FISTA-PALM: A Hybrid Approach to Speeding Up Alternating Minimisation with Nesterov Acceleration

Candidate Number: 1091424

MSc Special Topic

Optimisation for Datascience, HT 2025

Prof Coralia Cartis

University of Oxford

Mathematical Institute

Contents

Abstract

This paper investigates nonconvex, nonsmooth optimisation problems with block-separable structure, focusing on the Proximal Alternating Linearised Minimisation (PALM) algorithm. We propose an accelerated variant, **FISTA-PALM**, which incorporates extrapolation to enhance convergence speed. We evaluate its performance across a range of sparse matrix factorisation tasks, including both two-block and three-block formulations. Numerical results show that FISTA-PALM consistently outperforms standard PALM and inertial methods, achieving faster and often more accurate solutions, particularly in the early stages of optimisation and under high sparsity.

1 Introduction

Many optimisation problems in machine learning and signal processing are both *nonconvex* and *nonsmooth*, due to the presence of constraints, sparsity terms, or structured regularisers. While convex models offer elegant theoretical guarantees, they often fail to capture the complexity of real-world data. Consequently, solving nonconvex problems has become increasingly important for building models that are both expressive and effective.

A large subclass of these problems admits a *block-separable form*, where the objective naturally splits across groups of variables. A typical structure is the composite formulation:

$$\text{(M)} \qquad \min_{(x,y)\in\mathbb{R}^n\times\mathbb{R}^m} \Psi(x,y) := f(x) + g(y) + H(x,y),$$

where H is smooth with Lipschitz-continuous gradients, and f, g are nonsmooth (possibly nonconvex) functions acting on distinct variable blocks. This decomposition makes such problems amenable to block-coordinate methods.

The *Proximal Alternating Linearised Minimisation (PALM)* algorithm [1] is a widely used framework for such problems. It alternates gradient steps on the smooth term and proximal updates on the nonsmooth terms, applied to each block. PALM is valued for its *simplicity*, and under mild assumptions (e.g., the Kurdyka–Łojasiewicz property), it guarantees convergence to critical points. However, its worst-case convergence rate remains sublinear at $\mathcal{O}(1/k)$, which is often insufficient for large-scale or ill-conditioned problems.

To accelerate convergence, momentum-based extensions have been proposed. In particular, *inertial methods* like iPALM [2] introduce velocity-like terms into the updates, leveraging previous iterates to achieve empirical speedups. Although effective, these methods often require careful parameter tuning, and the inertial extrapolation used is relatively simple.

In contrast, the use of *acceleration*—in the sense of Nesterov's method—is less explored in PALM-style algorithms. Unlike simple inertia, acceleration computes a *predictive point* combining current and past iterates to obtain a more informative location for gradient evaluation. In convex settings, this approach underpins the *Fast Iterative Shrinkage-Thresholding Algorithm (FISTA)* [3], achieving an improved rate of $\mathcal{O}(1/k^2)$.

Our method, **FISTA-PALM**, introduces structured Nesterov-style extrapolation into the PALM framework in a modular way. At each step, we construct a lookahead point

$$\tilde{x}^k = x^k + \frac{t_{k-1} - 1}{t_k}(x^k - x^{k-1})$$

and apply the proximal-gradient update at this extrapolated location:

$$x^{k+1} = \operatorname{prox}_\eta^f \left(\tilde{x}^k - \frac{1}{\eta} \nabla H(\tilde{x}^k, y^k) \right),$$

with analogous updates for each block. This preserves the alternating structure of PALM while enhancing it with structured momentum.

In the remainder of the paper, we formally introduce the FISTA-PALM algorithm and describe how extrapolated updates integrate into block-wise PALM. We then evaluate its performance on several versions of sparse nonnegative matrix factorisation problems [4][5], providing a natural setting to assess the impact of Nesterov-style acceleration in block-coordinate methods. Comparative experiments against PALM and iPALM across different structural and initialisation settings are presented.

2 Problem Formulataion and Definitions

Problem Formulation. We consider the following nonsmooth, nonconvex optimisation problem:

$$\text{(M)} \qquad \min_{(x,y)\in\mathbb{R}^n\times\mathbb{R}^m} \Psi(x,y) := f(x) + g(y) + H(x,y),$$

where:

- $f : \mathbb{R}^n \to (-\infty, +\infty]$ and $g : \mathbb{R}^m \to (-\infty, +\infty]$ are proper, lower semicontinuous functions;
- $H : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$ is a C^1 function.

We now provide key definitions used throughout the algorithm description.

Convexity and Smoothness.

Definition: (Convexity)**.** A function $f : \mathbb{R}^n \to \mathbb{R}$ is *convex* if for all $x, y \in \mathbb{R}^n$ and $\lambda \in [0, 1]$,

$$f(\lambda x + (1 - \lambda)y) \leq \lambda f(x) + (1 - \lambda)f(y).$$

Definition: (Smoothness)**.** A function $f : \mathbb{R}^n \to \mathbb{R}$ is *smooth* if it is continuously differentiable ($f \in C^1(\mathbb{R}^n)$) and its gradient ∇f is Lipschitz continuous. That is, there exists $L > 0$ such that for all $x, y \in \mathbb{R}^n$,

$$\|\nabla f(x) - \nabla f(y)\| \leq L\|x - y\|.$$

Subdifferentials.

Definition: (Subdifferentials)**.** Let $\sigma : \mathbb{R}^d \to (-\infty, +\infty]$ be a proper, lower semicontinuous function.

- The *Fréchet subdifferential* of σ at $x \in \operatorname{dom}(\sigma)$ is

  $$\hat{\partial}\sigma(x) = \left\{ u \in \mathbb{R}^d \;\middle|\; \liminf_{y \to x, y \neq x} \frac{\sigma(y) - \sigma(x) - \langle u, y - x \rangle}{\|y - x\|} \geq 0 \right\}.$$

  If $x \notin \operatorname{dom}(\sigma)$, then $\hat{\partial}\sigma(x) = \emptyset$.

- The *limiting subdifferential* at $x \in \mathbb{R}^d$ is

  $$\partial\sigma(x) = \left\{ u \in \mathbb{R}^d \;\middle|\; \exists x^k \to x,\, \sigma(x^k) \to \sigma(x),\, u^k \in \hat{\partial}\sigma(x^k),\, u^k \to u \right\}.$$

Proximal Map.

Definition: (Proximal Operator)**.** Let $\sigma : \mathbb{R}^d \to (-\infty, +\infty]$ be proper and lower semicontinuous. For $x \in \mathbb{R}^d$ and $t > 0$, the *proximal map* is

$$\operatorname{prox}_t^{\sigma}(x) := \arg\min_{u \in \mathbb{R}^d} \left\{ \sigma(u) + \frac{t}{2}\|u - x\|^2 \right\}.$$

The proximal operator enables tractable updates by separating smooth and nonsmooth terms, crucial for convergence under nonconvexity and nondifferentiability.

Kurdyka–Łojasiewicz Property.

Definition: (Kurdyka–Łojasiewicz (KL) Property)**.** A proper, lower semicontinuous function $\sigma : \mathbb{R}^d \to (-\infty, +\infty]$ satisfies the *KL property* at $\bar{u} \in \operatorname{dom}(\partial\sigma)$ if there exist $\eta > 0$, a neighbourhood U of $\bar{u}$, and a desingularising function $\varphi \in \Phi_\eta$ such that for all

$$u \in U \quad \text{with} \quad \sigma(\bar{u}) < \sigma(u) < \sigma(\bar{u}) + \eta,$$

we have

$$\varphi'\big(\sigma(u) - \sigma(\bar{u})\big) \cdot \operatorname{dist}(0, \partial\sigma(u)) \geq 1.$$

If the KL property holds at all $\bar{u} \in \operatorname{dom}(\partial\sigma)$, then σ is called a *KL function.*

Semi-Algebraic Functions.

