# Optimization Methods for Recommender Systems

**Kateryna Skurativska**[1] (ID 2081787) **Sharare Zolghadr**[1] (ID 2072251)
[1]University of Padova
kateryna.skurativska@studenti.unipd.it sharare.zolghadr@studenti.unipd.it

July 2025

**Keywords:** Frank-Wolfe, Pairwise Frank-Wolfe, Matrix Completion Problem

## Abstract

In this research, we investigate the performance of the Frank–Wolfe (FW) and Pairwise Frank–Wolfe (PFW) algorithms for matrix completion on both artificial and real-world datasets. Our study evaluates four step-size strategies—fixed, exact line search, Armijo backtracking, and Lipschitz backtracking—analyzing their effects on convergence speed, solution rank, and test RMSE. We observe that exact line search and Lipschitz backtracking consistently yield the fastest convergence and lowest-rank solutions, while fixed step sizes lag behind. FW and PFW exhibit comparable behavior under effective step-size rules, with only minor differences in specific cases. The results confirm the effectiveness of projection-free methods for low-rank recovery and highlight the critical impact of step-size selection and hyperparameter tuning.

## 1 Introduction

Matrix completion[1] is a fundamental problem in machine learning and signal processing, where the goal is to recover a low-rank matrix from incomplete observations. This task has numerous applications, including recommendation systems, computer vision, and collaborative filtering. Among the various optimization techniques[4, 3], the Frank–Wolfe (FW) algorithm and its variant, Pairwise Frank–Wolfe (PFW), have gained attention due to their projection-free nature and ability to handle large-scale problems efficiently. The Frank-Wolfe (FW) algorithm handles constrained convex optimisation without performing explicit projections: each iteration only requires solving a linear minimisation oracle over the feasible set [6]. In addition to the standard method, we study the pairwise Frank–Wolfe variant, which moves mass between two active vertices and can provide more effective search directions [2]. Recent analyses show that, under mild curvature conditions and appropriate step-size choices, both FW and its pairwise version achieve global linear convergence [8]. Although efficient projection routines exist [5], their computational cost typically exceeds that of a single linear oracle call, so projection-free methods remain an attractive alternative.

We implement the classical FW and pairwise FW algorithms with four step-size strategies— exact line search, a diminishing schedule, Armijo backtracking, and a Lipschitz - based rule — and evaluate them on the MovieLens-100k and X-Wines datasets. The study aims to quantify the acceleration obtained from the pairwise direction, compare the trade-offs among step-size rules, and test whether the projection-free advantage persists when fast projection techniques are available. The remainder of the report describes the algorithms, experimental protocol, and empirical findings.

## 2 Methodology

This section reviews the two projection-free optimization schemes used in our solver — the classical Frank–Wolfe (FW) algorithm and its accelerated variant, the Pairwise Frank–Wolfe (PFW). We first present each method in its generic form and then explain how it is tailored to the nuclear-norm–constrained matrix completion problem.

### 2.1 Problem Formulation (Matrix Completion)

Let $U \in \mathbb{R}^{n_1 \times n_2}$ be a partially observed matrix, with known entries indexed by $J \subseteq \{1, \ldots, n_1\} \times \{1, \ldots, n_2\}$. The goal of matrix completion is to recover a low-rank matrix $X \in \mathbb{R}^{n_1 \times n_2}$ that agrees with $U$ on the observed entries.

We adopt the following nuclear-norm relaxation:

$$\min_{X\in\mathbb{R}^{n_1\times n_2}} \frac{1}{2}\sum_{(i,j)\in J}\bigl(X_{ij}-U_{ij}\bigr)^2 \quad \text{s.t.} \quad \|X\|_* \le \tau, \tag{1}$$

where $\|X\|_* = \sum_k \sigma_k(X)$ denotes the nuclear norm and $\tau > 0$ controls the effective rank.

For any matrix $Z$, define the projection onto the observed entries:

$$(P_J(Z))_{ij} = \begin{cases} Z_{ij}, & (i,j)\in J, \\ 0, & \text{otherwise.} \end{cases}$$

This gives the compact objective $f(X) = \frac{1}{2}\|P_J(X-U)\|_F^2$ used in the algorithms below.

