A Matrix Splitting Method for Composite Function Minimization

Ganzhao Yuan[1,2], Wei-Shi Zheng[2,3], Bernard Ghanem[1]
[1]King Abdullah University of Science and Technology (KAUST), Saudi Arabia
[2]School of Data and Computer Science, Sun Yat-sen University (SYSU), China
[3]Key Laboratory of Machine Intelligence and Advanced Computing (Sun Yat-sen University), Ministry of Education, China
yuanganzhao@gmail.com, wszheng@ieee.org, bernard.ghanem@kaust.edu.sa

Abstract

Composite function minimization captures a wide spectrum of applications in both computer vision and machine learning. It includes bound constrained optimization and cardinality regularized optimization as special cases. This paper proposes and analyzes a new Matrix Splitting Method (MSM) for minimizing composite functions. It can be viewed as a generalization of the classical Gauss-Seidel method and the Successive Over-Relaxation method for solving linear systems in the literature. Incorporating a new Gaussian elimination procedure, the matrix splitting method achieves state-of-the-art performance. For convex problems, we establish the global convergence, convergence rate, and iteration complexity of MSM, while for non-convex problems, we prove its global convergence. Finally, we validate the performance of our matrix splitting method on two particular applications: nonnegative matrix factorization and cardinality regularized sparse coding. Extensive experiments show that our method outperforms existing composite function minimization techniques in term of both efficiency and efficacy.

1. Introduction

In this paper, we focus on the following composite function minimization problem:

$$\min_{\mathbf{x}} \; f(\mathbf{x}) \triangleq q(\mathbf{x}) + h(\mathbf{x});\; q(\mathbf{x}) = \tfrac{1}{2}\mathbf{x}^T\mathbf{A}\mathbf{x} + \mathbf{x}^T\mathbf{b} \qquad (1)$$

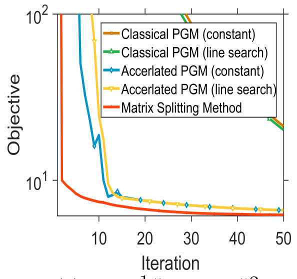
where $\mathbf{x} \in \mathbb{R}^n$, $\mathbf{b} \in \mathbb{R}^n$, $\mathbf{A} \in \mathbb{R}^{n\times n}$ is a positive semidefinite matrix, $h(\mathbf{x})$ is a piecewise separable function (*i.e.* $h(\mathbf{x}) = \sum_{i=1}^n h(\mathbf{x}_i)$) but not necessarily convex. Typical examples of $h(\mathbf{x})$ include the bound constrained function and the ℓ_0 and ℓ_1 norm functions.

The optimization in (1) is flexible enough to model a variety of applications of interest in both computer vision and machine learning, including compressive sensing [7], nonnegative matrix factorization [16, 18, 9], sparse coding [17, 1, 2, 29], support vector machine [11], logistic regression [38], subspace clustering [8], to name a few. Although we only focus on the quadratic function $q(\cdot)$, our method can be extended to handle general composite functions as well, by considering a typical Newton approximation of the objective [35, 40].

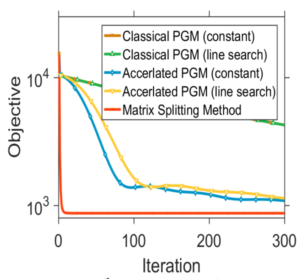
The most popular method for solving (1) is perhaps the proximal gradient method [25, 3]. It considers a fixed-point proximal iterative procedure $\mathbf{x}^{k+1} = \text{prox}_{\gamma h}(\mathbf{x}^k - \gamma\nabla q(\mathbf{x}^k))$ based on the current gradient $\nabla q(\mathbf{x}^k)$. Here the proximal operator $\text{prox}_{\tilde{h}}(\mathbf{a}) = \arg\min_{\mathbf{x}} \frac{1}{2}\|\mathbf{x}-\mathbf{a}\|_2^2 + \tilde{h}(\mathbf{x})$ can often be evaluated analytically, $\gamma = 1/L$ is the step size with L being the local (or global) Lipschitz constant. It is guaranteed to decrease the objective at a rate of $\mathcal{O}(L/k)$, where k is the iteration number. The accelerated proximal gradient method can further boost the rate to $\mathcal{O}(L/k^2)$. Tighter estimates of the local Lipschitz constant leads to better convergence rate, but it scarifies additional computation overhead to compute L. Our method is also a fixed-point iterative method, but it does not rely on a sparse eigenvalue solver or line search backtracking to compute such a Lipschitz constant, and it can exploit the specified structure of the quadratic Hessian matrix $\mathbf{A}$.

The proposed method is essentially a generalization of the classical Gauss-Seidel (GS) method and Successive Over-Relaxation (SOR) method [6, 31]. In numerical linear algebra, the Gauss-Seidel method, also known as the successive displacement method, is a fast iterative method for solving a linear system of equations. It works by solving a sequence of triangular matrix equations. The method of SOR is a variant of the GS method and it often leads to faster convergence. Similar iterative methods for solving linear systems include the Jacobi method and symmetric SOR. Our proposed method can solve versatile composite function minimization problems, while inheriting the efficiency of modern linear algebra techniques.