Definition: (Semi-Algebraic Function)**.** A function $f : \mathbb{R}^n \to \mathbb{R}$ is *semi-algebraic* if its graph

$$\big\{(x, f(x)) \in \mathbb{R}^{n+1} \,\big|\, x \in \mathbb{R}^n\big\}$$

is a semi-algebraic set, i.e., defined by a finite number of polynomial equalities and inequalities.

3 Background: Proximal Alternating Linearised Minimisation (PALM)

Before introducing our hybrid method, we briefly describe the PALM algorithm [1], which serves as its foundation. PALM—short for *Proximal Alternating Linearised Minimisation*—is a general algorithm for solving structured, nonconvex, and nonsmooth optimisation problems. It is particularly suited to problems where the objective can be split into blocks and contains both smooth and nonsmooth terms.

PALM is designed for problems of the form:

$$\text{(M)} \qquad \min_{(x,y)\in\mathbb{R}^n\times\mathbb{R}^m} \Psi(x, y) := f(x) + g(y) + H(x, y),$$

where:

- f and g are proper, lower semicontinuous functions (possibly nonsmooth or nonconvex),
- H is a smooth coupling function with Lipschitz-continuous partial gradients.

3.1 Algorithm Structure

PALM alternates updates between the variable blocks x and y. At each step, the smooth term H is linearised, and a proximal step is applied to the nonsmooth components. A typical update follows:

$$x^{k+1} \in \operatorname{prox}_{\sigma}^{t}\left(x^k - \frac{1}{t}\nabla_x H(x^k, y^k)\right),$$

which arises from minimising the local quadratic approximation:

$$x^{k+1} \in \arg\min_{x\in\mathbb{R}^d}\left\{\langle x - x^k, \nabla h(x^k)\rangle + \frac{t}{2}\|x - x^k\|^2 + \sigma(x)\right\}.$$

3.2 PALM Algorithm

Algorithm 1 Proximal Alternating Linearised Minimisation (PALM)

```
Require: Initial point $(x^0, y^0) \in \mathbb{R}^n \times \mathbb{R}^m$, parameters $\gamma_1 > 1$, $\gamma_2 > 1$
1: for each iteration $k = 0, 1, 2, \dots$ do
2:     Compute $c_k = \gamma_1 \cdot L_1(y^k)$
3:     Update $x^{k+1} \in \operatorname{prox}_{c_k}^{f}\left(x^k - \frac{1}{c_k}\nabla_x H(x^k, y^k)\right)$
4:     Compute $d_k = \gamma_2 \cdot L_2(x^{k+1})$
5:     Update $y^{k+1} \in \operatorname{prox}_{d_k}^{g}\left(y^k - \frac{1}{d_k}\nabla_y H(x^{k+1}, y^k)\right)$
6: end for
```

3.3 Convergence of PALM

The convergence of PALM is guaranteed under the following assumptions:

(A1) The functions Ψ, f, and g are bounded from below.

(A2) The partial gradients $\nabla_x H(x, y)$ and $\nabla_y H(x, y)$ are Lipschitz continuous.

(A3) The local Lipschitz constants satisfy

$$\lambda_1^- \le L_1(y^k) \le \lambda_1^+, \quad \lambda_2^- \le L_2(x^k) \le \lambda_2^+.$$

(A4) The full gradient ∇H is Lipschitz continuous on bounded subsets.

(A5) The objective Ψ satisfies the Kurdyka–Łojasiewicz (KL) property.

If these assumptions hold and Ψ satisfies the KL property, then PALM is guaranteed to converge based on the following theorem, as stated in [1]:

Theorem 1 (Finite Length and Convergence of PALM)**.** *Let $\{z_k\}$ denote the sequence generated by PALM. Assume further that $\{z_k\}$ is bounded. Then:*

(i) The sequence has finite length:

$$\sum_{k=1}^{\infty} \|z_{k+1} - z_k\| < \infty.$$

(ii) The sequence $\{z_k\}$ converges to a critical point $z^* = (x^*, y^*)$ of Ψ.

A detailed proof of this theorem is provided in Appendix [A].

3.4 Convergence Rate

If Ψ is semi-algebraic—a property satisfied by many objectives involving ℓ_0, ℓ_1, or indicator functions—then the KL property ensures that convergence occurs at a sublinear rate. In the worst case, PALM satisfies:

$$\Psi(x^k, y^k) - \Psi(x^\star, y^\star) = \mathcal{O}\left(\frac{1}{k}\right),$$

where $(x^\star, y^\star)$ is a critical point of the objective. This sublinear rate is typical of first-order methods without acceleration [6].

4 Related Work on Momentum in PALM

To improve the convergence speed of PALM in nonconvex and nonsmooth settings, a number of momentum-based extensions have been proposed. These differ primarily in how momentum is applied—either through inertial terms added directly to the update direction, or through extrapolation strategies that modify the evaluation point of the proximal step.

One of the earliest and most well-known variants is **iPALM** [2], which introduces inertial dynamics by incorporating a weighted difference of previous iterates into each update:

$$x^{k+1} = \text{prox}^{f}_{\eta_k}\left(x^k + \alpha_1^k(x^k - x^{k-1}) - \frac{1}{\eta_k}\nabla_x H(x^k, y^k)\right).$$

The same structure is used for the y-block. While iPALM has shown empirical improvements over PALM, the choice of inertial parameters α_k, β_k is critical and affects both stability and performance. Partial convergence results are available under specific assumptions, including boundedness of iterates and Lipschitz continuity of the coupling term.

A different approach is taken by **PALM-NUT** [7], which applies Nesterov-style extrapolation to the input of the gradient and proximal steps. Each block is extrapolated independently using a tunable factor:

$$\tilde{x}^k = x^k + \theta_1^k (x^k - x^{k-1}),$$

and the update proceeds using the PALM framework at the extrapolated point. This form of momentum is more structured than simple inertia and often yields more stable performance. However, the extrapolation weights must still be chosen carefully, and convergence analysis is more limited than for iPALM.

A further extension is **ACC-PALM** [8], which draws from variational inequality formulations and uses a FISTA-like extrapolation strategy followed by averaging:

$$x^{k+1} = \operatorname{prox}_{\eta_k}^{f}\left(\tilde{x}^k - \frac{1}{\eta_k}\nabla_x H(\tilde{x}^k, y^k)\right), \quad \tilde{x}^{k+1} = x^{k+1} + \beta_k (x^{k+1} - x^k).$$

While ACC-PALM is theoretically appealing in smooth or convex settings, its benefit in highly nonconvex problems is less clear due to the smoothing effect of the averaging step.

In addition to these deterministic methods, several **stochastic variants of PALM** [9] have been developed, particularly for large-scale problems where evaluating full gradients is expensive. These methods perform partial or mini-batch updates, often using coordinate sampling or randomized block selection. Stochastic iPALM variants [10], for example, incorporate inertial terms while updating only a subset of coordinates at each iteration. However, integrating structured momentum like extrapolation into stochastic updates remains challenging due to potential instability and variance amplification.