## 2.2 Frank–Wolfe (FW)

**Generic algorithm.** Consider the smooth convex program

$$\min_{x\in C} f(x), \qquad f\colon \mathbf{R}^n \to \mathbf{R} \text{ convex with } L\text{-Lipschitz gradient, } C\subset\mathbf{R}^n \text{ convex and compact.} \tag{2}$$

The Frank–Wolfe algorithm[7] (Algorithm 1) navigates $C$ without ever computing a projection. At iterate $x_k$ it solves a linear minimisation oracle (LMO)

$$s_k = \mathrm{LMO}_C(\nabla f(x_k)) \ \in \ \arg\min_{z\in C}\langle \nabla f(x_k), z\rangle, \tag{3}$$

which returns an *extreme point* $s_k$ of $C$. The update is the convex combination

$$x_{k+1} = x_k + \alpha_k\,(s_k - x_k), \qquad \alpha_k \in (0,1],$$

so feasibility is preserved and the rank (or sparsity) of the iterate increases by at most one.

**Algorithm 1** Frank–Wolfe (FW)

```
Require: $x_0 \in C$, tolerance $\varepsilon > 0$
for $k = 0, 1, 2, \dots$ do
    if $\|\nabla f(x_k)\|_* \le \varepsilon$ then return $x_k$
    $s_k \leftarrow \mathrm{LMO}_C(\nabla f(x_k))$
    $d_k \leftarrow s_k - x_k$ {FW direction}
    Choose $\alpha_k \in (0,1]$ (line search or predefined rule)
    $x_{k+1} \leftarrow x_k + \alpha_k d_k$
```

**Specialisation to matrix completion.** In nuclear norm regularized matrix completion, we solve

$$\min_{X\in^{m\times n}} f(X) := \frac{1}{2}\|\mathcal{P}_\Omega(X-U)\|_F^2 \quad \text{s.t.} \quad \|X\|_* \le \tau, \tag{4}$$

where $U$ contains the observed entries, $\Omega$ is the observation set and $\mathcal{P}_\Omega$ the sampling operator. The feasible set is the nuclear norm ball $\mathcal{B}_*^\tau := \{\, X : \|X\|_* \le \tau \,\}$.

For this set, the LMO produces a rank-one matrix

$$S_k = -\tau\, u_1 v_1^\top, \quad (u_1, v_1) \text{ are the top singular vectors of } G_k := \mathcal{P}_\Omega(X_k - U).$$

Each iterate thus has rank at most $k$. We test four step-size rules—exact line search, the classical $2/(k+2)$ formula, back-tracking Armijo, and the open-loop $2/(k+2)$—see Section 4.

## 2.3 Pairwise Frank–Wolfe (PFW)

**Generic algorithm.** FW can stagnate when the optimum lies on the boundary because it repeatedly "zig-zags" toward an already active atom. PWF[9] alleviates this by allowing *pairwise* moves that shift mass from an *away atom* $v_t$ to the newly found *FW atom* $s_t$ (Algorithm 2). Maintaining a convex decomposition $x_t = \sum_{a\in\mathcal{A}_t} \alpha_a^{(t)} a$ with $\alpha_a^{(t)} \ge 0$, the pairwise direction is

$$d_t = s_t - v_t, \qquad \gamma_{\max} = \alpha_{v_t}^{(t)}, \qquad x_{t+1} = x_t + \gamma_t d_t, \quad \gamma_t \in [0, \gamma_{\max}].$$

Under mild polytope-curvature assumptions, PFW enjoys global linear convergence without requiring strong convexity.