Our method is closely related to coordinate gradient descent and its variants such as randomized coordinate descent [11, 28], cyclic coordinate descent [32], block coor-



(a) $\min_{\mathbf{x}\geq\mathbf{0}} \frac{1}{2}\|\mathbf{Cx}-\mathbf{d}\|_2^2$



(b) $\min_{\mathbf{x}} \frac{1}{2}\|\mathbf{Cx}-\mathbf{d}\|_2^2 + \|\mathbf{x}\|_0$

Figure 1: Convergence behavior for solving (a) convex nonnegative least squares and (b) nonconvex ℓ_0 norm sparse least squares. We generate $\mathbf{C} \in \mathbb{R}^{200\times 1000}$ and $\mathbf{d} \in \mathbb{R}^{200}$ from a (0-1) uniform distribution. All methods share the same initial point. Our matrix splitting method significantly outperforms existing popular proximal gradient methods [3, 25] such as classical Proximal Gradient Method (PGM) with constant step size, classical PGM with line search, accelerated PGM with constant step size and accelerated PGM with line search. Note that all the methods have the same computational complexity for one iteration.

dinate descent [24, 4, 10], randomized block coordinate descent [30, 22], accelerated randomized coordinate descent [24, 19, 21] and others [20, 13, 42]. However, all these work are based on gradient-descent type iterations and a constant Lipschitz step size. They work by solving a first-order majorization/surrogate function via closed form updates. Their algorithm design and convergence result cannot be applied here. In contrast, our proposed method does not rely on computing the Lipschicz constant step size, yet it adopts a triangle matrix factorization strategy, where the triangle subproblem can be solved by an alternating cyclic coordinate strategy.

Contributions. **(a)** We propose a new Matrix Splitting Method (MSM) for composite function minimization. **(b)** For convex problems, we establish the global convergence, convergence rate, and iteration complexity of MSM, while for non-convex problems, we prove its global convergence. **(c)** Our experiments on nonnegative matrix factorization and sparse coding show that MSM achieves state-of-the-art performance. Before proceeding, we present a running example in Figure 1 to show the performance of our proposed method, as compared with existing ones.

Notation. We use lowercase and uppercase boldfaced letters to denote real vectors and matrices respectively. The Euclidean inner product between $\mathbf{x}$ and $\mathbf{y}$ is denoted by $\langle \mathbf{x}, \mathbf{y}\rangle$ or $\mathbf{x}^T\mathbf{y}$. We denote $\|\mathbf{x}\| = \|\mathbf{x}\|_2 = \sqrt{\langle \mathbf{x}, \mathbf{x}\rangle}$, and $\|\mathbf{C}\|$ as the spectral norm (*i.e.* the largest singular value) of $\mathbf{C}$. We denote the i^{th} element of vector $\mathbf{x}$ as $\mathbf{x}_i$ and the $(i,j)^{\text{th}}$ element of matrix $\mathbf{C}$ as $\mathbf{C}_{i,j}$. $diag(\mathbf{D}) \in \mathbb{R}^n$ is a column vector formed from the main diagonal of $\mathbf{D} \in \mathbb{R}^{n\times n}$. $\mathbf{C} \succeq 0$ indicates that the matrix $\mathbf{C} \in \mathbb{R}^{n\times n}$ is positive semidefinite (not necessarily symmetric)[1]. Finally, we denote $\mathbf{D}$ as a diagonal matrix of $\mathbf{A}$ and $\mathbf{L}$ as a strictly lower triangle matrix of $\mathbf{A}$ [2]. Thus, we have $\mathbf{A} = \mathbf{L} + \mathbf{D} + \mathbf{L}^T$.

2. Proposed Matrix Splitting Method

In this section, we present our proposed matrix splitting method for solving (1). Throughout this section, we assume that $h(\mathbf{x})$ is convex and postpone the discussion for nonconvex $h(\mathbf{x})$ to Section 3.

Our solution algorithm is derived from a fixed-point iterative method based on the first-order optimal condition of (1). It is not hard to validate that a solution $\mathbf{x}$ is the optimal solution of (1) if and only if $\mathbf{x}$ satisfies the following nonlinear equation ("$\triangleq$" means define):

$$\mathbf{0} \in \partial f(\mathbf{x}) \triangleq \mathbf{Ax} + \mathbf{b} + \partial h(\mathbf{x}) \tag{2}$$

where $\partial h(\mathbf{x})$ is the sub-gradient of $h(\cdot)$ in $\mathbf{x}$. In numerical analysis, a point $\mathbf{x}$ is called a fixed point if it satisfies the equation $\mathbf{x} \in \mathcal{T}(\mathbf{x})$, for some operator $\mathcal{T}(\cdot)$. Converting the transcendental equation $\mathbf{0} \in \partial f(\mathbf{x})$ algebraically into the form $\mathbf{x} \in \mathcal{T}(\mathbf{x})$, we obtain the following iterative scheme with recursive relation:

$$\mathbf{x}^{k+1} \in \mathcal{T}(\mathbf{x}^k),\ k = 0, 1, 2, ... \tag{3}$$

We now discuss how to adapt our algorithm into the iterative scheme in (3). First, we split the matrix $\mathbf{A}$ in (2) using the following strategy:

$$\mathbf{A} = \underbrace{\mathbf{L} + \tfrac{1}{\omega}(\mathbf{D} + \theta\mathbf{I})}_{\mathbf{B}} + \underbrace{\mathbf{L}^T + \tfrac{1}{\omega}((\omega - 1)\mathbf{D} - \theta\mathbf{I})}_{\mathbf{C}} \tag{4}$$