Together, these works highlight the diversity of momentum mechanisms that can be applied to PALM, each with distinct trade-offs in terms of tuning complexity, convergence behaviour, and suitability for large-scale or noisy settings.

5 The FISTA-PALM Algorithm

FISTA-PALM is a hybrid algorithm that combines the alternating structure of PALM with the momentum-based extrapolation of FISTA [3]. This introduces structured acceleration into block-coordinate optimisation, enabling faster convergence while preserving PALM's modular updates.

Momentum techniques like iPALM accelerate PALM by adding inertial terms directly to the iterates. In contrast, FISTA-style acceleration modifies the evaluation point before updating. This extrapolation-based momentum is simple to implement, tuneless, and often leads to faster empirical convergence. FISTA-PALM adapts this principle to block-coordinate settings by applying extrapolation before each block-wise update.

5.1 Problem Setting

We consider problems of the form:

$$\min_{(x,y)} \ \Psi(x,y) := f(x) + g(y) + H(x,y),$$

where:

- f and g are proper, lower semicontinuous, possibly nonsmooth or nonconvex functions,
- H is smooth with Lipschitz-continuous partial gradients $\nabla_x H$ and $\nabla_y H$.

5.2 Assumptions

Since FISTA-PALM builds on PALM, we adopt the standard convergence assumptions stated in Section 3.3. Although extrapolation is introduced, empirical stability similar to PALM is observed.

5.3 Algorithm Description

At each iteration k, FISTA-PALM performs:

1. **Extrapolation**:

$$\tilde{x}^k = x^k + \frac{t_{k-1}-1}{t_k}(x^k - x^{k-1}),$$

$$\tilde{y}^k = y^k + \frac{t_{k-1}-1}{t_k}(y^k - y^{k-1}),$$

2. **Proximal Updates**:

$$x^{k+1} = \text{prox}^{f}_{\eta_k}\left(\tilde{x}^k - \frac{1}{\eta_k}\nabla_x H(\tilde{x}^k, y^k)\right),$$
$$y^{k+1} = \text{prox}^{g}_{\tau_k}\left(\tilde{y}^k - \frac{1}{\tau_k}\nabla_y H(x^{k+1}, \tilde{y}^k)\right),$$

3. **Momentum Update**:

$$t_{k+1} = \frac{1 + \sqrt{1 + 4t_k^2}}{2}.$$

5.4 Parameter Selection

The local stepsizes are computed from blockwise Lipschitz constants:

$$L_1(y_k) = \|y_k^\top y_k\|_2, \quad L_2(x_{k+1}) = \|x_{k+1}^\top x_{k+1}\|_2,$$

and set as

$$\eta_k = \gamma_1 L_1(y_k), \quad \tau_k = \gamma_2 L_2(x_{k+1}),$$

with overshoot parameters $\gamma_1, \gamma_2 > 1$. When direct estimation of L_1 and L_2 is infeasible, a backtracking strategy can be used Appendix [B].

5.5 Initialisation

The method is initialised by:

- selecting feasible points (x_0, y_0),
- setting history $x_{-1} = x_0$, $y_{-1} = y_0$,
- setting momentum parameter $t_0 = 1$.

5.6 Convergence Rate Considerations

FISTA achieves $\mathcal{O}(1/k^2)$ convergence in convex settings [3], while PALM achieves $\mathcal{O}(1/k)$ under the KL property. Although formal guarantees for nonconvex FISTA-PALM are lacking, empirical results suggest similar accelerated behaviour, especially with conservative stepsizes and bounded iterates.

5.7 Monotonicity and Accuracy

Due to the extrapolation step, FISTA-PALM is not guaranteed to be monotonic: the objective may oscillate before converging, consistent with classical FISTA behaviour[3]. While this can cause slight overshooting near the minimum, empirical results show that FISTA-PALM still achieves competitive final objective values, with significantly faster early-stage convergence compared to PALM. Careful stepsize selection or adaptive restart strategies could further improve monotonicity and improve solution accuracy, although these refinements are not pursued here.

5.8 Additional Remarks

Memory and computation: FISTA-PALM requires only one additional vector per block for extrapolation and retains PALM's low computational overhead. Practical experiments show that it can achieve faster per-iteration times by making more efficient use of gradient evaluations.

Extension to multi-block problems: FISTA-PALM generalises naturally to more than two blocks, with each block extrapolated and updated independently.

Error accumulation: Although Nesterov-type methods can amplify numerical errors, no such instability was observed. FISTA-PALM remained stable even in low-curvature and ill-conditioned regimes. This could be due to the regularising effect of PALM's alternating proximal structure, though further analysis would be needed to fully characterise this behaviour.

Caveats: No theoretical convergence proofs currently exist for nonconvex FISTA-PALM; it remains an empirically effective method, with formal convergence analysis left for future work.

6 Numerical Results

We now evaluate the performance of our proposed hybrid algorithm, **FISTA-PALM**, on a range of matrix factorisation tasks. The goal is to assess how well it performs under different data regimes, initialisation strategies, and sparsity levels. For reference, we compare against standard **PALM** and the inertial variant **iPALM**.

Matrix factorisation provides a strong testbed for these experiments, as it naturally exhibits a block-separable structure and arises in applications across recommender systems, signal processing, and computer vision.

6.1 Problem Formulation

We consider the general form of the sparse nonnegative matrix factorisation (NMF) problem:

$$\min\left\{\frac{1}{2}\|A - XY\|_F^2 : X \geq 0,\ Y \geq 0\right\},$$

where $A \in \mathbb{R}^{m\times n}$, and the goal is to approximate A using low-rank nonnegative factors $X \in \mathbb{R}^{m\times r}$ and $Y \in \mathbb{R}^{r\times n}$.

To promote sparsity, we consider two variants of this problem depending on the experiment:

- **ℓ_0-constrained formulation (Synthetic and ORL Faces):**

$$\min\left\{\frac{1}{2}\|A - XY\|_F^2 : X \geq 0,\ Y \geq 0,\ \|X\|_0 \leq s_x,\ \|Y\|_0 \leq s_y\right\}.$$

- **ℓ_1-regularised formulation (Berkeley Patches):**

$$\min\left\{\frac{1}{2}\|A - XY\|_F^2 + \lambda\left(\|X\|_1 + \|Y\|_1\right) : X \geq 0,\ Y \geq 0\right\}.$$

The ℓ_0 formulation allows for direct control over sparsity, while the ℓ_1 penalty acts as a convex surrogate. Both regularisers are semi-algebraic, satisfying the structural conditions required for PALM-type convergence guarantees.

The block-separable, nonsmooth, and nonconvex nature of the problem fits the design of PALM, and thus FISTA-PALM algorithms, which exploit such structures through alternating updates.

6.2 Proximal Step

Each update in FISTA-PALM requires computing the following proximal operator:

$$\operatorname{prox}_1^f(U) = \arg\min\left\{\frac{1}{2}\|X - U\|_F^2 : X \geq 0,\ \|X\|_0 \leq s\right\}.$$

We observe that due to the nonnegativity constraint, any negative entries in U will be zeroed out in the solution. Let:

$$P_+(U) := \max(U, 0),$$

denote projection onto the nonnegative region. The proximal problem becomes:

$$\operatorname{prox}_1^f(U) = \arg\min\left\{\|X - P_+(U)\|_F^2 : \|X\|_0 \leq s\right\}.$$

This is a standard sparse approximation problem in Frobenius norm. The optimal solution is obtained by applying the hard-thresholding operator T_s, which retains the top s entries in magnitude:

$$\mathrm{prox}_1^f(U) = T_s(P_+(U)).$$

This closed-form update is efficient and fits naturally into the alternating block structure of FISTA-PALM. The full calculation is provided in Appendix [C]

6.3 Experimental Setup

We evaluate all methods across the following datasets:

- **Synthetic matrices:** randomly generated low-rank, sparse nonnegative matrices.
- **ORL Faces** [11]: structured, low-noise greyscale face images with variations in expression and lighting.
- **Berkeley patches (BSDS500)** [12]: natural, noisy image patches extracted from the Berkeley Segmentation Dataset.
- **Columbia Object Image Library (COIL-20)** [13]: structured greyscale object images with rotational variations.

