the dataset and the lack of a precise and specific pre-processing phase, like the one we did for the AMES dataset.



(a) standard Frank-Wolfe (b) pairwise Frank-Wolfe

Figure 11: MSE for different regularization parameters

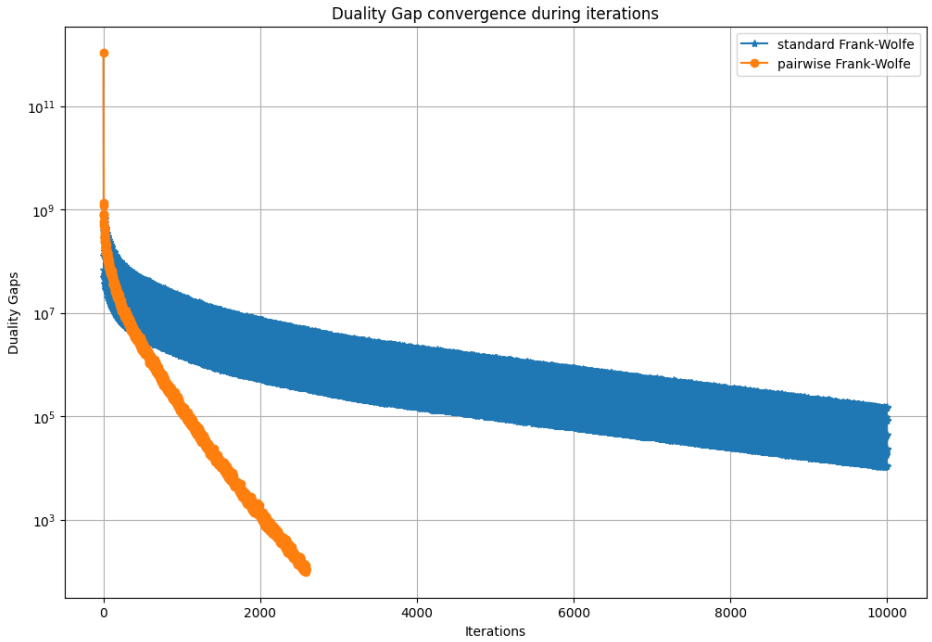


Figure 12: duality gaps for standard and pairwise Frank-Wolfe and fixed best lambda (2154.435)

Figure 12 confirms once again our considerations for this dataset: convergence is significantly harder. This increases even more the efficiency gain of the pairwise variant with respect to the vanilla one. The iterations needed for convergence are 2586, while the vanilla gets stuck and does not reach the tolerance target even after 9999 iterations. Given that the vanilla variant does not

reach the desired target, while the pairwise does, a comparison of the number of iterations would not be fair. Still, this result confirmes once again, and even more strongly, what the other problems had already shown: the gain in efficiency of the pairwise variant with respect to the vanilla one increases with complexity, allowing for a significant reduction in the needed number of iterations, and consequently operations.

6 Conclusion

This essay set out to examine the Frank-Wolfe algorithm and its pairwise variant. Our goal was to test whether these projection-free methods effectively address the LASSO problem in both simulated and real-world settings, as well as to verify the improvements of the pairwise variant w.r.t the vanilla Frank-Wolfe algorithm.

The results show that the pairwise Frank-Wolfe algorithm consistently converges faster than the vanilla version, particularly on larger or more complex datasets like AMES Housing and Year Prediction MSD. Despite their different update steps, both variants reached nearly identical solutions in terms of mean squared error and R^2, confirming that faster convergence does not come at the cost of poorer final performance. These observations agree with the theoretical findings in the three papers, which highlight that cutting down on projection steps can reduce the computational burden without undermining the quality of the solution, and that a "smarter" direction step can consistently reduce the number of operations needed for convergence.

Overall, our experiments reinforce the value of projection-free approaches for large-scale optimization. By matching the solution quality of vanilla Frank-Wolfe while reducing iteration and operation counts, the pairwise variant proves especially suitable for high-dimensional data. Hence, the conclusions drawn here affirm the practical benefits proposed in the reviewed papers, highlighting the Frank-Wolfe family of methods as powerful tools in modern data science, particularly for problems where standard projection steps are prohibitively expensive.

References

- *Revisiting Frank-Wolfe: Projection-Free Sparse Convex Optimization:*
  Martin Jaggi.
- *On the Global Linear Convergence of Frank-Wolfe Optimization Variants:*
  Simon Lacoste-Julien, Martin Jaggi.
- *Frank–Wolfe and Friends: A Journey into Projection-Free First-Order Optimization Methods:*
  Immanuel M. Bomze1, Francesco Rinaldi, Damiano Zeffiro.
- *Complexity of Linear Minimization and Projection on Some Sets:*
  Cyrille W. Combettes, Sebastian Pokutta.