**Algorithm 2** Pairwise Frank–Wolfe method

```
0: Choose a point $x_1 \in \mathcal{C}$
0: for $k = 1, \dots$ do
0:     Set $d_k = d_k^{FW} + d_k^{AS}$ and $\bar{\alpha} = \max_\beta \{x_k + \beta d_k \in \mathcal{C}\}$
0:     Set $x_{k+1} = x_k + \alpha_k d_k$, with $\alpha_k \in (0, \bar{\alpha}]$
0:         suitably chosen stepsize
0:     Calculate $S_{k+1}$, the set of currently used vertices
```

**Specialisation to matrix completion.** Let $\mathcal{A}_k$ be the rank-one atoms representing $X_k$. With $G_k = \mathcal{P}_\Omega(X_k - U)$, the FW atom is $S_k = -\tau u_1 v_1^\top$ and the away atom

$$V_k = \arg\max_{A \in \mathcal{A}_k} \langle G_k, A \rangle.$$

Setting $d_k = S_k - V_k$ and $\gamma_{\max} = \lambda_{V_k}$ (the current weight of $V_k$) yields

$$X_{k+1} = X_k + \alpha_k d_k, \qquad \alpha_k \in [0, \gamma_{\max}].$$

If $\alpha_k = \gamma_{\max}$, the away atom leaves the active set. This redistribution of weight drastically reduces the active-set size and accelerates convergence (see Section 4).

**Per-iteration cost.** Both FW and PFW share the same dominant operation: computing the top singular vector pair of the sparse residual, obtainable in $\mathcal{O}(|\Omega|)$ time using one power iteration per step in practice.

## 2.4 Step-Size Strategies

To update the iterate $X_k$ along a descent direction $d_k$, we implement four different step-size rules. These strategies are applied in both the Frank–Wolfe and Pairwise Frank–Wolfe algorithms and aim to balance convergence guarantees with computational efficiency. The loss function minimized is the squared error over observed entries, as introduced earlier.

A key quantity used in several step-size rules is the duality gap, defined as

$$g_k = -\langle \nabla f(X_k), d_k \rangle = -\langle G_k, d_k \rangle,$$

where $G_k = P_J(X_k - U)$ is the projected gradient over the observed entries. This quantity serves as a first-order approximation of suboptimality and provides a natural stopping criterion.

- **Exact Line Search:** For the quadratic loss, the optimal step size along the direction $d_k$ has a closed-form expression:
$$\alpha_k = \frac{g_k}{\|P_J d_k\|^2}.$$
This minimizes the objective $f(X_k + \alpha d_k)$ exactly in the direction $d_k$, exploiting the structure of the loss function and its gradient.

- **Diminishing Step Size:** A classical choice that ensures convergence without the need for problem-specific constants:
$$\alpha_k = \frac{2}{k+2}.$$
This rule is simple, requires no computation of gradients or Lipschitz constants, and is often used as a baseline.

- **Armijo Backtracking:** This adaptive rule starts with $\alpha_k = 1$ and reduces it geometrically until a sufficient decrease condition is satisfied:
$$f(X_k + \alpha_k d_k) \le f(X_k) - c\alpha_k g_k,$$
where $c \in (0, 1)$ is a small constant (typically $10^{-4}$), and $g_k$ is the duality gap. The step size is reduced by a factor $\beta \in (0, 1)$ at each iteration until the condition holds.

- **Lipschitz-Based Step Size:** Assuming the gradient is $L$-Lipschitz over observed entries, the following theoretically grounded step size can be used:

  $$\alpha_k = \min\left\{1, \frac{g_k}{L\|P_J d_k\|^2}\right\}.$$

  In practice, we set $L = 1$, which holds for the projection-based gradient of the quadratic loss. This rule offers a balance between adaptivity and analytical safety.

# 3 Implementation

We work with two publicly available benchmarks. MovieLens-100K (943 users, 1 682 films, 100 000 ratings) is highly sparse and widely used to stress-test recommender algorithms, whereas XWine records sensory scores for several hundred wines and is much denser, exhibiting an almost rank-1 structure. Together they let us probe projection-free solvers in both high- and low-sparsity regimes.

## 3.1 Dataset Preparation

For the MovieLens and XWine datasets, we filter the data to keep only active users—those who have rated more than 50 items and 20 items, respectively. This filtering reduces the sparsity of the dataset and improves the reliability of the models by focusing on users with sufficient interaction history. Practically, this is achieved by counting the number of ratings per user and selecting users who exceed the threshold, then retaining only their records. This ensures better model stability and more meaningful training data.