Comparison Baselines. We compare FISTA-PALM against two baselines: the standard PALM algorithm and the inertial variant iPALM. Although iPALM is not the primary focus of this study, it serves as a useful adapted baseline for evaluating the benefits of momentum schemes. Notably, iPALM requires careful tuning of inertial parameters (α, β) via grid search, whereas FISTA-PALM achieves strong performance with minimal parameter adjustment. In larger-scale experiments, identifying stable inertial parameters for iPALM proved challenging, further highlighting the robustness and practicality of FISTA-PALM.

Initialisation and Sparsity. Experiments are conducted with both random and SVD-based initialisation to assess sensitivity to starting conditions. We test three levels of sparsity—10%, 25%, and 33%—applied to either X, Y, or both, depending on the setting. Random seeds and hyperparameters are fixed across all methods to ensure fair comparison.

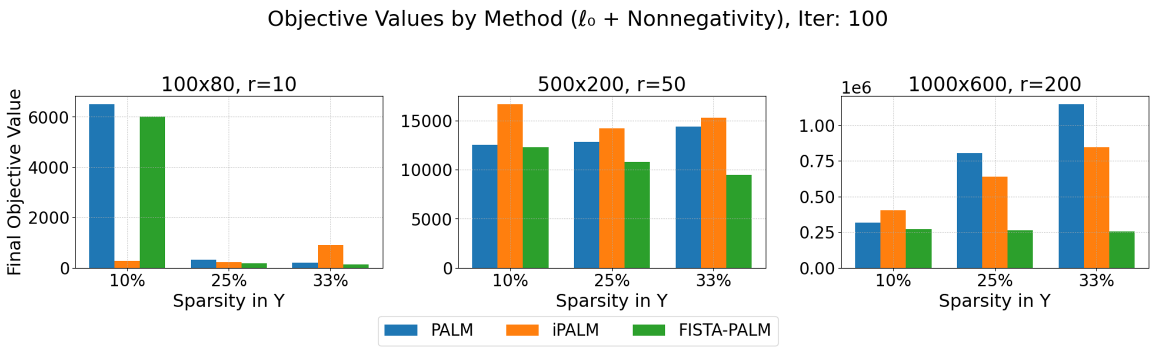

Figure 1: Effect of Sparsity across different matrix sizes

Synthetic Data Generation. Synthetic datasets are generated by setting $A = X^*(Y^*)^\top$, where X^* and Y^* are low-rank, sparse, nonnegative matrices. Matrix sizes are varied as follows:

Rows	**Columns**	**Rank**	**Size Category**
100	80	10	Small
500	200	50	Medium
1000	600	200	Large

Table 1: Dimensions, ranks, and size categories for synthetic matrices.

These controlled settings enable evaluation of convergence behaviour, scalability, and the impact of initialisation strategies.

Termination Criteria. For synthetic datasets, experiments are terminated after 100 iterations. For real datasets (ORL Faces, Berkeley patches, COIL-20), the limit is 200 iterations. These choices reflect stagnation issues commonly observed in block-coordinate methods such as PALM [14]. Additionally, a residual-based stopping criterion is evaluated in selected experiments to assess convergence dynamically.

6.4 Empirical Observations

6.4.1 Synthetic Data

Effect of Sparsity on Objective Values. We first examine the effect of sparsity levels on final objective values after 100 iterations, as shown in Fig. ??. Across all matrix sizes, increasing sparsity generally leads to lower objective values. FISTA-PALM performs best at moderate sparsity levels (25% and 33%), where enough nonzero elements remain for stable updates. At very high sparsity (10%), FISTA-PALM's performance becomes slightly less stable due to overshooting when few variables are active. Compared to iPALM, FISTA-PALM achieves lower objective values in most settings; however, in small-scale, highly constrained problems, carefully tuned iPALM can outperform it, suggesting that cautious inertial updates are better suited to severely limited solution spaces. These results indicate that maintaining moderate sparsity is important for FISTA-PALM to fully leverage its acceleration and achieve consistent convergence.

Effect of Initialisation and Final Objective Values. We next examine the effect of initialisation strategies on convergence. Choosing SVD-based initialisation over random initialisation benefits all methods, particularly PALM and iPALM. FISTA-PALM also benefits, making rapid early progress even from random starts, but tends to overshoot when close to the minimum, leading to slower final refinement compared to the baselines.

Despite the faster initial descent with SVD initialisation, the number of iterations required to reach convergence, defined by the relative tolerance

$$\frac{\|X_{k+1} - X_k\|_F}{\|X_k\|_F} < \epsilon, \quad \epsilon = 10^{-4},$$

remains broadly similar between random and SVD initialisation across all methods. Table 2 reports the final objective values and iteration counts across different matrix sizes. PALM and iPALM show significant reductions in iteration counts with SVD initialisation, especially for smaller matrices. In contrast, FISTA-PALM remains largely insensitive to initialisation, while still consistently achieving lower final objective values compared to PALM and iPALM.

Method	**Small**		**Medium**		**Large**	
	Obj	**Iter**	**Obj**	**Iter**	**Obj**	**Iter**
PALM (Random)	116	3621	12100	266	2.73e5	701
PALM (SVD)	299	46	5480	126	1.15e5	376
iPALM (Random)	353	418	5530	3700	1.74e5	4430
iPALM (SVD)	1600	5000	4520	1100	9.88e4	1178
FISTA-PALM	93	386	2540	1470	3.74e4	3035
FISTA-PALM (SVD)	94	365	2720	1139	3.53e4	2382

Table 2: Final objective values and iteration counts for PALM, iPALM, and FISTA-PALM across different matrix sizes.

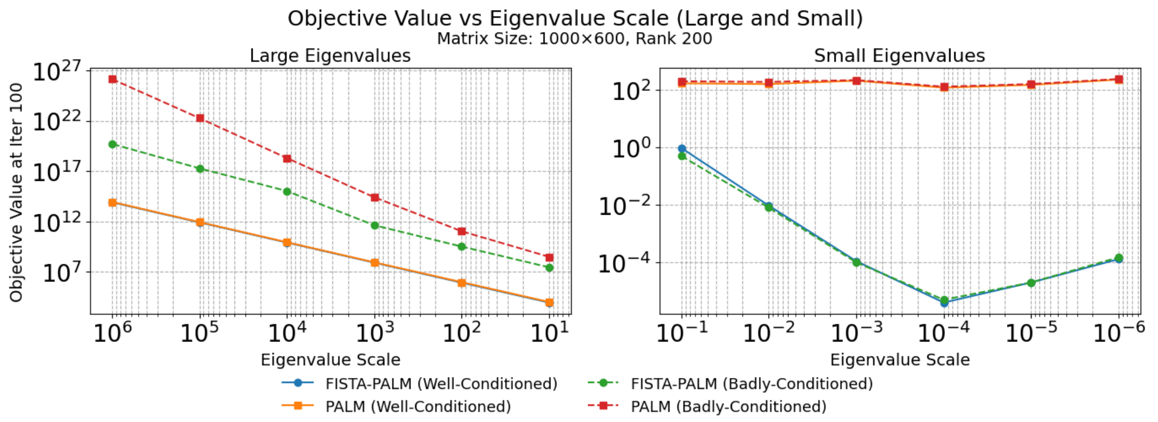

Figure 2: Sensitivity of PALM and FISTA-PALM to eigenvalue scaling under varying matrix conditioning.

