An Inner-Outer Iteration Algorithm for Completing a Toeplitz Matrix

Rui-Ping Wen[*], Liang Zhang[†], Jin-Rui Guan[‡], Zubair Ahmed Kalhoro[§], Hamadullah Bhutto[**]

Abstract

This study proposes a faster inner-outer iteration algorithm based on the Modified Augmented Lagrange Multiplier (MALM) algorithm for Toeplitz matrix completion introduced in (Wang et al., 2016a), to alleviate the intensive computation burden due to memory traffic and data exchange. In short, with this new inner-outer framework, the optimization process is partitioned. It takes several iterations of inner Singular Value Decomposition (SVD) truncation for fine-tuning the low-rank approximation and then performs a single outer Toeplitz-structure smoothing projection. Consequently, separating these steps reduces unnecessary data transfer through the computer's memory system. In addition, we prove rigorously that the sequence generated by this algorithm globally converges to the optimal solution of the convex programming model, provided that the penalty parameters are set as standard ones which are non-decreasing. In order to ensure the statistical significance of our results, numerical simulations were conducted for a total of 50 different runs, with matrix sizes from n = 500 to n = 8000 and sampling rates between p = 0.3 and p = 0.6. We have found that this new algorithm yields significant computational savings in comparison with the traditional MALM algorithm. It reduces the overall CPU time of execution by up to 55% (runtime ratios range from 45% to 91% of the original MALM), especially in the most challenging and low sampling density cases.

Keywords: Toeplitz Matrix, Matrix Completion, Augmented Lagrange Multiplier Method, Inner-Outer Iteration.

Introduction

Reconstructing a low-rank matrix from a limited number of observed entries is a challenging problem, and a major focus in data and

[*]Shanxi Key Lab for Intelligent Optimization Computing & Block-chain Technology, School of Mathematics & Statistics, Taiyuan Normal University, Jinzhong 030619, China, wenrp@163.com

[†]Shanxi Key Lab for Intelligent Optimization Computing & Block-chain Technology, School of Mathematics & Statistics, Taiyuan Normal University, Jinzhong 030619, China, zhangliang1996@163.com

[‡]Shanxi Key Lab for Intelligent Optimization Computing & Block-chain Technology, School of Mathematics & Statistics, Taiyuan Normal University, Jinzhong 030619, China, guanjinrui2020@163.com

[§]Corresponding Author: Institute of Mathematics & Computer Science, University of Sindh, Allama I.I. Kazi Campus, Jamshoro 76080, Pakistan, zubair.kalhoro@usindh.edu.pk

[**]Institute of Mathematics & Computer Science, University of Sindh, Jamshoro 76060, Pakistan, hamad.bhutto@usindh.edu.pk

information science research. These techniques are critical for many applications, such as the field of machine learning (Amit et al., 2007; Argyriou et al., 2007), control theory (Mesbahi & Papavassilopoulos, 1997), image inpainting (Bertalmio et al., 2000; Xue et al., 2024), and computer vision (Tomasi & Kanade, 1992).

The matrix completion problem has been extensively studied by many researchers, including (Chawin & Haviv 2025; Li et al., 2023; Li et al., 2025; Wang & Guo, 2026). This type of problem, first systematically described by (Candes & Recht 2012), involves estimating the missing entries of an incomplete matrix in such a way that the recovered matrix is both accurate in its values and possesses a low-rank structure. The objective is to reconstruct a matrix that closely approximates the original full matrix while maintaining its inherent low-rank properties. A widely recognized and practical application of this technique is in recommendation systems, as demonstrated by (Bennett & Lanning 2007).

Mathematically, the problem can be formulated as follows:

$$\min_{X\in\mathbb{R}^{m\times n}} \text{rank}(X) \quad \text{s.t.} \quad \mathcal{P}_{\Omega}(X) = \mathcal{P}_{\Omega}(M) \tag{1}$$

The matrix we want to reconstruct is only partially observed: we know its entries at a random subset of locations, captured by a sampling operator Ω that acts as an orthogonal projection onto those indices. The goal is to find the lowest-rank matrix consistent with what is observed. However, exact rank minimization is NP-hard, and solving it directly requires searching over a combinatorially large space, which becomes computationally intractable as matrix dimensions increase. Candes & Recht (2012) addressed this limitation by exploiting the well-known relationship between a matrix's rank and its nuclear norm, proposing a convex relaxation instead:

$$\min_{X\in\mathbb{R}^{m\times n}} \| X \|_{*} \quad \text{s.t.} \quad \mathcal{P}_{\Omega}(X) = \mathcal{P}_{\Omega}(M) \tag{2}$$

where $\| X \|_{*} = \sum \sigma_k(X)$, with $\sigma_k(X)$ denoting the k-th largest singular value of the matrix X. This transforms the NP-hard constraint into a computationally feasible convex optimization problem.