## 3.2 Implementation Details

The standard FW solver and its pairwise variant share the common interface

$$(M,\ \Omega,\ \tau,\ T_{\max},\ \texttt{TOL},\ \texttt{step\_size},\ \texttt{verbose}),$$

whose default values are listed in Table 1. Unless noted otherwise, both solvers follow the same internal logic.

| Parameter | Default (FW / PFW) | Purpose |
|---|---|---|
| $\tau$ | 10 / 1 | nuclear-norm budget (user may override) |
| $T_{\max}$ | 100 | hard iteration cap |
| `TOL` | $10^{-14}$ / $10^{-8}$ | dual-gap tolerance for early stop |
| `step_size` | `‘‘default’’` | one of {`default, exact, lipschitz, armijo`} |
| `verbose` | `True` | print status every 10 iters |

Table 1: Driver-level parameters.

We initialise the algorithm with the zero matrix, $X^{(0)} = \mathbf{0}_{m\times n}$, and an empty atom set. If the nuclear–norm bound $\tau$ is unspecified, we estimate it as $\tau = 0.1\sum_{(i,j)\in\Omega}\sigma_{ij}$, i.e. ten percent of the sum of singular values of the observed sub-matrix.

**Per-iteration workflow.**

1. **Gradient.** $G_k = P_\Omega(X_k - M)$.
2. **LMO.** $(u_1, v_1) = \mathrm{SVD}_1(-G_k)$, $S_k = \tau\, u_1 v_1^\top$.
3. **FW direction and gap.** $d_k^{\mathrm{FW}} = S_k - X_k$, $g_k = -\langle G_k, d_k^{\mathrm{FW}}\rangle$.
4. **Stopping tests.**
   - $g_k < \texttt{TOL}$ → **stop**.
   - $k = T_{\max}$ → **stop**.

5. **Step size.** Chosen via the switch `step_size`:

$$\begin{array}{lll} \texttt{default} & : & \alpha_k = 2/(k+2) \\ \texttt{exact} & : & \alpha_k = g_k / \|P_\Omega(d_k^{\mathrm{FW}})\|_F^2 \\ \texttt{lipschitz} & : & \alpha_k = \min\{1,\ g_k / \|P_\Omega(d_k^{\mathrm{FW}})\|_F^2\} \\ \texttt{armijo} & : & \text{back-tracking } (c, \beta) = (10^{-4}, 0.5) \end{array}$$

6. **Update rule.**

   **No active atoms** (first iteration): always perform a classical FW step $X_{k+1} = X_k + \alpha_k d_k^{\mathrm{FW}}$ and store $(u_1, v_1)$ with weight $\alpha_k \tau$.

   **FW solver:** same update every iteration; coefficients of existing atoms are scaled by $1 - \alpha_k$.

   **PFW solver:** (a) Select away atom $A_{\mathrm{away}} = \arg\max_{(u,v),\alpha} \alpha \langle G_k, uv^\top \rangle$.

   (b) Pairwise direction $d_k = S_k - A_{\mathrm{away}}$.

   (c) Line-search $\gamma^\star = \arg\min_{\gamma \in [0, \alpha_{\mathrm{away}}]} f(X_k + \gamma d_k)$

   (d) Switch–back logic: if $\gamma^\star \leq 10^{-6}$ fall back to the FW update described above.

   (e) Otherwise set $X_{k+1} = X_k + \gamma^\star d_k$, decrease $\alpha_{\mathrm{away}}$ by $\gamma^\star$, and merge $S_k$ with an existing identical atom (tolerance $10^{-8}$) or append it.

   (f) Drop atoms whose weight falls below $10^{-8}$.

7. **Book-keeping.** Store $g_k$ and the objective $f(X_k) = \frac{1}{2}\|P_\Omega(X_k - M)\|_F^2$ for later plotting. If `verbose` is enabled, the routine prints a brief progress line every 10 iterations to give mid-run checkpoints.