Effect of Eigenvalue Scaling. We evaluate the robustness of PALM and FISTA-PALM under extreme eigenvalue distributions, a key challenge in matrix factorisation where real-world data often exhibit ill-conditioning. Synthetic matrices were generated with eigenvalues either in the small range (10^{-1} to 10^{-6}) or the large range (10^{1} to 10^{6}), simulating both well-conditioned and badly-conditioned settings.

As shown in Fig. 2, PALM becomes unstable under extreme eigenvalues: diverging for large eigenvalues and stagnating for small eigenvalues. FISTA-PALM, in contrast, maintains stable progress across most cases. Notably, large eigenvalues do not significantly degrade FISTA-PALM's performance, as the method continues to take controlled extrapolated steps. However, for small eigenvalues (below 10^{-4}), overshooting occurs even in well-conditioned matrices, leading to slight increases in objective value. This suggests that the primary difficulty lies not in conditioning, but in the small scale of the eigenvalues themselves, which amplify the effects of momentum.

Overall, FISTA-PALM demonstrates greater robustness than PALM, although sensitivity to very small eigenvalues highlights a natural limitation of acceleration-based methods.

6.4.2 ORL Face Dataset

Performance on Benchmarked Images. We now evaluate performance on the ORL dataset, a longstanding benchmark for assessing matrix factorisation and sparse coding algorithms. The dataset comprises 400 greyscale face images of size 64×64, arranged as columns in a 4096×400 matrix. This structure naturally supports low-

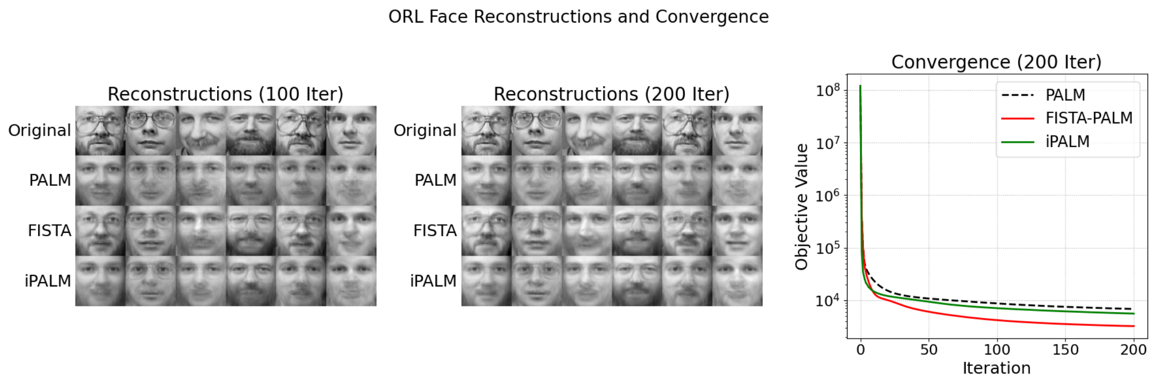

Figure 3: Comparitive results of running PALM, iPALM and Fista for 100 and 200 iterations on ORL dataset

Iterations	PALM	iPALM	FISTA-PALM
100	8686.89	6302.89	4171.40
200	6812.32	4610.40	3213.42
500	5176.95	3500.76	2849.15
1000	4233.43	3180.63	2826.36
2000	3644.30	2998.68	2825.69

Table 3: Comparison of objective values after multiple iterations for PALM, iPALM, and FISTA-PALM on the ORL dataset.

rank decomposition under sparsity constraints.

Using the same experimental setup as in the synthetic case, we observe that FISTA-PALM outperforms both PALM and iPALM from the outset. As shown in Fig. 3, reconstructed faces produced by FISTA-PALM become noticeably clearer within the first 100 iterations. The early emergence of fine detail suggests that FISTA-PALM is particularly well-suited to this dataset, likely due to its smooth structure and relatively low noise levels.

Quantitative Convergence. Objective values across different iteration counts are summarised in Table 3.

As shown, FISTA-PALM achieves nearly half the loss of PALM within the first 100 iterations. However, as the number of iterations increases, its rate of improvement begins to diminish. This plateauing behaviour mirrors the synthetic initialisation experiments (Section 6.4.1), where FISTA-PALM made rapid early progress but occasionally struggled to converge fully due to overshooting near the minimum,

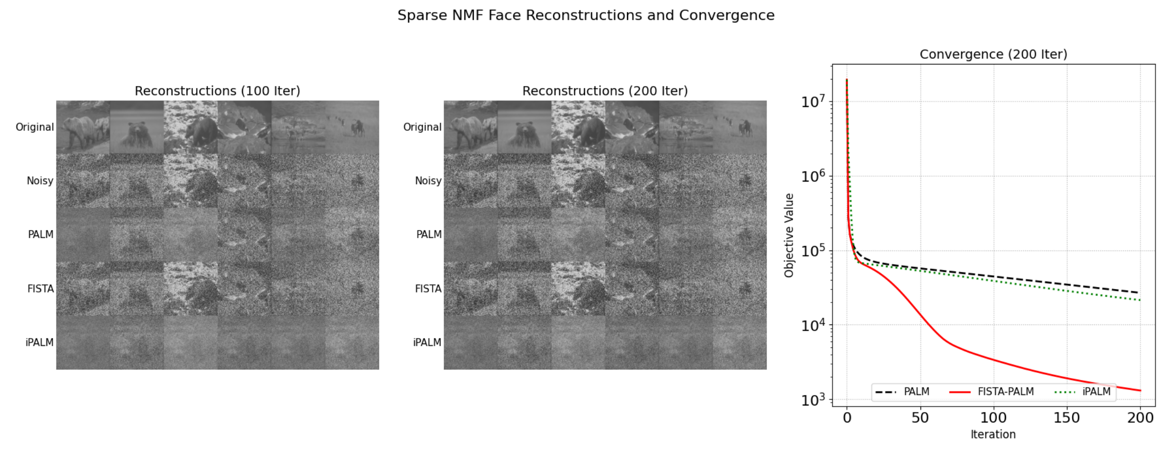

Figure 4: Comparitive results of running PALM, iPALM and Fista for 100 and 200 iterations

a common feature of acceleration methods such as Nesterov's accelerated gradient.

Nonetheless, both iPALM and FISTA-PALM consistently outperform the standard PALM algorithm on this dataset, reinforcing the importance of incorporating acceleration schemes for achieving practical performance gains.

6.4.3 Berkeley Patches

A More Challenging Problem. We next consider a more challenging setting involving natural image data from the Berkeley Segmentation Dataset (BSDS500). In this task, the objective incorporates ℓ_1 regularisation alongside nonnegativity constraints, replacing the hard ℓ_0 sparsity used previously. This modification increases the problem's complexity due to the high dimensionality of the patches (128×128) and the addition of independent Gaussian noise $(\mathcal{N}(0, 0.3^2))$ clipped to $[0, 1]$ on each pixel.