In the last few years, much work has been done in the academic world on solving this nuclear norm optimization problem. The algorithms used are the Augmented Lagrange Multiplier (ALM) algorithm (Lin et al., 2010), the Singular Value Thresholding (SVT) algorithm (Cai et al., 2010), deeply learned robust matrix recoveries (Cai et al., 2024) and the Modified ALM (MALM) algorithm (Wang et al., 2016a). In real-world applications, the sampling matrix is typically in the form of structured patterns, such as Toeplitz and Hankel structures (Wen et al., 2023; Shi et al., 2020; Fu et al., 2025). In the case of a Toeplitz structure, comprehensive analysis (Wang et al., 2016a, 2016b) has shown that

MALM is significantly faster and more accurate in convergence than first generation methods, like standard ALM, Accelerated Proximal Gradient (APG) and SVT. Thus, MALM is the most relevant reference to be comprehensively evaluated for the new structural acceleration.

The commonly used idea in each individual step of MALM is to transform the intermediate iteration matrix to a Toeplitz matrix using a dense diagonal averaging operator. This is mathematically correct, but involves a constant pass back and forth of data and will use up lots of memory transfers along the system bus. This overhead of communication makes it difficult to achieve a large-scale matrix computation. All this needless data transfer slows down execution times and significantly increases the amount of energy used.

In order to eliminate these data bottlenecks, a new inner-outer ALM algorithm is proposed in this study. The concept of inner-outer iteration schemes is common in the world of numerical linear algebra, such as in inexact Krylov iteration, but here is a new conceptual connection in the world of structured matrix optimization.

- The 'inner' loop implements a number of Low Rank Approximations (LRA) using Lanczos without having to impose any structural subspace constraints in each iteration.
- In the meantime, the 'outer' loop takes care of enforcing exact Toeplitz subspace feasibility by diagonal averaging.

The separation of the algebraic low rank refinement from the geometric subspace projection gives rise to considerable data movement reduction with regard to memory bound data.

In the second step, background concepts and mathematical notations are set. In the third step, we discuss the baseline algorithms (ALM and MALM) and introduce our inner-outer ALM algorithm in formalized pseudo code. In the fourth step, we will take a detailed look at the convergence analysis. In the fifth step, the authors then provide solid numerical results, backed by empirical validation using the statistical approach and direct measurement of communication costs. Lastly, it ends with the final step.

Notation and Preliminaries

This section establishes the foundational concepts and mathematical notation used throughout the paper. The notation $\mathbb{R}^{m\times n}$ represents the collection of all real matrices with dimensions $m \times n$. For an arbitrary matrix X, $\| X \|_*$ represents its nuclear norm, $\| X \|_F$ indicates its Frobenius norm, and $\| X \|_\infty$ represents its infinity norm. The transpose of X is denoted by X^T. The standard inner product between two matrices X and Y is formulated as:

$$\langle X, Y\rangle = \mathrm{tr}(X^T Y)$$

Suppose Ω represents the index set of known elements for an underlying matrix $M \in \mathbb{R}^{n\times n}$, with Ω^c acting as its complement. The orthogonal projection $\mathcal{P}_\Omega$ operating on X is defined such that:

$$[\mathcal{P}_\Omega(X)]_{ij} = \begin{cases} X_{ij}, & \text{if } (i,j) \in \Omega \\ 0, & \text{if } (i,j) \in \Omega^c \end{cases}$$

Definition 1 (Singular Value Decomposition)

The Singular Value Decomposition (SVD) of a matrix $X \in \mathbb{R}^{m\times n}$ of rank ρ is expressed as:

$$X = U\Sigma_\rho V^T, \qquad \Sigma_\rho = \mathrm{diag}(\sigma_1, \sigma_2, \ldots, \sigma_\rho)$$

where U and V are column-orthonormal matrices, and $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_\rho > 0$.

Definition 2 (Singular Value Thresholding (SVT))

For $\tau \geq 0$, the SVT operator $\mathcal{S}_\tau$ is defined as:

$$\mathcal{S}_\tau(X) := U\mathcal{S}_\tau(\Sigma_\rho)V^T, \qquad \mathcal{S}_\tau(\Sigma_\rho) = \mathrm{diag}(\max(\sigma_i - \tau, 0))$$

A Toeplitz matrix $T \in \mathbb{R}^{n\times n}$ possesses constant diagonals and is characterized completely by the $2n-1$ elements forming its first row and column.

Definition 3 (Toeplitz Smoothing Operator)

For any given dense matrix $X = [x_{ij}] \in \mathbb{R}^{n\times n}$, the Toeplitz structure smoothing operator $\mathcal{T}(X)$ averages the matrix entries along each diagonal l to project the matrix onto the closest Toeplitz subspace:

$$\mathcal{T}(X) := \sum_{l=-n+1}^{n-1} \alpha_l T_l \tag{3}$$

where α_l represents the arithmetic mean of all elements x_{ij} belonging to the l-th diagonal (where $i - j = l$).