Here, $\omega \in (0, 2)$ is a relaxation parameter and $\theta \in [0, \infty)$ is a parameter for strong convexity that enforces $diag(\mathbf{B}) > \mathbf{0}$. These parameters are specified by the user beforehand. Using these notations, we obtain the following optimality condition which is equivalent to (2):

$$-\mathbf{Cx} - \mathbf{b} \in (\mathbf{B} + \partial h)(\mathbf{x})$$

Then, we have the following equivalent fixed-point equation:

$$\mathbf{x} \in \mathcal{T}(\mathbf{x}) \triangleq (\mathbf{B} + \partial h)^{-1}(-\mathbf{Cx} - \mathbf{b}) \tag{5}$$

Here, we name $\mathcal{T}$ the triangle proximal operator, which is novel in this paper[3]. Due to the triangle property of the

[1] $\mathbf{C} \succeq 0 \Leftrightarrow \forall \mathbf{x},\ \mathbf{x}^T\mathbf{Cx} \geq 0 \Leftrightarrow \forall \mathbf{x},\ \frac{1}{2}\mathbf{x}^T(\mathbf{C} + \mathbf{C}^T)\mathbf{x} \geq 0$

[2] For example, when $n = 3$, $\mathbf{D}$ and $\mathbf{L}$ take the following form:
$$\mathbf{D} = \begin{bmatrix} \mathbf{A}_{1,1} & 0 & 0 \\ 0 & \mathbf{A}_{2,2} & 0 \\ 0 & 0 & \mathbf{A}_{3,3} \end{bmatrix},\ \mathbf{L} = \begin{bmatrix} 0 & 0 & 0 \\ \mathbf{A}_{2,1} & 0 & 0 \\ \mathbf{A}_{3,1} & \mathbf{A}_{3,2} & 0 \end{bmatrix}$$

[3] This is in contrast with Moreau's proximal operator [27]: $\text{prox}_h(\mathbf{a}) = \arg\min_{\mathbf{x}} \frac{1}{2}\|\mathbf{x} - \mathbf{a}\|_2^2 + h(\mathbf{x}) = (\mathbf{I} + \partial h)^{-1}(\mathbf{a})$, where the mapping $(\mathbf{I} + \partial h)^{-1}$ is called the resolvent of the subdifferential operator ∂h.

matrix $\mathbf{B}$ and the element-wise separable structure of $h(\cdot)$, the triangle proximal operator $\mathcal{T}(\mathbf{x})$ in (5) can be computed exactly and analytically, by a generalized Gaussian elimination procedure (discussed later in Section 2.1). Our matrix splitting method iteratively applies $\mathbf{x}^{k+1} \Leftarrow \mathcal{T}(\mathbf{x}^k)$ until convergence. We summarize our algorithm in Algorithm 1.

In what follows, we show how to compute $\mathcal{T}(\mathbf{x})$ in (5) in Section 2.1, and then we study the convergence properties of Algorithm 1 in Section 2.2.

2.1. Computing the Triangle Proximal Operator

We now present how to compute the triangle proximal operator in (5), which is based on a new generalized Gaussian elimination procedure. Notice that (5) seeks a solution $\mathbf{z}^* \triangleq \mathcal{T}(\mathbf{x}^k)$ that satisfies the following nonlinear system:

$$\mathbf{0} \in \mathbf{B}\mathbf{z}^* + \mathbf{u} + \partial h(\mathbf{z}^*), \text{ where } \mathbf{u} = \mathbf{b} + \mathbf{C}\mathbf{x}^k \tag{6}$$

By taking advantage of the triangular form of $\mathbf{B}$ and the element-wise structure of $h(\cdot)$, the elements of $\mathbf{z}^*$ can be computed sequentially using forward substitution. Specifically, the above equation can be written as a system of nonlinear equations:

$$\mathbf{0} \in \begin{bmatrix} \mathbf{B}_{1,1} & 0 & 0 & 0 & 0 \\ \mathbf{B}_{2,1} & \mathbf{B}_{2,2} & 0 & 0 & 0 \\ \vdots & \vdots & \ddots & 0 & 0 \\ \mathbf{B}_{n-1,1} & \mathbf{B}_{n-1,2} & \cdots & \mathbf{B}_{n-1,n-1} & 0 \\ \mathbf{B}_{n,1} & \mathbf{B}_{n,2} & \cdots & \mathbf{B}_{n,n-1} & \mathbf{B}_{n,n} \end{bmatrix} \begin{bmatrix} \mathbf{z}_1^* \\ \mathbf{z}_2^* \\ \vdots \\ \mathbf{z}_{n-1}^* \\ \mathbf{z}_n^* \end{bmatrix} + \mathbf{u} + \partial h(\mathbf{z}^*)$$