Despite the increased difficulty and extreme sparsity, FISTA-PALM continues to perform robustly. As shown in Fig. 4, it achieves cleaner reconstructions more rapidly than both PALM and iPALM. Notably, FISTA-PALM is less affected by noise, even though natural textures and fine details are particularly vulnerable to corruption. The superiority of FISTA-PALM is further reflected in the objective values reported in Table 4.

FISTA-PALM attains nearly half the loss of PALM within 100 iterations and, by 200 iterations, outperforms iPALM by more than an order of magnitude. This improvement can be attributed to FISTA-PALM's use of extrapolation, which accelerates convergence by leveraging information from previous iterates. The momentum-

Iterations	PALM	iPALM	FISTA-PALM
100	27815.71	29670.61	14947.54
200	26799.22	21391.59	1307.41

Table 4: Comparison of objective values after 100 and 200 iterations for PALM, iPALM, and FISTA-PALM.

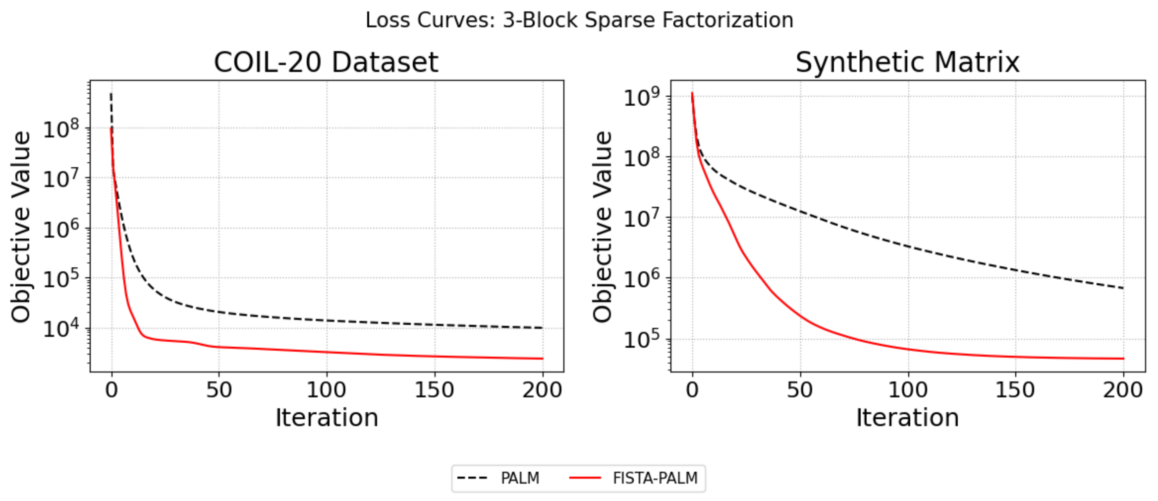

Figure 5: Loss curves for the three-block case on the COIL-20 dataset and synthetic data.

like updates help overcome small oscillations introduced by noise and enable more efficient progress through flat or ill-conditioned regions. Notably, this behaviour is consistent with the eigenvalue sensitivity analysis presented earlier (Section 6.4.1), where FISTA-PALM demonstrated robustness across a wide range of eigenvalue scales, except when extreme flatness ($< 10^{-4}$) was present. In the present Berkeley setting, eigenvalues are moderately small but not extreme, allowing FISTA-PALM's extrapolation to remain highly effective. These properties allow FISTA-PALM to maintain accelerated convergence under noise and sparsity, establishing it as a versatile solver.

6.4.4 3-Block Factorisation

We extend the standard two-block matrix factorisation to a three-block structure within the FISTA-PALM framework (see Appendix [D] for algorithm). This formulation, relevant in settings involving an intermediate latent transformation, is given by:

$$\min_{X,B,Y} \frac{1}{2} \left\| A - XBY^\top \right\|_F^2 \quad \text{s.t.} \quad (X,B,Y) \geq 0,\ \|X\|_0 \leq s_x,\ \|B\|_0 \leq s_b,\ \|Y\|_0 \leq s_y.$$

We evaluate performance on two datasets: a synthetically generated low-rank matrix and the COIL-20 image dataset [13], a standard benchmark for object recognition. COIL-20 images are converted to greyscale vectors and stacked column-wise to form the matrix A. As shown in Fig. 5, FISTA-PALM consistently achieves faster early-stage convergence than PALM across both datasets, even in the more complex three-block setting. These results demonstrate the generalisability of the extrapolation mechanism to richer factorisation structures and suggest that FISTA-PALM remains effective for more advanced block-structured problems, maintaining accelerated convergence without compromising reconstruction fidelity.

7 Conclusion

In this work, we introduced FISTA-PALM, an accelerated variant of PALM that incorporates structured FISTA-style extrapolation into block-coordinate updates for nonconvex, nonsmooth optimisation problems. Compared to existing momentum-based extensions such as iPALM, FISTA-PALM requires significantly less parameter tuning while preserving PALM's simplicity and low computational overhead.

We evaluated FISTA-PALM across a broad range of matrix factorisation tasks, including Frobenius norm objectives, ℓ_0 sparsity constraints, and ℓ_1 regularisation. Experiments on synthetic data, the ORL face dataset, Berkeley natural images, and a three-block factorisation setting show that FISTA-PALM consistently achieves faster early-stage convergence than PALM and iPALM across varying data structures.

Nonetheless, we observed some limitations. While FISTA-PALM remains robust even under ill-conditioned settings, it occasionally overshoots near the minimum, leading to oscillations and slower final refinement. This behaviour is typical of Nesterov-type acceleration methods and suggests that incorporating adaptive restart strategies could further improve convergence stability.

Future work could explore dynamic restart schemes, extensions to more complex structured problems beyond matrix factorisation, and theoretical convergence guarantees for the extrapolated block-coordinate framework. Overall, FISTA-PALM provides an effective and practical alternative to existing PALM variants, achieving accelerated convergence with minimal tuning effort across diverse matrix factorisation problems.

References

[1] Jérôme Bolte, Shoham Sabach, and Marc Teboulle. Proximal alternating linearized minimization for nonconvex and nonsmooth problems. *Mathematical Programming*, 146:459–494, 2014.

[2] Thomas Pock and Shoham Sabach. Inertial proximal alternating linearized minimization (ipalm) for nonconvex and nonsmooth problems. *SIAM Journal on Imaging Sciences*, 9:1756–1787, 2016.

[3] Amir Beck and Marc Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences*, 2:183–202, 2009.

[4] Robert Peharz and Franz Pernkopf. Sparse nonnegative matrix factorization with 0-constraints. *Neurocomputing*, 80:38–46, 2012.

[5] Michael W. Berry, Murray Browne, Amy N. Langville, V. Paul Pauca, and Robert J. Plemmons. Algorithms and applications for approximate nonnegative matrix factorization. *Computational Statistics & Data Analysis*, 52:155–173, 2007.

[6] Yurii Nesterov. *Introductory Lectures on Convex Optimization: A Basic Course*. Springer, New York, 2004.

[7] Yunsong Liu and Justin P. Haldar. Palmnut: An enhanced proximal alternating linearized minimization algorithm with application to separate regularization of magnitude and phase, 2020.

[8] Damek Davis. The asynchronous palm algorithm for nonsmooth nonconvex problems, 2016.

[9] Derek Driggs, Junqi Tang, Jingwei Liang, Mike Davies, and Carola-Bibiane Schönlieb. Spring: A fast stochastic proximal alternating method for non-smooth non-convex optimization, 2021.