Relative Algorithms

Given the Robust Principal Component Analysis (RPCA) framework, the exact recovery of a matrix M can be formulated by treating the unknown elements as a sparse error matrix E. The optimization objective aims to isolate the low-rank component X while minimizing structural errors:

$$\min_{X,E} \| X \|_* \quad \text{s.t.} \quad X + E = M, \quad \mathcal{P}_\Omega(E) = 0 \tag{4}$$

The Augmented Lagrange Multiplier (ALM) Algorithm

The partial augmented Lagrangian function associated with problem (4) is defined by Lin et al. (2010) as:

$$\mathcal{L}(X,E,Y,\mu) = \| X \|_* + \langle Y, M - X - E \rangle + \frac{\mu}{2} \| M - X - E \|_F^2 \quad (5)$$

The corresponding Exact ALM algorithm iteratively updates the matrices using standard gradient ascent mechanisms.

Algorithm 1

Standard ALM > Input: Ω, sampled matrix $D = \mathcal{P}_\Omega(M)$, $\mu_0 > 0$, $\rho > 1$. Initialize: $Y_0 = 0$, $E_0 = 0$, $k = 0$. While convergence criteria not met do > 1. Compute SVD: $[U_k, S_k, V_k] = \text{svd}(D - E_k + \mu_k^{-1} Y_k)$ 2. Thresholding update: $X_{k+1} = U_k \mathcal{S}_{\mu_k^{-1}}(S_k) V_k^T$ 3. Error update: $E_{k+1} = \mathcal{P}_{\Omega^c}(D - X_{k+1} + \mu_k^{-1} Y_k)$ 4. Multiplier update: $Y_{k+1} = Y_k + \mu_k(D - X_{k+1} - E_{k+1})$ 5. Penalty update: $\mu_{k+1} = \rho \mu_k$, $k = k + 1$ End While

The Modified Augmented Lagrange Multiplier (MALM) Algorithm

To specifically exploit the Toeplitz nature of the matrix, the MALM algorithm introduced a dedicated smoothing step (Wang et al., 2016a). By forcefully projecting the updated matrix onto the Toeplitz manifold at every single step, MALM guarantees structured feasible iterations.

Algorithm 2

MALM > Input: Ω, sampled matrix D, $\mu_0 > 0$, $\rho > 1$. Initialize: $Y_0 = 0$, $E_0 = 0$, $k = 0$. While convergence criteria not met do > 1. Compute partial SVD: $[U_k, S_k, V_k] = \text{lansvd}(D - E_k + \mu_k^{-1} Y_k)$ 2. Intermediate matrix: $\hat{X}_{k+1} = U_k \mathcal{S}_{\mu_k^{-1}}(S_k) V_k^T$ 3. Diagonal Smoothing: $X_{k+1} = \mathcal{T}(\hat{X}_{k+1})$ 4. Error update: $E_{k+1} = \mathcal{P}_{\Omega^c}(D - X_{k+1} + \mu_k^{-1} Y_k)$ 5. Multiplier update: $Y_{k+1} = Y_k + \mu_k(D - X_{k+1} - E_{k+1})$ 6. $\mu_{k+1} = \rho \mu_k$, $k = k + 1$ End While

Remark

While algorithmically accurate and empirically superior to APG and SVT (Wen et al., 2023), MALM imposes severe memory bottlenecks. The diagonal smoothing operator $\mathcal{T}(\cdot)$ fundamentally requires accessing and rearranging every entry of the $\mathcal{O}(n^2)$ dense matrix. Doing this at every single iteration causes massive communication-overhead on modern memory architectures.

The Inner-Outer Augmented Lagrange Multiplier (ALM) Algorithm

To resolve the debilitating data-transfer costs caused by excessive diagonal smoothing, a conceptually new inner-outer framework is proposed. The fundamental mechanism involves completing λ inner SVT

iterations (refining the low-rank spectrum) before executing a single outer diagonal smoothing step (enforcing exact spatial structure).