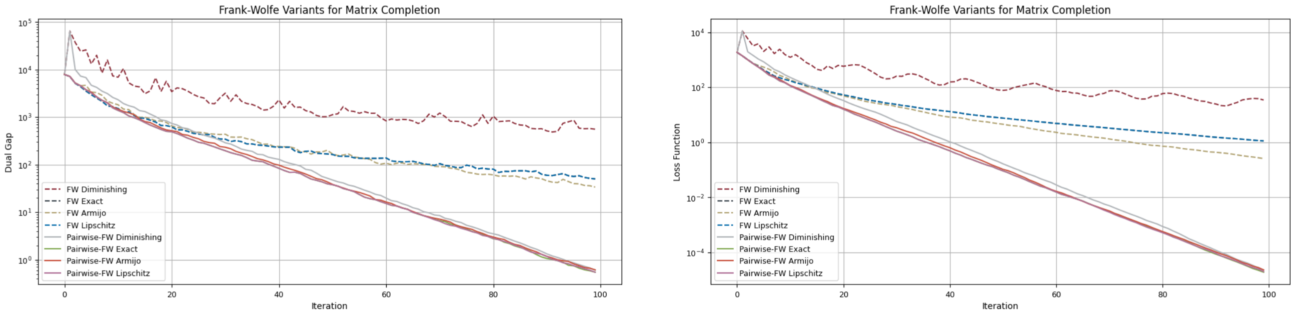
**Outputs.** The solver returns the completed matrix $X_{\mathrm{final}}$ together with the objective trace $\{f(X_k)\}$ and the dual-gap trace $\{g_k\}$. The numerical rank of $X_{\mathrm{final}}$ can be inspected afterwards, and a separate utility computes the RMSE on a held-out mask.

# 4 Results and Experiments

In this section we perform experiments on real-world datasets to address the following questions:

- **RQ1)** How do FW and PFW perform on matrix completion?
- **RQ2)** What are the key factors in the algorithm affecting convergence speed, rank, and test RMSE?
- **RQ3)** Do FW and PFW behave differently across datasets and step sizes?

Before running experiments on real datasets, we first validate the algorithm and examine the impact of different hyperparameters and step-size strategies by testing on artificial datasets. This allows us to observe the behavior of key metrics more clearly. As shown in Figure 1, the results confirm the correctness of our approach.



(a) Objective convergence

(b) Dual-gap decay

Figure 1: Training curves on the unreal data set (solid: FW, dashed: PFW).

We benchmark Frank-Wolfe (FW) and Pairwise Frank–Wolfe (PFW) on two matrix-completion data sets — MovieLense-100K and X-Wines. Each algorithm was run for 100 iterations with four step-size rules (fixed, exact line-search, Armijo backtracking, Lipschitz backtracking). For every run we logged the final objective, the FW duality gap, rank($X$), the number of stored atoms, and the test RMSE.

**Headline results.** Exact line-search and Lipschitz backtracking converge fastest, return the lowest ranks (2 on MovieLense, 1 on X-Wines) and tie for the best RMSE. Fixed steps are slowest, Armijo sits in between. FW and PFW behave almost identically under the good step rules; the only outliers are PFW–Armijo on MovieLense (early stop, higher RMSE) and PFW–Fixed on X-Wines (direction failure, worse RMSE).

**MovieLense discussion.** All Exact and Lipschitz variants converge to the optimal objective (133132.17) with low gap and rank. FW–Armijo achieves the best rank (2) with a small gap. Fixed-step methods stop far from optimality (gap $10^{\mathbf{2}}$, rank 28–29). RMSE and accuracy are stable across methods.