[10] Johannes Hertrich and Gabriele Steidl. Inertial stochastic palm and applications in machine learning. *Sampling Theory, Signal Processing, and Data Analysis*, 20:1–24, 2022.

[11] Ferdinand Samaria and Andy Harter. Parameterisation of a stochastic model for human face identification, 1994.

[12] Pablo Arbeláez, Michael Maire, Charless Fowlkes, and Jitendra Malik. Contour detection and hierarchical image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33:898–916, 2011.

[13] S. A. Nene, S. K. Nayar, and H. Murase. Columbia object image library (coil-20). Technical Report CUCS-005-96, Department of Computer Science, Columbia University, February 1996.

[14] Amir Beck and Luba Tetruashvili. On the convergence of block coordinate descent type methods. *SIAM Journal on Optimization*, 23(4):2037–2060, 2013.

A Proof of PALM Convergence

We provide a proof of convergence for the PALM algorithm, following the structure and notation of [1].

Lemma 1 (Descent Lemma). *Let $h : \mathbb{R}^d \to \mathbb{R}$ be a continuously differentiable function with L_h-Lipschitz continuous gradient. Then, for all $u, v \in \mathbb{R}^d$,*

$$h(u) \leq h(v) + \langle u - v, \nabla h(v) \rangle + \frac{L_h}{2} \|u - v\|^2.$$

Lemma 2 (Sufficient Decrease Property). *Let $h : \mathbb{R}^d \to \mathbb{R}$ be continuously differentiable with L_h-Lipschitz gradient, and let $\sigma : \mathbb{R}^d \to (-\infty, +\infty]$ be a proper, lower semicontinuous function with $\inf \sigma > -\infty$. Fix $t > L_h$. Define*

$$u^+ \in \operatorname{prox}_t^\sigma \left(u - \frac{1}{t} \nabla h(u) \right).$$

Then,

$$h(u^+) + \sigma(u^+) \leq h(u) + \sigma(u) - \frac{1}{2}(t - L_h)\|u^+ - u\|^2.$$

Proof. From the definition of the proximal operator, we have

$$0 \in \nabla h(u) + t(u^+ - u) + \partial \sigma(u^+).$$

Thus,

$$t(u - u^+) - \nabla h(u) \in \partial \sigma(u^+).$$

By the subdifferential inequality,

$$\sigma(v) \geq \sigma(u^+) + \langle v - u^+, t(u - u^+) - \nabla h(u) \rangle \quad \forall v \in \mathbb{R}^d.$$

Choosing $v = u$ and expanding:

$$\sigma(u) \geq \sigma(u^+) + t\|u^+ - u\|^2 + \langle \nabla h(u), u^+ - u \rangle.$$

Applying the descent lemma,

$$h(u^+) \leq h(u) + \langle \nabla h(u), u^+ - u \rangle + \frac{L_h}{2}\|u^+ - u\|^2.$$

Adding the two inequalities gives the result. $\square$

Lemma 3 (Sufficient Decrease for PALM Iterates). *Let $\{z_k = (x_k, y_k)\}$ be the sequence generated by PALM. Then:*

$$\Psi(z_k) - \Psi(z_{k+1}) \geq \frac{\rho_1}{2}\|z_{k+1} - z_k\|^2,$$

where $\rho_1 := \min\{(\gamma_1 - 1)\lambda_1^-, (\gamma_2 - 1)\lambda_2^-\}$.

Proof. Apply the sufficient decrease lemma separately to the x- and y-updates. For the x-update:

$$H(x_{k+1}, y_k) + f(x_{k+1}) \leq H(x_k, y_k) + f(x_k) - \frac{1}{2}(\gamma_1 - 1)L_1(y_k)\|x_{k+1} - x_k\|^2.$$

For the y-update:

$$H(x_{k+1}, y_{k+1}) + g(y_{k+1}) \leq H(x_{k+1}, y_k) + g(y_k) - \frac{1}{2}(\gamma_2 - 1)L_2(x_{k+1})\|y_{k+1} - y_k\|^2.$$

Adding gives:

$$\Psi(z_k) - \Psi(z_{k+1}) \geq \frac{1}{2}(\gamma_1 - 1)L_1(y_k)\|x_{k+1} - x_k\|^2 + \frac{1}{2}(\gamma_2 - 1)L_2(x_{k+1})\|y_{k+1} - y_k\|^2.$$

Using $L_1(y_k) \geq \lambda_1^-$ and $L_2(x_{k+1}) \geq \lambda_2^-$ yields the result. $\square$

Lemma 4 (Finite Length). *The sequence $\{z_k\}$ satisfies:*

$$\sum_{k=0}^{\infty} \|z_{k+1} - z_k\|^2 < \infty, \quad \text{and} \quad \|z_{k+1} - z_k\| \to 0.$$

Proof. Summing the sufficient decrease inequality from $k = 0$ to $N - 1$ gives:

$$\sum_{k=0}^{N-1} \|z_{k+1} - z_k\|^2 \leq \frac{2}{\rho_1}(\Psi(z_0) - \Psi(z_N)).$$

Since Ψ is bounded below, the right-hand side is bounded as $N \to \infty$, proving convergence of the series. Thus $\|z_{k+1} - z_k\| \to 0$. $\square$

Lemma 5 (Subgradient Bound)**.** *Define*

$$A_k^x := c_{k-1}(x_{k-1} - x_k) + \nabla_x H(x_k, y_k) - \nabla_x H(x_{k-1}, y_{k-1}),$$

$$A_k^y := d_{k-1}(y_{k-1} - y_k) + \nabla_y H(x_k, y_k) - \nabla_y H(x_k, y_{k-1}).$$

Then:

(i) $(A_k^x, A_k^y) \in \partial\Psi(x_k, y_k)$*,*

(ii) There exists $M > 0$ *such that*

$$\|(A_k^x, A_k^y)\| \le (2M + 3\rho_2)\|z_k - z_{k-1}\|,$$

where $\rho_2 := \max\{\gamma_1\lambda_1^+, \gamma_2\lambda_2^+\}$*.*

Proof. The optimality conditions imply (i). The Lipschitz continuity of ∇H on bounded sets ensures (ii). □

Theorem 2 (Global Convergence of PALM)**.** *Suppose Assumptions A1 to A5 hold and* Ψ *is a KL function. Then:*

(i) The sequence $\{z_k\}$ *has finite length.*

(ii) The sequence $\{z_k\}$ *converges to a critical point* z^**.*

Proof. From the finite-length property and subgradient bounds, $\|(A_k^x, A_k^y)\| \to 0$. By the KL property and boundedness of $\{z_k\}$, we conclude that $\{z_k\}$ is a Cauchy sequence and converges to a critical point z^*. □

B Calculation of Lipschitz Constant using Backtracking

To adaptively estimate local Lipschitz constants in FISTA-PALM, we employ a backtracking strategy following the classical FISTA approach. At each block update, the Lipschitz constant is adjusted to satisfy a sufficient decrease condition based on a quadratic approximation of the smooth coupling term H.