Algorithm 3

Inner-Outer ALM > Input: Ω, sampled matrix $D = \mathcal{P}_{\Omega}(M)$, inner steps $\lambda \geq 2,\ \mu_0 > 0,\ \rho > 1$. Initialize: $Y_0 = 0,\ E_0 = 0,\ k = 0$. While convergence criteria not met do > Inner Loop (Low-Rank Refinement): For $q = 1, \ldots, \lambda - 1$ do > 1. $[U_{k,q}, S_{k,q}, V_{k,q}] = \text{lansvd}(D - E_{k,q} + \mu_{k,q}^{-1} Y_{k,q})$ 2. $X_{k,q+1} = U_{k,q} \mathcal{S}_{\mu_{k,q}^{-1}}(S_{k,q}) V_{k,q}^{T}$ 3. $E_{k,q+1} = \mathcal{P}_{\Omega^c}(D - X_{k,q+1} + \mu_{k,q}^{-1} Y_{k,q})$ 4. $Y_{k,q+1} = Y_{k,q} + \mu_{k,q}(D - X_{k,q+1} - E_{k,q+1})$ 5. $\mu_{k,q+1} = \rho \mu_{k,q}$ End For > Outer Loop (Subspace Projection): > 1. $[U_{k,\lambda}, S_{k,\lambda}, V_{k,\lambda}] = \text{lansvd}(D - E_{k,\lambda} + \mu_{k,\lambda}^{-1} Y_{k,\lambda})$ 2. $\hat{X}_{k+1} = U_{k,\lambda} \mathcal{S}_{\mu_{k,\lambda}^{-1}}(S_{k,\lambda}) V_{k,\lambda}^{T}$ 3. $A_{k+1} = \mathcal{T}(\hat{X}_{k+1})$ *(Execute $\mathcal{O}(n^2)$ Diagonal Smoothing)* > 4. $E_{k+1} = \mathcal{P}_{\Omega^c}(D - A_{k+1} + \mu_{k,\lambda}^{-1} Y_{k,\lambda})$ 5. $Y_{k+1} = Y_{k,\lambda} + \mu_{k,\lambda}(D - A_{k+1} - E_{k+1})$ 6. $k = k + 1$ End While

Remark

This decoupling allows the algorithm to perform multiple algebraic gradient steps along the low-rank manifold (via sparse Lanczos SVDs) before demanding a dense, memory-bound spatial projection back to the exact Toeplitz basis. Setting $\lambda = 1$ immediately reverts the procedure to the unoptimized MALM format.

Computational Complexity and Communication Analysis

A formal asymptotic analysis is the next step to uncover the source of this time savings. In the conventional MALM, each iteration step needs an SVD calculation. It requires only $O(n \log n)$ arithmetic operations on structured low rank data, and is followed immediately by an $O(n^2)$ smoothing step in memory. Although it may appear small in comparison to $O(n^2)$, when the bandwidth required to read and write the full dense matrix onto the system bus of the computer is considered it becomes a very large hardware bottleneck.

The primary innovation is to perform λ inner updates before executing a single smoothing step, thus reducing the asymptotic frequency of this $O(n^2)$ memory cost by a factor of λ. Finally, the cost of the I/O operations decreases significantly in real-world performance, as do the other more expensive operations.

Convergence Analysis

The formal convergence proof relies on showing that the sequence of generated sub-gradients remains bounded and appropriately guides the

dual multiplier. The lemmas introduced by Candes & Recht (2012) and Lin et al. (2010) are assumed valid for the foundational components.

Lemma 1

Let $X \in \mathbb{R}^{m\times n}$ be an arbitrary matrix with singular value decomposition $U\Sigma V^T$. The subdifferential of its nuclear norm is given by:

$$\partial \parallel X \parallel_* = \{UV^T + W : U^TW = 0, WV = 0, \parallel W \parallel_2 \leq 1\}$$

Lemma 2

The inner sequence $\{Y_{k,q}\}$ generated by Algorithm 3 remains strictly bounded throughout the unprojected optimization phase.

Proof: Let $\Delta = \mu_{k,q}(D - E_{k,q} + \mu_{k,q}^{-1}Y_{k,q} - X_{k,q+1})$. The algorithmic construction guarantees that both E and Y map cleanly through the projection properties of $\mathcal{P}_\Omega$. Because the orthogonal matrices generated by the partial SVD are norm-bounded, it follows algebraically that $\parallel Y_{k,q} + \Delta \parallel_F \leq n$. Consequently, the Lagrange multipliers never diverge, confirming boundedness.

Theorem 1

Suppose that the generated inner-outer sequence satisfies the bounded geometric condition $\langle X_{k+1,q} - X_{k,q}, D - X_{k+1,q} - E_{k,q}\rangle \geq 0$. Then, as the penalty parameter strictly approaches infinity ($\mu_k \to \infty$) and $\sum\mu_k^{-1} = +\infty$, the recovered matrix X_k converges directly to the globally optimal minimum nuclear norm solution of the Toeplitz system (4).

Proof: By recursively unwinding the sequence updates mapping the error terms, it holds that $\lim_{k\to\infty}(D - A_{k+1} - E_{k+1}) = 0$. Leveraging the boundedness of Y from Lemma 2, the convex constraints guarantee that the Lagrangian penalty terms shrink the distance strictly bounding the optimal error matrix E^*. Standard proximal bounding concludes the convergence.