If $\mathbf{z}^*$ satisfies the equations above, it must solve the following one-dimensional subproblems:

$$0 \in \mathbf{B}_{j,j}\mathbf{z}_j^* + \mathbf{w}_j + \partial h(\mathbf{z}_j^*), \ \forall j = 1, 2, \ \ldots, n,$$
$$\mathbf{w}_j = \mathbf{u}_j + \textstyle\sum_{i=1}^{j-1} \mathbf{B}_{j,i}\mathbf{z}_i^*$$

This is equivalent to solving the following one-dimensional problem for all $j = 1, 2, ..., n$:

$$\mathbf{z}_j^* = t^* \triangleq \arg\min_t \ \tfrac{1}{2}\mathbf{B}_{j,j}t^2 + \mathbf{w}_j t + h(t) \tag{7}$$

Note that the computation of $\mathbf{z}^*$ uses only the elements of $\mathbf{z}^*$ that have already been computed and a successive displacement strategy is applied to find $\mathbf{z}^*$.

We remark that the one-dimensional subproblem in (7) often admits a closed form solution for many problems of interest. For example, when $h(t) = I_{[lb,ub]}(t)$ with $I(\cdot)$ denoting an indicator function on the box constraint $lb \leq \mathbf{x} \leq ub$, the optimal solution can be computed as: $t^* = \min(ub, \max(lb, -\mathbf{w}_j/\mathbf{B}_{j,j}))$; when $h(t) = \lambda|t|$ (*e.g.* in the case of the ℓ_1 norm), the optimal solution can be computed as: $t^* = -\max(0, |\mathbf{w}_j/\mathbf{B}_{j,j}| - \lambda/\mathbf{B}_{j,j}) \cdot \mathrm{sign}(\mathbf{w}_j/\mathbf{B}_{j,j})$.

Our generalized Gaussian elimination procedure for computing $\mathcal{T}(\mathbf{x}^k)$ is summarized in Algorithm 2. Note that its computational complexity is $\mathcal{O}(n^2)$, which is the same as computing a matrix-vector product.

Algorithm 1 **MSM**: A Matrix Splitting Method for Solving the Composite Function Minimization Problem in (1)

```
1: Choose $\omega \in (0, 2)$, $\theta \in [0, \infty)$. Initialize $\mathbf{x}^0$, $k = 0$.
2: while not converge
3:    $\mathbf{x}^{k+1} = \mathcal{T}(\mathbf{x}^k)$ (Solve (6) by Algorithm 2)
4:    $k = k + 1$
5: end while
6: Output $\mathbf{x}^{k+1}$
```

Algorithm 2 A Generalized Gaussian Elimination Procedure for Computing the Triangle Proximal Operator $\mathcal{T}(\mathbf{x}^k)$.

```
1: Input $\mathbf{x}^k$
2: Initialization: compute $\mathbf{u} = \mathbf{b} + \mathbf{C}\mathbf{x}^k$
3: $\mathbf{x}_1 = \arg\min_t \frac{1}{2}\mathbf{B}_{1,1}t^2 + (\mathbf{u}_1)t + h(t)$
4: $\mathbf{x}_2 = \arg\min_t \frac{1}{2}\mathbf{B}_{2,2}t^2 + (\mathbf{u}_2 + \mathbf{B}_{2,1}\mathbf{x}_1)t + h(t)$
5: $\mathbf{x}_3 = \arg\min_t \frac{1}{2}\mathbf{B}_{3,3}t^2 + (\mathbf{u}_3 + \mathbf{B}_{3,1}\mathbf{x}_1 + \mathbf{B}_{3,2}\mathbf{x}_2)t + h(t)$
6: ...
7: $\mathbf{x}_n = \arg\min_t \frac{1}{2}\mathbf{B}_{n,n}t^2 + (\mathbf{u}_n + \sum_{i=1}^{n-1}\mathbf{B}_{n,i}\mathbf{x}_i)t + h(t)$
8: Collect $(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, ..., \mathbf{x}_n)^T$ as $\mathbf{x}^{k+1}$ and Output $\mathbf{x}^{k+1}$
```

2.2. Convergence Analysis

In what follows, we present our convergence analysis for Algorithm 1. We let $\mathbf{x}^*$ be the optimal solution set of (1). For notation simplicity, we denote:

$$\begin{aligned} \mathbf{r}^k \triangleq \mathbf{x}^k - \mathbf{x}^*, \ \mathbf{d}^k \triangleq \mathbf{x}^{k+1} - \mathbf{x}^k \\ u^k \triangleq f(\mathbf{x}^k) - f(\mathbf{x}^*), \ f^k \triangleq f(\mathbf{x}^k), \ f^* \triangleq f(\mathbf{x}^*) \end{aligned} \tag{8}$$

The following lemma characterizes the optimality of $\mathcal{T}(\mathbf{y})$.