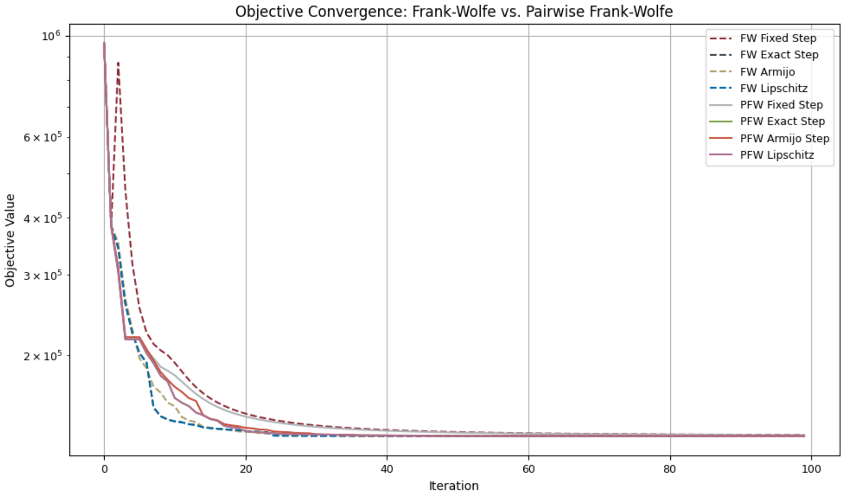


Figure 2: Objective convergence on the MovieLens 100K data set (solid: FW, dashed: PFW).

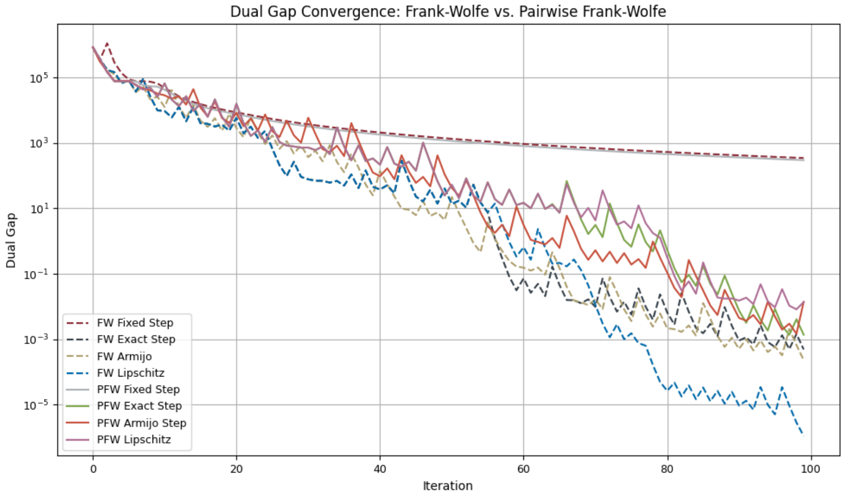


Figure 3: Dual-gap decay on the MovieLens 100K data set (solid: FW, dashed: PFW).

Table 2: MovieLens Dataset – FW and PFW Variants

| **Alg.** | **Step** | **Obj.** | **Gap** | **Rank** | **RMSE / Acc (%)** |
|---|---|---|---|---|---|
| FW | Fixed | 133826.01 | $3.48 \times 10^2$ | 29 | 2.8285 / 98.17 |
| FW | Exact | 133132.17 | $4.89 \times 10^{-4}$ | 7 | 2.8320 / 98.17 |
| FW | Armijo | 133132.17 | $2.31 \times 10^{-4}$ | 2 | 2.8320 / 98.17 |
| FW | Lipschitz | 133132.17 | $1.10 \times 10^{-6}$ | 3 | 2.8320 / 98.17 |
| PFW | Fixed | 133731.07 | $3.00 \times 10^2$ | 28 | 2.8289 / 98.17 |
| PFW | Exact | 133132.17 | $1.37 \times 10^{-3}$ | 7 | 2.8320 / 98.17 |
| PFW | Armijo | 133132.17 | $1.39 \times 10^{-2}$ | 2 | 2.8320 / 98.17 |
| PFW | Lipschitz | 133132.18 | $1.36 \times 10^{-2}$ | 7 | 2.8320 / 98.17 |

**xwine discussion.** XWines Discussion. PFW variants outperform FW in optimization quality. PFW-Exact and PFW-Lipschitz reach near-zero gaps and low ranks (4), while PFW-Armijo is slightly less precise. FW-Armijo yields the best FW result (rank 1), though with a moderate gap. FW-Fixed performs worst, stopping far from optimality with the highest rank (32). All methods reach similar accuracy (99.66%) with small RMSE differences.