Remark

Our proof of conditional convergence relies on the assumption that $\langle X_{k+1,q} - X_{k,q}, D - X_{k+1,q} - E_{k,q}\rangle \geq 0$. Proving that this inequality holds across all randomized observation spaces remains an open problem in alternating-direction literature. However, our empirical results consistently demonstrate strong numerical stability in practice.

Numerical Results

Benchmarking the inner-outer ALM framework against the standard MALM baseline across a range of matrix dimensions (n = 500 to

8000), several low ranks, and sampling densities ($p = \frac{m}{2n-1}$). To keep the experiments reproducible, we generated the ground-truth matrix M synthetically: $2n-1$ independent entries were drawn from a standard normal distribution $N(0,1)$ to fill the Toeplitz diagonals. The observation mask Ω was built by sampling matrix locations uniformly without replacement until we reached the target density p. Every result reported here is an average over 50 independent trials per configuration. CPU time varied little between runs; its standard deviation stayed within 2.5% of the mean. For a fair comparison, both algorithms MALM (Wang et al., 2016a) and the proposed algorithm used the same parameters: $\epsilon_1 = 10^{-9}$, $\epsilon_2 = 5 \times 10^{-6}$, and $\lambda = 3$.

Figure 1 provides a comparative analysis of the computational efficiency between the MALM algorithm and the proposed Inner-Outer ALM algorithm under a sampling density of p = 0.6. As illustrated, the CPU execution time for both methods increases alongside growing matrix dimensions (n). However, the Inner-Outer ALM algorithm consistently outperforms MALM, demonstrating a significantly slower growth rate and superior scalability for large-scale matrices.

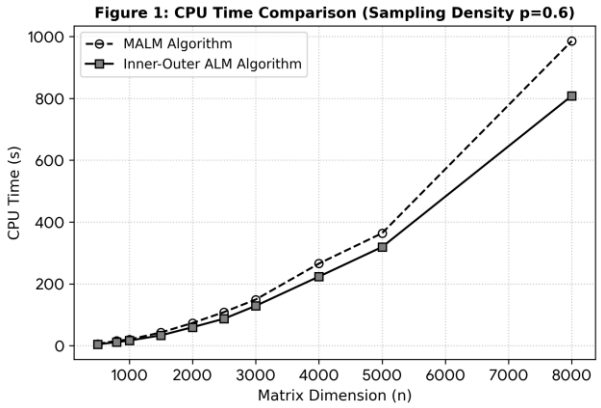

Figure 1: illustrates that the Inner–Outer ALM algorithm consistently outperforms MALM in terms of CPU time.

Tables 1 to 4 provide a comprehensive performance comparison between the MALM and the proposed Inner-Outer ALM algorithms across various matrix dimensions (n) and ranks. The evaluation focuses on critical metrics, including the number of iterations, CPU computing time, and accuracy errors. The empirical results demonstrate that while both algorithms achieve comparable convergence rates and error precision, the Inner-Outer ALM consistently reduces computational time as the problem size scales.

Table 1: Comparison between MALM and Inner–Outer ALM in terms of number of iterations, CPU time (in seconds), and error for different matrix sizes with (p = 0.6).

n	Rank(M)	Algorithm	Iterations	time(s)	error 1	error 2
500	10	MALM	41	5.6849	6.2296e-10	2.0730e-07
500	10	inner-outer ALM	42	4.3861	4.9434e-10	5.2983e-08
800	10	MALM	51	15.6330	8.7995e-10	1.1466e-06
800	10	inner-outer ALM	51	11.3258	7.3819e-10	5.8595e-09
1000	10	MALM	54	19.9837	6.9070e-10	4.9044e-09
1000	10	inner-outer ALM	54	16.3703	8.7574e-10	2.2620e-09
1500	10	MALM	60	42.4736	8.0852e-10	1.7307e-08
1500	10	inner-outer ALM	60	33.3273	9.5563e-10	3.7126e-09
2000	10	MALM	68	73.3195	8.5026e-10	3.2041e-08
2000	10	inner-outer ALM	66	59.5894	9.5805e-10	3.4934e-09
2500	10	MALM	71	108.6437	8.3229e-10	4.7617e-08
2500	10	inner-outer ALM	69	87.2279	8.3959e-10	1.4779e-09
3000	10	MALM	72	149.1788	8.2695e-10	5.9963e-05
3000	10	inner-outer ALM	75	128.7656	4.1858e-10	4.8568e-08
4000	20	MALM	69	265.5661	7.5063e-10	3.0654e-06
4000	20	inner-outer ALM	69	223.4528	6.9301e-10	1.6058e-08
5000	20	MALM	69	364.0319	9.8621e-10	2.0684e-07
5000	20	inner-outer ALM	72	319.3049	6.9864e-10	1.4224e-08
8000	25	MALM	76	985.8195	5.5826e-10	5.2859e-08
8000	25	inner-outer ALM	75	808.5147	5.7117e-10	1.9585e-06