Lemma 1. *For all* $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$*, it holds that:*

$$\mathbf{v} \in \partial h(\mathcal{T}(\mathbf{x})), \ \|\mathbf{A}\mathcal{T}(\mathbf{x}) + \mathbf{b} + \mathbf{v}\| \leq \|\mathbf{C}\|\|\mathbf{x} - \mathcal{T}(\mathbf{x})\| \tag{9}$$

$$\begin{aligned} f(\mathcal{T}(\mathbf{y})) - f(\mathbf{x}) \leq \langle \mathcal{T}(\mathbf{y}) - \mathbf{x}, \mathbf{C}(\mathcal{T}(\mathbf{y}) - \mathbf{y}) \\ - \tfrac{1}{2}(\mathbf{x} - \mathcal{T}(\mathbf{y}))^T\mathbf{A}(\mathbf{x} - \mathcal{T}(\mathbf{y}))\rangle \end{aligned} \tag{10}$$

Proof. (i) We now prove (9). By the optimality of $\mathcal{T}(\mathbf{x})$, we have: $\forall \mathbf{v} \in \partial h(\mathcal{T}(\mathbf{x}))$, $0 = \mathbf{B}\mathcal{T}(\mathbf{x}) + \mathbf{C}\mathcal{T}(\mathbf{x}) + \mathbf{v} + \mathbf{b} + \mathbf{C}\mathbf{x} - \mathbf{C}\mathcal{T}(\mathbf{x})$. Therefore, we obtain: $\mathbf{A}\mathcal{T}(\mathbf{x}) + \mathbf{b} + \mathbf{v} = \mathbf{C}(\mathcal{T}(\mathbf{x}) - \mathbf{x})$. Applying a norm inequality, we have (9).

(ii) We now prove (10). For simplicity, we denote $\mathbf{z}^* \triangleq \mathcal{T}(\mathbf{y})$. Thus, we obtain: $\mathbf{0} \in \mathbf{B}\mathbf{z}^* + \mathbf{b} + \mathbf{C}\mathbf{y} + \partial h(\mathbf{z}^*) \Rightarrow 0 \in \langle \mathbf{x} - \mathbf{z}^*, \mathbf{B}\mathbf{z}^* + \mathbf{b} + \mathbf{C}\mathbf{y} + \partial h(\mathbf{z}^*)\rangle$, $\forall \mathbf{x}$. Since $h(\cdot)$ is convex, we have:

$$\langle \mathbf{x} - \mathbf{z}^*, \partial h(\mathbf{z}^*)\rangle \leq h(\mathbf{x}) - h(\mathbf{z}^*) \tag{11}$$

Then we have this inequality: $\forall \mathbf{x}: \ h(\mathbf{x}) - h(\mathbf{z}^*) + \langle \mathbf{x} - \mathbf{z}^*, \mathbf{B}\mathbf{z}^* + \mathbf{b} + \mathbf{C}\mathbf{y}\rangle \geq 0$. We naturally derive the following results: $f(\mathbf{z}^*) - f(\mathbf{x}) = h(\mathbf{z}^*) - h(\mathbf{x}) + q(\mathbf{z}^*) - q(\mathbf{x}) \leq \langle \mathbf{x} - \mathbf{z}^*, \mathbf{B}\mathbf{z}^* + \mathbf{b} + \mathbf{C}\mathbf{y}\rangle + q(\mathbf{z}^*) - q(\mathbf{x}) = \langle \mathbf{x} - \mathbf{z}^*, \mathbf{B}\mathbf{z}^* +$

$\mathbf{Cy}\rangle + \frac{1}{2}\mathbf{z}^{*T}\mathbf{A}\mathbf{z}^* - \frac{1}{2}\mathbf{x}^T\mathbf{A}\mathbf{x} = \langle \mathbf{x}-\mathbf{z}^*, (\mathbf{B}-\mathbf{A})\mathbf{z}^* + \mathbf{Cy}\rangle - \frac{1}{2}(\mathbf{x}-\mathbf{z}^*)^T\mathbf{A}(\mathbf{x}-\mathbf{z}^*) = \langle \mathbf{x}-\mathbf{z}^*, \mathbf{C}(\mathbf{y}-\mathbf{z}^*)\rangle - \frac{1}{2}(\mathbf{x}-\mathbf{z}^*)^T\mathbf{A}(\mathbf{x}-\mathbf{z}^*)$.

□

Theorem 1. *(Proof of Global Convergence) We define $\delta \triangleq \frac{2\theta}{\omega} + \frac{2-\omega}{\omega}\min(diag(\mathbf{D}))$. Assume that ω and θ are chosen such that $\delta \in (0, \infty)$, Algorithm 1 is globally convergent.*

Proof. (i) First, the following results hold for all $\mathbf{z} \in \mathbb{R}^n$:

$$\begin{aligned} \mathbf{z}^T(\mathbf{A}-2\mathbf{C})\mathbf{z} = \ & \mathbf{z}^T(\mathbf{L}-\mathbf{L}^T + \tfrac{2-\omega}{\omega}\mathbf{D} + \tfrac{2\theta}{\omega}\mathbf{I})\mathbf{z} \\ = \ & \mathbf{z}^T(\tfrac{2\theta}{\omega}\mathbf{I} + \tfrac{2-\omega}{\omega}\mathbf{D})\mathbf{z} \geq \delta\|\mathbf{z}\|_2^2 \end{aligned} \quad (12)$$

where we have used the definition of $\mathbf{A}$ and $\mathbf{C}$, and the fact that $\mathbf{z}^T\mathbf{L}\mathbf{z} = \mathbf{z}^T\mathbf{L}^T\mathbf{z}$, $\forall \mathbf{z}$.