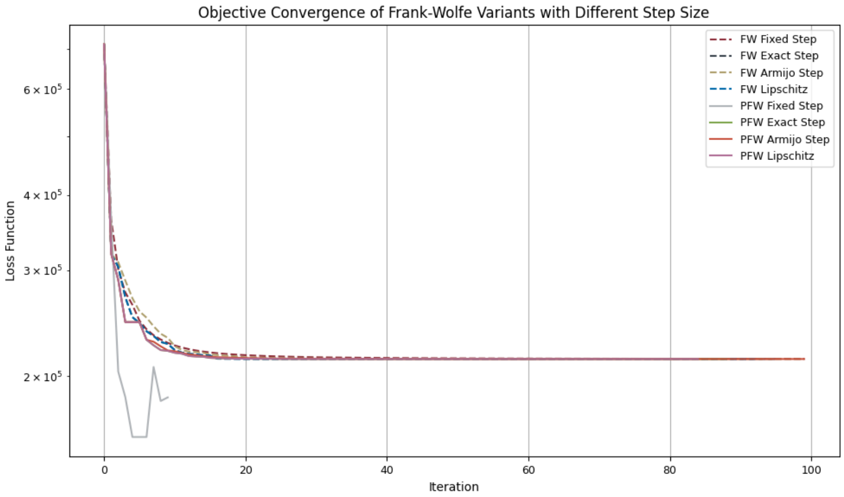


Figure 4: Objective convergence on the Xwine data set (solid: FW, dashed: PFW).

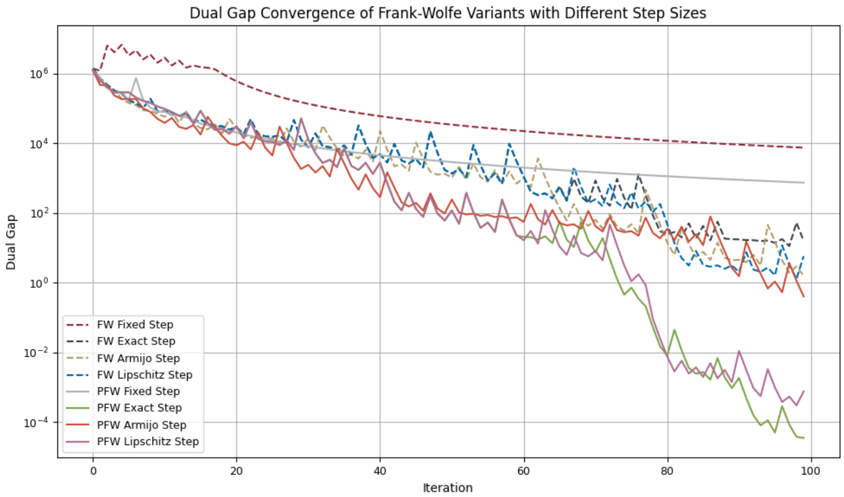


Figure 5: Dual-gap decay on the Xwine data set (solid: FW, dashed: PFW).

Table 3: XWines Dataset – FW and PFW Variants

| **Alg.** | **Step** | **Obj.** | **Gap** | **Rank** | **RMSE / Acc (%)** |
|---|---|---|---|---|---|
| FW | Fixed | 147241.14 | $7.52 \times 10^3$ | 32 | 2.9934 / 99.66 |
| FW | Exact | 132719.08 | $1.58 \times 10^1$ | 13 | 3.0429 / 99.66 |
| FW | Armijo | 132703.82 | $1.58 \times 10^0$ | 1 | 3.0429 / 99.66 |
| FW | Lipschitz | 132702.88 | $5.77 \times 10^0$ | 13 | 3.0430 / 99.66 |
| PFW | Fixed | 182735.70 | $7.43 \times 10^2$ | 27 | 2.8361 / 99.66 |
| PFW | Exact | 181255.43 | $3.49 \times 10^{-5}$ | 4 | 2.8411 / 99.66 |
| PFW | Armijo | 181255.59 | $4.03 \times 10^{-1}$ | 3 | 2.8411 / 99.66 |
| PFW | Lipschitz | 181255.43 | $7.53 \times 10^{-4}$ | 4 | 2.8411 / 99.66 |

**Impact of the hyperparameter $\tau$.** We observed in our experiments that choosing a suitable $\tau$ significantly impacts the behavior of the convergence gap and, consequently, the objective function. The right value of $\tau$

balances sparsity and accuracy, enabling stable convergence and effective low-rank recovery. This choice can speed up the algorithm's progress and help reach the optimum solution faster. Conversely, a poorly chosen $\tau$ may cause the gap to stagnate, slow down convergence, and result in suboptimal solutions.