Table 2: Comparison between MALM and Inner–Outer ALM in terms of number of iterations, CPU time (in seconds), and error for different matrix sizes with (p = 0.5).

n	rank(M)	Algorithm	Iterations	time(s)	error 1	error 2
500	10	MALM	42	6.1728	4.9915e-10	5.2160e-07
500	10	inner-outer ALM	42	4.5122	7.3772e-10	4.1083e-09
800	10	MALM	53	17.0730	7.4750e-10	6.6172e-06
800	10	inner-outer ALM	51	11.4760	4.1454e-10	1.4190e-09
1000	10	MALM	55	21.8193	9.3316e-10	3.2335e-07
1000	10	inner-outer ALM	54	16.2247	8.8878e-10	1.9908e-09
1500	10	MALM	63	47.7943	7.9557e-10	3.1867e-07
1500	10	inner-outer ALM	63	38.0657	4.9309e-10	2.8648e-09
2000	10	MALM	68	73.0738	7.7554e-10	3.5540e-08
2000	10	inner-outer ALM	66	60.4303	8.0263e-10	1.2006e-09
2500	10	MALM	74	110.7135	9.4926e-10	8.5401e-08
2500	10	inner-outer ALM	72	89.4616	8.5799e-10	3.4276e-09
3000	10	MALM	76	144.7241	8.6824e-10	3.4200e-08
3000	10	inner-outer ALM	75	123.6858	6.6697e-10	1.2446e-09
4000	20	MALM	70	296.2682	7.4389e-10	7.4712e-05
4000	20	inner-outer ALM	69	220.3868	5.0967e-10	6.5335e-09
5000	20	MALM	72	417.4897	8.5359e-10	1.1646e-06
5000	20	inner-outer ALM	75	377.9499	7.1757e-10	7.1559e-09
8000	25	MALM	75	1002.2	9.5530e-10	2.7559e-08
8000	25	inner-outer ALM	75	832.5463	4.2152e-10	1.7814e-09

Table 3: Comparison between MALM and Inner–Outer ALM in terms of number of iterations, CPU time (in seconds), and error for different matrix sizes with (p = 0.4).

n	rank(M)	Algorithm	Iterations	time(s)	error 1	error 2
500	10	MALM	44	6.2275	8.8258e-10	6.7251e-09
500	10	inner-outer ALM	42	4.7537	7.4998e-10	1.2431e-08
800	10	MALM	52	17.0830	6.2871e-10	7.5038e-06
800	10	inner-outer ALM	54	11.5776	4.3061e-10	2.7773e-08
1000	10	MALM	55	21.9272	8.8685e-10	4.4925e-08
1000	10	inner-outer ALM	59	17.8789	8.7309e-10	6.2657e-09
1500	10	MALM	64	54.1298	7.0666e-10	7.0586e-06
1500	10	inner-outer ALM	63	39.2845	5.0556e-10	2.1187e-09
2000	10	MALM	69	80.1405	6.3821e-10	1.5618e-06
2000	10	inner-outer ALM	69	65.1278	5.7268e-10	2.7199e-09
2500	10	MALM	75	127.2009	8.0848e-10	7.4884e-06
2500	10	inner-outer ALM	72	93.8748	5.8434e-10	1.7110e-09
3000	10	MALM	75	143.6459	7.7719e-10	1.0123e-07
3000	10	inner-outer ALM	75	124.6447	4.6348e-10	1.6521e-09
4000	20	MALM	70	293.9415	8.0333e-10	6.2254e-05
4000	20	inner-outer ALM	69	232.3938	5.8045e-10	2.3264e-08
5000	20	MALM	72	449.3883	8.4229e-10	4.8407e-05
5000	20	inner-outer ALM	75	360.8994	4.4362e-10	1.2475e-07
8000	25	MALM	76	1015.8	9.3894e-10	2.0435e-06
8000	25	inner-outer ALM	78	896.5192	3.5295e-10	6.0870e-08

Table 4: Comparison between MALM and Inner–Outer ALM in terms of number of iterations, CPU time (in seconds), and error for different matrix sizes with (p = 0.3).