We invoke (10) in Lemma 1 with $\mathbf{x} = \mathbf{x}^k$, $\mathbf{y} = \mathbf{x}^k$ and combine the inequality in (12) to obtain:

$$f^{k+1} - f^k \leq -\tfrac{1}{2}\langle \mathbf{d}^k, (\mathbf{A}-2\mathbf{C})\mathbf{d}^k\rangle \leq -\tfrac{\delta}{2}\|\mathbf{d}^k\|_2^2 \quad (13)$$

(ii) Second, we invoke (9) in Lemma 1 with $\mathbf{x} = \mathbf{x}^k$ and obtain: $\mathbf{v} \in \partial h(\mathbf{x}^{k+1})$, $\frac{1}{\|\mathbf{C}\|}\|\mathbf{A}\mathbf{x}^{k+1} + \mathbf{b} + \mathbf{v}\| \leq \|\mathbf{x}^k - \mathbf{x}^{k+1}\|$. Combining with (13), we have: $\delta\|\partial f(\mathbf{x}^{k+1})\|_2^2/(2\|\mathbf{C}\|) \leq f^k - f^{k+1}$, where $\partial f(\mathbf{x}^k)$ is defined in (2). Summing this inequality over $i = 0, ..., k-1$, we have: $\delta\sum_{i=0}^{k-1}\|\partial f(\mathbf{x}^i)\|_2^2/(2\|\mathbf{C}\|) \leq f^0 - f^k \leq f^0 - f^*$, where we use $f^* \leq f^k$. As $k \to \infty$, we have $\partial f(\mathbf{x}^k) \to \mathbf{0}$, which implies the convergence of the algorithm.

Note that guaranteeing $\delta \in (0, \infty)$ can be achieved by simply choosing $\omega \in (0, 2)$ and setting θ to a small number.

□

We now prove the convergence rate of Algorithm 1. We make the following assumption, which characterizes the relations between $\mathcal{T}(\mathbf{x})$ and $\mathbf{x}^*$ for any $\mathbf{x}$.

Assumption 1. *If $\mathbf{x}$ is not the optimum of (1), there exists a constant $\eta \in (0, \infty)$ such that $\|\mathbf{x}-\mathbf{x}^*\| \leq \eta\|\mathbf{x} - \mathcal{T}(\mathbf{x})\|$.*

We remark that this assumption is similar to the classical local proximal error bound assumption in the literature [23, 35, 34, 41], and it is mild. Firstly, if $\mathbf{x}$ is not the optimum, we have $\mathbf{x} \neq \mathcal{T}(\mathbf{x})$. This is because when $\mathbf{x} = \mathcal{T}(\mathbf{x})$, we have $0 = \|\mathbf{C}\|\|\mathbf{x} - \mathcal{T}(\mathbf{x})\| \geq \|\mathbf{A}\mathcal{T}(\mathbf{x}) + \mathbf{b} + \mathbf{v}\|$, $\forall \mathbf{v} \in \partial h(\mathcal{T}(\mathbf{x}))$ (refer to the optimal condition of $\mathcal{T}(\mathbf{x})$ in (9)), which contradicts with the condition that $\mathbf{x}$ is not the optimal solution. Secondly, by the boundedness of $\mathbf{x}$ and $\mathbf{x}^*$, there exists a sufficiently large constant $\eta \in (0, \infty)$ such that $\|\mathbf{x}-\mathbf{x}^*\| \leq \eta\|\mathbf{x} - \mathcal{T}(\mathbf{x})\|$.

We now prove the convergence rate of Algorithm 1.

Theorem 2. *(Proof of Convergence Rate) We define $\delta \triangleq \frac{2\theta}{\omega} + \frac{2-\omega}{\omega}\min(diag(\mathbf{D}))$. Assume that ω and θ are chosen such that $\delta \in (0, \infty)$ and $\mathbf{x}^k$ is bound for all k, we have:*

$$\frac{f(\mathbf{x}^{k+1}) - f(\mathbf{x}^*)}{f(\mathbf{x}^k) - f(\mathbf{x}^*)} \leq \frac{C_1}{1 + C_1} \quad (14)$$

where $C_1 = ((3 + \eta^2\|\mathbf{C}\| + (2\eta^2 + 2)\|\mathbf{A}\|)/\delta$. In other words, Algorithm 1 converges to the optimal solution Q-linearly.

Proof. Invoking Assumption 1 with $\mathbf{x} = \mathbf{x}^k$, we obtain:

$$\|\mathbf{x}^k - \mathbf{x}^*\| \leq \eta\|\mathbf{x}^k - \mathcal{T}(\mathbf{x}^k)\| \ \Rightarrow \ \|\mathbf{r}^k\| \leq \eta\|\mathbf{d}^k\| \quad (15)$$