Given (x_k, x_{k-1}), y_k, and the momentum parameter t_k, we compute the extrapolated point

$$\tilde{x}_k = x_k + \frac{t_{k-1} - 1}{t_k}(x_k - x_{k-1}),$$

initialise $L_x > 0$, and perform the proximal update

$$x_{k+1} = \operatorname{prox}_{1/L_x} f\left(\tilde{x}_k - \frac{1}{L_x}\nabla_x H(\tilde{x}_k, y_k)\right).$$

The Lipschitz constant L_x is increased until the sufficient decrease condition

$$\Psi(x_{k+1}, y_k) \le H(\tilde{x}_k, y_k) + \langle \nabla_x H(\tilde{x}_k, y_k), x_{k+1} - \tilde{x}_k\rangle + \frac{L_x}{2}\|x_{k+1} - \tilde{x}_k\|^2 + f(x_{k+1}) + g(y_k)$$

is satisfied. The same procedure is applied to the y-block update, holding x_{k+1} fixed.

While backtracking introduces additional computational overhead, it enhances adaptivity to local curvature and improves the robustness of each block update.

C Calculation of Proximal Operator for NMF Problems

Definition:. Given any matrix $U \in \mathbb{R}^{m\times n}$, define the operator $T_s : \mathbb{R}^{m\times n} \rightrightarrows \mathbb{R}^{m\times n}$ by

$$T_s(U) := \arg\min_{V\in\mathbb{R}^{m\times n}} \left\{\|U - V\|_F^2 : \|V\|_0 \le s\right\}.$$

Proposition 1 (Proximal map formula)**.** *Let $U \in \mathbb{R}^{m\times n}$ and define $f := \delta_{\{X\ge 0\}} + \delta_{\{\|X\|_0\le s\}}$. Then*

$$\operatorname{prox}_f^1(U) = \arg\min\left\{\frac{1}{2}\|X - U\|_F^2 : X \ge 0, \|X\|_0 \le s\right\} = T_s\left(P_+(U)\right),$$

Definition:. Given any matrix $U \in \mathbb{R}^{m\times n}$, define the operator $T_s : \mathbb{R}^{m\times n} \rightrightarrows \mathbb{R}^{m\times n}$ by

$$T_s(U) := \arg\min_{V\in\mathbb{R}^{m\times n}} \left\{\|U - V\|_F^2 : \|V\|_0 \le s\right\}.$$

Proof. Let $U \in \mathbb{R}^{m\times n}$ be given. Introduce the notations:

$$\|X\|_+^2 := \sum_{(i,j)\in\mathcal{I}^+} X_{ij}^2, \quad \|X\|_-^2 := \sum_{(i,j)\in\mathcal{I}^-} X_{ij}^2, \quad \text{and} \quad \|X\|_F^2 = \|X\|_+^2 + \|X\|_-^2,$$

where

$$\mathcal{I}^+ := \{(i,j) \in \{1,\dots,m\} \times \{1,\dots,n\} : U_{ij} \ge 0\},$$

$$\mathcal{I}^- := \{(i,j) \in \{1,\dots,m\} \times \{1,\dots,n\} : U_{ij} < 0\}.$$

Observe that the following relations hold:

(i) $\|X\|_F^2 = \|X\|_+^2 + \|X\|_-^2$,

(ii) $\|X - U\|_F^2 = \|X - U\|_+^2 + \|X - U\|_-^2$,

(iii) $\|X\|_-^2 = 0$ if and only if $X_{ij} = 0$ for all $(i,j) \in \mathcal{I}^-$.

Using these facts, we analyze the proximal operator:

$$\bar{X} \in \operatorname{prox}_f^1(U) \quad \text{if and only if} \quad \bar{X} \in \arg\min \left\{ \frac{1}{2}\|X - U\|_F^2 : X \geq 0, \|X\|_0 \leq s \right\}.$$

Expanding $\|X - U\|_F^2$ using (ii), we obtain:

$$\|X - U\|_F^2 = \|X - U\|_+^2 + \|X - U\|_-^2.$$

Since $(P_+(U))_{ij} = U_{ij}$ for $(i,j) \in \mathcal{I}^+$ and $(P_+(U))_{ij} = 0$ for $(i,j) \in \mathcal{I}^-$, we can expand:

$$\|X - U\|_F^2 = \|X - P_+(U)\|_+^2 + \sum_{(i,j)\in\mathcal{I}^-} (X_{ij} - U_{ij})^2.$$

For $(i,j) \in \mathcal{I}^-$, since $U_{ij} < 0$ and $X_{ij} \geq 0$, it holds that

$$(X_{ij} - U_{ij})^2 = X_{ij}^2 - 2X_{ij}U_{ij} + U_{ij}^2.$$

Thus:

$$\sum_{(i,j)\in\mathcal{I}^-} (X_{ij} - U_{ij})^2 = \|X\|_-^2 - 2\sum_{(i,j)\in\mathcal{I}^-} X_{ij}U_{ij} + \sum_{(i,j)\in\mathcal{I}^-} U_{ij}^2.$$

Since $\sum_{(i,j)\in\mathcal{I}^-} U_{ij}^2$ does not depend on X, it can be ignored in the minimization. Therefore, minimizing the original objective is equivalent to minimizing

$$\|X - P_+(U)\|_+^2 + \|X\|_-^2 - 2\sum_{(i,j)\in\mathcal{I}^-} X_{ij}U_{ij}.$$

Using (iii), we can set $X_{ij} = 0$ for all $(i,j) \in \mathcal{I}^-$, eliminating the second and third terms, and reducing the problem to:

$$\bar{X} \in \arg\min \left\{ \|X - P_+(U)\|_F^2 : \|X\|_0 \leq s \right\},$$

where the nonnegativity constraint $X \geq 0$ is automatically satisfied.

By Definition C, we conclude:

$$\operatorname{prox}_f^1(U) = T_s(P_+(U)).$$

$\square$

D Pseudocode for 3-Block Fista PALM Algorithm

Algorithm 2 FISTA-PALM for 3-Block Matrix Factorization

```
Require: Initial point $(X^0, B^0, Y^0)$, step size parameter $\gamma > 1$
 1: Set $(X^{-1}, B^{-1}, Y^{-1}) = (X^0, B^0, Y^0)$, $t_0 = 1$
 2: for each iteration $k = 0, 1, 2, \dots$ do
 3:     $t_{k+1} = \frac{1+\sqrt{1+4t_k^2}}{2}$
 4:     $\beta_k = \frac{t_k - 1}{t_{k+1}}$
 5:     Extrapolate:
          5.1  $X^{k*} = X^k + \beta_k(X^k - X^{k-1})$
          5.2  $B^{k*} = B^k + \beta_k(B^k - B^{k-1})$
          5.3  $Y^{k*} = Y^k + \beta_k(Y^k - Y^{k-1})$
 6:     Compute $L_X = \gamma_1 \cdot \|B^{k*} Y^{k*T} Y^{k*} B^{k*T}\|_2$
 7:     Update $X^{k+1} \in \text{prox}^+_{s_x}\left(X^{k*} - \frac{1}{L_X}\nabla_X H(X^{k*}, B^{k*}, Y^{k*})\right)$
 8:     Compute $L_B = \gamma_2 \cdot \|(X^{k+1})^T X^{k+1} B^{k*} Y^{k*T} Y^{k*}\|_2$
 9:     Update $B^{k+1} \in \text{prox}_{s_b}\left(B^{k*} - \frac{1}{L_B}\nabla_B H(X^{k+1}, B^{k*}, Y^{k*})\right)$
10:     Compute $L_Y = \gamma_3 \cdot \|(B^{k+1})^T (X^{k+1})^T X^{k+1} B^{k+1}\|_2$
11:     Update $Y^{k+1} \in \text{prox}^+_{s_y}\left(Y^{k*} - \frac{1}{L_Y}\nabla_Y H(X^{k+1}, B^{k+1}, Y^{k*})\right)$
12: end for
```