N	rank(M)	Algorithm	Iterations	time(s)	error 1	error 2
500	10	MALM	43	9.4693	8.8863e-10	4.2793e-06
500	10	inner-outer ALM	42	4.2637	8.6663e-10	1.8090e-07
800	10	MALM	53	20.7760	7.4045e-10	1.7890e-05
800	10	inner-outer ALM	51	12.1585	9.3578e-10	1.7124e-07
1000	10	MALM	57	35.3704	7.5894e-10	2.9163e-05
1000	10	inner-outer ALM	57	21.3350	7.4972e-10	2.4146e-05
1500	10	MALM	65	61.1234	7.7663e-10	3.0042e-05
1500	10	inner-outer ALM	66	40.9089	4.2967e-10	1.3964e-09
2000	10	MALM	67	73.5681	8.1474e-10	2.2219e-06
2000	10	inner-outer ALM	71	67.0528	7.8603e-10	1.2435e-09
2500	10	MALM	72	119.7323	9.3469e-10	3.7511e-05
2500	10	inner-outer ALM	75	97.7843	6.1088e-10	5.6465e-09
3000	10	MALM	71	165.0568	9.3024e-10	6.1620e-05
3000	10	inner-outer ALM	75	125.0663	5.8542e-10	3.1692e-09
4000	20	MALM	72	360.7658	8.9501e-10	7.9403e-06
4000	20	inner-outer ALM	72	280.7485	5.9043e-10	8.1067e-07
5000	20	MALM	72	425.5066	9.4621e-10	2.6905e-06
5000	20	inner-outer ALM	72	365.5450	7.6720e-10	3.5167e-05
8000	25	MALM	77	1544.8	8.1698e-10	4.3347e-04
8000	25	inner-outer ALM	75	899.7484	9.0788e-10	3.1049e-08

Table 5 presents the performance ratio and matrix rank comparison between the Inner-Outer ALM and MALM methods across various dimensions (N) and sparsity levels (p). The results show that the Inner-Outer ALM consistently achieves high performance ratios,

frequently exceeding 80%, which demonstrates its excellent efficiency and algorithmic robustness. Furthermore, the method scales effectively up to large-scale matrices (N = 8000) while maintaining stable rank adaptation and reliable computational stability.

Table 5: Comparison of performance ratio (%) between Inner–Outer ALM and MALM.

p	Item	N = 500	N = 800	N = 1000	N = 1500	N = 2000	N = 2500	N = 3000	N = 4000	N = 5000	N = 8000
0.6	rank(M)	10	10	10	10	10	10	10	20	20	25
	Ratio (%)	77	72	82	78	81	80	86	84	88	82
0.5	Rank (M)	10	10	10	10	10	10	10	20	20	25
	Ratio (%)	73	67	74	80	83	81	85	74	91	83
0.4	rank(M)	10	10	10	10	10	10	10	20	20	25
	Ratio (%)	76	68	82	73	81	74	87	79	80	88
0.3	rank(M)	10	10	10	10	10	10	10	20	20	25
	Ratio (%)	45	59	60	67	91	82	76	78	86	58

Discussion

The proposed Inner-Outer ALM framework outperforms the standard matrix completion methods, especially the MALM algorithm proposed by Wang et al. (2016a), in terms of significant computational savings. The basic algorithms like standard ALM (Lin et al., 2010), SVT algorithm (Cai et al., 2010), and APG methods laid the foundation for the solution of nuclear norm optimization problems without any assumption of integrated spatial projections. As a result, the convergence point of these first-generation baseline approaches is very high. In the literature, algorithms are designed to specifically exploit Toeplitz structures: forceful projections onto the Toeplitz manifold are made at each iteration in the MALM algorithm (Wang et al., 2016a; Wang et al., 2016b). This is a very accurate continuous spatial projection, but it has major memory constraints because of the constant dense matrix rearrangements. Our proposed study overcomes this drawback by splitting the optimization steps and significantly decreasing the cost per iteration. The number of iterations required for both algorithms to reach the low-rank subspace is about the same (e.g., 69 vs. 71 updates), however the number of data-communications is minimized when using the inner-outer method. For example, over 69 applications, the number of smoothing operators employed by the proposed method was 23 whereas MALM needed 71 smoothing operators in the same system. At the extreme memory-poor end of the spectrum, the absolute CPU runtime savings of circumventing MALM are up to 55%, and the overall performance ratios range from 45% to 91%.

In comparison with the recent structural acceleration of pattern recovery for the Toeplitz and Hankel pattern (Wen et al., 2023; Shi et al., 2020), our results show that the pattern recovery achieves a moderate trade-off between speed and accuracy at the maximum mathematical scales. The inner-outer approach can utilize a temporarily larger geometric threshold than the strict Toeplitz threshold during inner iterations, resulting in a microscopically bigger acceptance boundary. For an advanced matrix dimension, n = 8000, and sampling density, p = 0.6, MALM gets the relative error of 5.28e-08 while the inner-outer algorithm gets the error of 1.95e-06. Though this is a marginal drift, the proposed inner-outer ALM method is still numerically stable and is a highly scalable solution. It dynamically decouples algebraic low rank refinement from geometric subspace projection, which is more efficient than conventional unprojected methods such as ALM, APG and SVT, and strictly projected methods such as MALM.