Invoking (10) in Lemma 1 with $\mathbf{x} = \mathbf{x}^*$, $\mathbf{y} = \mathbf{x}^k$, we derive the following inequalities:

$$\begin{aligned} & f^{k+1} - f^* \\ \leq \ & \langle \mathbf{r}^{k+1}, \mathbf{C}\mathbf{d}^k\rangle - \tfrac{1}{2}\langle \mathbf{r}^{k+1}, \mathbf{A}\mathbf{r}^{k+1}\rangle \\ = \ & \langle \mathbf{r}^k + \mathbf{d}^k, \mathbf{C}\mathbf{d}^k\rangle - \tfrac{1}{2}\langle \mathbf{r}^k + \mathbf{d}^k, \mathbf{A}(\mathbf{r}^k + \mathbf{d}^k)\rangle \\ \leq \ & \|\mathbf{C}\|(\|\mathbf{d}^k\|\|\mathbf{r}^k\| + \|\mathbf{d}^k\|_2^2) + \tfrac{1}{2}\|\mathbf{A}\|\|\mathbf{r}^k + \mathbf{d}^k\|_2^2 \\ \leq \ & \|\mathbf{C}\|(\tfrac{3}{2}\|\mathbf{d}^k\|_2^2 + \tfrac{1}{2}\|\mathbf{r}^k\|_2^2) + \|\mathbf{A}\|(\|\mathbf{r}^k\|_2^2 + \|\mathbf{d}^k\|_2^2) \quad (16) \\ \leq \ & \|\mathbf{C}\|\tfrac{3+\eta^2}{2}\|\mathbf{d}^k\|_2^2 + \|\mathbf{A}\|(\eta^2+1)\|\mathbf{d}^k\|_2^2 \\ \leq \ & ((3 + \eta^2\|\mathbf{C}\| + (2\eta^2 + 2)\|\mathbf{A}\|) \cdot (f^k - f^{k+1})/\delta \\ = \ & C_1(f^k - f^{k+1}) \\ = \ & C_1(f^k - f^*) - C_1(f^{k+1} - f^*) \end{aligned}$$

where the second step uses the fact that $\mathbf{r}^{k+1} = \mathbf{r}^k + \mathbf{d}^k$; the third step uses the Cauchy-Schwarz inequality $\langle \mathbf{x}, \mathbf{y}\rangle \leq \|\mathbf{x}\|\|\mathbf{y}\|$, $\forall \mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ and the norm inequality $\|\mathbf{A}\mathbf{x}\| \leq \|\mathbf{A}\|\|\mathbf{x}\|$, $\forall \mathbf{x} \in \mathbb{R}^n$; the fourth step uses the fact that $1/2 \cdot \|\mathbf{x}+\mathbf{y}\|_2^2 \leq \|\mathbf{x}\|_2^2 + \|\mathbf{y}\|_2^2$, $\forall \mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ and $ab \leq 1/2 \cdot a^2 + 1/2 \cdot b^2$, $\forall a, b \in \mathbb{R}$; the fifth step uses (15); the sixth step uses the descent condition in (13). Rearranging the last inequality in (16), we have $(1 + C_1)f(\mathbf{x}^{k+1}) - f(\mathbf{x}^*) \leq C_1(f(\mathbf{x}^k) - f(\mathbf{x}^*))$ and obtain the inequality in (14). Since $\frac{C_1}{1+C_1} < 1$, the sequence $\{f(\mathbf{x}^k)\}_{k=0}^{\infty}$ converges to $f(\mathbf{x}^*)$ linearly in the quotient sense.

□

The following lemma is useful in our proof of iteration complexity.

Lemma 2. *Suppose a nonnegative sequence $\{u^k\}_{k=0}^{\infty}$ satisfies $u^{k+1} \leq -2C + 2C\sqrt{1 + \frac{u^k}{C}}$ for some constant $C \geq 0$. It holds that: $u^{k+1} \leq \frac{C_2}{k+1}$, where $C_2 = \max(8C, 2\sqrt{Cu^0})$.*

Proof. The proof of this lemma can be obtained by mathematical induction. (i) When $k = 0$, we have $u^1 \leq -2C + 2C\sqrt{1 + \frac{1}{C}u^0} \leq -2C + 2C(1 + \sqrt{\frac{u^0}{C}}) = 2\sqrt{Cu^0} \leq \frac{C_0}{k+1}$. (ii) When $k \geq 1$, we assume that $u^k \leq \frac{C_2}{k}$ holds. We derive the following results: $k \geq 1 \Rightarrow \frac{k+1}{k} \leq 2 \Rightarrow 4C\frac{k+1}{k} \leq 8C \leq C_2 \Rightarrow \frac{4C}{k(k+1)} = 4C(\frac{1}{k} - \frac{1}{k+1}) \leq \frac{C_2}{(k+1)^2} \Rightarrow \frac{4C}{k} \leq \frac{4C}{k+1} + \frac{C_2}{(k+1)^2} \Rightarrow 4C^2(1 + \frac{C_2}{kC}) \leq 4C^2 + \frac{4CC_2}{k+1} + \frac{C_2^2}{(k+1)^2} \Rightarrow 2C\sqrt{1 + \frac{C_2}{Ck}} \leq 2C + \frac{C_2}{k+1} \Rightarrow -2C + 2C\sqrt{1 + \frac{C_2}{Ck}} \leq \frac{C_2}{k+1} \Rightarrow -2C + 2C\sqrt{1 + \frac{u^k}{C}} \leq \frac{C_2}{k+1} \Rightarrow u^{k+1} \leq \frac{C_2}{k+1}$. □

We now prove the iteration complexity of Algorithm 1.