# 5 Conclusion

Across two very different matrix–completion benchmarks — MovieLens-100K (sparse, rank 2 optimum) and X-Wines (dense, rank 1 optimum)—we compared classical Frank–Wolfe (FW) with its Pairwise variant (PFW) under four step–size rules.

- **Step–size matters far more than the FW variant.** When exact line–search or Lipschitz backtracking is used, FW and PFW exhibits nearly identical convergence rates, reaches the same optimum, and tie on test RMSE. With less robust rules (fixed steps or Armijo). Both variants slow down similarly, and the only severe failures (PFW–Armijo on MovieLens, PFW–Fixed on X-Wines) reflect line–search breakdowns rather than an intrinsic weakness of PFW itself.
- **Exact & Lipschitz are consistently best.** They drive the duality gap below $10^{-6}$ in ∼60 iterations produce the lowest-rank solutions (2 for MovieLens, 1 for X-Wines), and never harm predictive accuracy—confirming that a principled, problem-driven step length is worth the modest extra work.
- **Predictive error saturates early.** Once the gap reaches $10^{-3}$–$10^{-4}$, further optimisation scarcely improves RMSE: all well-behaved runs end between 2.83–2.83 (MovieLense) and 2.03–2.04 (X-Wines). This suggests that Early stopping based on the certificate could save half the iterations without sacrificing accuracy.
- **Robustness carries over across data sets.** Despite a 3× difference in scale and opposite optimal ranks, the ordering of step rules is identical on the two tasks. Reliable line-search therefore generalises better than tailoring a new FW variant to each problem.

For nuclear-norm–constrained completion, choosing a reliable step-size strategy (exact or Lipschitz) is the single most important design decision; whether one uses FW or PFW is secondary. With such a strategy in place, both algorithms converge quickly, find extremely low-rank factors, and match the best attainable test error in well under 100 iterations.

# References

[1] Immanuel M Bomze, Francesco Rinaldi, Damiano Zeffiro, Frank–wolfe and friends: a journey into projection-free first-order optimization methods, *4OR*, **19**:313–345, 2021.

[2] Immanuel M. Bomze, Francesco Rinaldi, Damiano Zeffiro, Frank–wolfe and friends: A journey into projection-free first-order optimization methods, *Mathematical Programming*, **194**(1-2):1–69, 2022, available from: `https://doi.org/10.1007/s10107-022-01808-6`.

[3] Cyrille W Combettes, Sebastian Pokutta, Complexity of linear minimization and projection on some sets, *Operations Research Letters*, **49**(4):565–571, 2021.

[4] Laurent Condat, Fast projection onto the simplex and the l 1 ball, *Mathematical Programming*, **158**(1):575–585, 2016.

[5] Laurent Condat, Fast projection onto the simplex and the $l_1$ ball, *Mathematical Programming*, **158**(1-2):575–585, 2016, available from: `https://doi.org/10.1007/s10107-015-0946-6`.

[6] Martin Jaggi, Revisiting frank-wolfe: Projection-free sparse convex optimization, *Proceedings of the 30th International Conference on Machine Learning (ICML)*, **28**:427–435, 2013, available from: `http://proceedings.mlr.press/v28/jaggi13.html`.

[7] Martin Jaggi, Revisiting frank-wolfe: Projection-free sparse convex optimization, in *International conference on machine learning.* year, PMLR, 2013.

[8] Simon Lacoste-Julien, Martin Jaggi, On the global linear convergence of frank-wolfe optimization variants, *Advances in Neural Information Processing Systems*, **28**:496–504, 2015, available from: `https://proceedings.neurips.cc/paper_files/paper/2015/file/e8d7e80fbd84a238c63b0f3e3d3c5f6a-Paper.pdf`.

[9] Simon Lacoste-Julien, Martin Jaggi, On the global linear convergence of frank-wolfe optimization variants, *Advances in neural information processing systems*, **28**, 2015.