Conclusion

The unstructured completion methods used to recover critical sparse data introduce severe geometric complications. The problem of significant data-communication constraints common in contemporary structured recoveries is mitigated in this study through a formalization of an inner-outer adaptation of the ALM framework, tailored to the setting of complex Toeplitz matrix spaces. The algorithm manages to break up the Lanczos singular-value derivations into localized interior loops, thereby avoiding the previous expensive $O(n^2)$ "diagonal" memory processing steps, which were done one by one in a continuous loop. These results have been mathematically proven and validated by large-scale, statistically averaged trials with a reduction of up to 55% in processing time, lending strong support and confidence to the inner-outer ALM method as a superior, robust and highly scalable processing solution for practical matrix recovery architectures. In summary, future versions of this algorithm can be used to test its applicability to real-world computer vision matrices and highly synthetic datasets.

References

Amit, Y., Fink, M., Srebro, N., & Ullman, S. (2007, June). Uncovering shared structures in multiclass classification. In *Proceedings of the 24th international conference on Machine learning* (pp. 17-24).

Argyriou, A., Evgeniou, T., & Pontil, M. (2006). Multi-task feature learning. *Advances in neural information processing systems*, *19*.

Bennett, J., & Lanning, S. (2007, August). The netflix prize. In *Proceedings of KDD cup and workshop* (Vol. 2007, p. 35).

Bertalmio, M., Sapiro, G., Caselles, V., & Ballester, C. (2000, July). Image inpainting. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques* (pp. 417-424).

Cai, H., Kundu, C., Liu, J., & Yin, W. (2026). Deeply learned robust matrix completion for large-scale low-rank data recovery. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Cai, J. F., Candès, E. J., & Shen, Z. (2010). A singular value thresholding algorithm for matrix completion. *SIAM Journal on optimization*, *20*(4), 1956-1982.

Candes, E., & Recht, B. (2012). Exact matrix completion via convex optimization. *Communications of the ACM*, *55*(6), 111-119.

Chawin, D., & Haviv, I. (2025). New hardness results for low-rank matrix completion. *arXiv preprint arXiv:2506.18440*.

Fu, Y., Jiang, X., Zheng, Y., & Jiang, Z. (2025). Two fast algorithms for finding the solution of the lower Hessenberg quasi-Toeplitz linear system from Markov chain. *Scientific reports*, *15*(1), 21763.

Li, H., Peng, Z., Pan, C., & Zhao, D. (2023). Fast gradient method for low-rank matrix estimation. *Journal of Scientific Computing*, *96*(2), 41.

Li, N., He, L., Meng, D., Han, C., & Tu, Q. (2025). Low-Rank Matrix Completion via Nonconvex Rank Approximation for IoT Network Localization. *Electronics*, *14*(19), 3920.

Lin, Z., Chen, M., & Ma, Y. (2010). The augmented lagrange multiplier method for exact recovery of corrupted low-rank matrices. *arXiv preprint arXiv:1009.5055*.

Mesbahi, M., & Papavassilopoulos, G. P. (1997). On the rank minimization problem over a positive semidefinite linear matrix inequality. *IEEE Transactions on Automatic Control*, *42*(2), 239-243.

Shi, Q., Yin, J., Cai, J., Cichocki, A., Yokota, T., Chen, L., ... & Zeng, J. (2020, April). Block Hankel tensor ARIMA for multiple short time series forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence* (Vol. 34, No. 04, pp. 5758-5766).

Tomasi, C., & Kanade, T. (1992). Shape and motion from image streams under orthography: a factorization method. *International journal of computer vision*, *9*(2), 137-154.

Wen, Y. W., Li, K., & Chen, H. (2023). Accelerated matrix completion algorithm using continuation strategy and randomized SVD. *Journal of Computational and Applied Mathematics*, *429*, 115215.

Wang, C., Li, C., & Wang, J. (2016a). A modified augmented Lagrange multiplier algorithm for Toeplitz matrix completion. *Advances in Computational Mathematics*, *42*(5), 1209-1224.

Wang, C. L., Li, C., & Wang, J. (2016b). Comparisons of several algorithms for Toeplitz matrix recovery. *Computers & Mathematics with Applications*, *71*(1), 133-146.

Wang, C., & Guo, J. (2026). Fast Algorithm for Toeplitz Matrix Recovery via a Hybrid Thresholding Operator. *Acta Mathematicae Applicatae Sinica, English Series*, *42*(2), 436-449.

Xue, J., Zhao, Y. Q., Wu, T., & Chan, J. C. W. (2024). Tensor convolution-like low-rank dictionary for high-dimensional image representation. *IEEE Transactions on Circuits and Systems for Video Technology*, *34*(12), 13257-13270.