Theorem 3. *(Proof of Iteration Complexity) We define $\delta \triangleq \frac{2\theta}{\omega} + \frac{2-\omega}{\omega}\min(diag(\mathbf{D}))$. Assume that ω and θ are chosen such that $\delta \in (0, \infty)$ and $\|\mathbf{x}^k\| \leq R$, $\forall k$, we have:*

$$u^k \leq \begin{cases} u^0(\frac{2C_4}{2C_4+1})^k, & \text{if } \sqrt{f^k - f^{k+1}} \geq C_3/C_4,\ \forall k \leq \bar{k} \\ \frac{C_5}{k}, & \text{if } \sqrt{f^k - f^{k+1}} < C_3/C_4,\ \forall k \geq 0 \end{cases}$$

where $C_3 = 2R\|\mathbf{C}\|(\frac{\delta}{2})^{1/2}$, $C_4 = \frac{\delta}{2}(\|\mathbf{C}\| + \|\mathbf{A}\|(\eta+1))$, $C_5 = \max(8C_3^2, 2C_3\sqrt{u^0})$, and $\bar{k}$ is some unknown iteration index.

Proof. Using similar strategies used in deriving (16), we have the following results:

$$\begin{aligned}
&u^{k+1} \\
&\leq \langle \mathbf{r}^{k+1}, \mathbf{C}\mathbf{d}^k \rangle - \tfrac{1}{2}\langle \mathbf{r}^{k+1}, \mathbf{A}\mathbf{r}^{k+1} \rangle \\
&= \langle \mathbf{r}^k + \mathbf{d}^k, \mathbf{C}\mathbf{d}^k \rangle - \tfrac{1}{2}\langle \mathbf{r}^k + \mathbf{d}^k, \mathbf{A}(\mathbf{r}^k + \mathbf{d}^k) \rangle \\
&\leq \|\mathbf{C}\|(\|\mathbf{r}^k\|\|\mathbf{d}^k\| + \|\mathbf{d}^k\|_2^2) + \tfrac{\|\mathbf{A}\|}{2}\|\mathbf{r}^k + \mathbf{d}^k\|_2^2 \qquad (17) \\
&\leq \|\mathbf{C}\|(\|\mathbf{r}^k\|\|\mathbf{d}^k\| + \|\mathbf{d}^k\|_2^2) + \|\mathbf{A}\|(\|\mathbf{r}^k\|_2^2 + \|\mathbf{d}^k\|_2^2) \\
&\leq \|\mathbf{C}\|(2R\|\mathbf{d}^k\| + \|\mathbf{d}^k\|_2^2) + \|\mathbf{A}\|(\eta\|\mathbf{d}^k\|_2^2 + \|\mathbf{d}^k\|_2^2) \\
&\leq C_3\sqrt{u^k - u^{k+1}} + C_4(u^k - u^{k+1})
\end{aligned}$$

Now we consider the two cases for the recursion formula in (17): (i) $\sqrt{u^k - u^{k+1}} \geq \frac{C_3}{C_4}$ for some $k \leq \bar{k}$ (ii) $\sqrt{u^k - u^{k+1}} \leq \frac{C_3}{C_4}$ for all $k \geq 0$. In case (i), (17) implies that we have $u^{k+1} \leq 2C_4(u^k - u^{k+1})$ and rearranging terms gives: $u^{k+1} \leq \frac{2C_4}{2C_4+1}u^k$. Thus, we have: $u^{k+1} \leq (\frac{2C_4}{2C_4+1})^{k+1}u^0$. We now focus on case (ii). When $\sqrt{u^k - u^{k+1}} \leq \frac{C_3}{C_4}$, (17) implies that we have $u^{k+1} \leq 2C_3\sqrt{u^k - u^{k+1}}$ and rearranging terms yields: $\frac{(u^{k+1})^2}{4C_3^2} + u^{k+1} \leq u^k$. Solving this quadratic inequality, we have: $u^{k+1} \leq -2C_3^2 + 2C_3^2\sqrt{1 + \frac{1}{C_3^2}u^k}$; solving this recursive formulation by Lemma 2, we obtain $u^{k+1} \leq \frac{C_5}{k+1}$. □

We have a few comments on Algorithm 1. **(i)** When $h(\cdot)$ is empty and $\theta = 0$, it reduces to the classical Gauss-Seidel method ($\omega = 1$) and Successive Over-Relaxation method ($\omega \neq 1$). **(ii)** When $\mathbf{A}$ contains zeros in its diagonal entries, one needs to set θ to a strictly positive number. This is to guarantee the strong convexity of the one dimensional subproblem and a bounded solution for any $h(\cdot)$. We remark that the introduction of the parameter θ is novel in this paper and it removes the assumption that $\mathbf{A}$ is strictly positive-definite or strictly diagonally dominant, which is used in the classical result of GS and SOC method [31, 6].

3. Extensions

This section discusses several extensions of our proposed matrix splitting method for solving (1).

3.1. When h is Nonconvex

When $h(\mathbf{x})$ is nonconvex, our theoretical analysis breaks down in (11) and the exact solution to the triangle proximal operator $\mathcal{T}(\mathbf{x}^k)$ in (6) cannot be guaranteed. However, our Gaussian elimination procedure in Algorithm 2 can still be applied. What one needs is to solve a one-dimensional nonconvex subproblem in (7). For example, when $h(t) = \lambda|t|_0$ (*e.g.* in the case of the ℓ_0 norm), it has an analytical solution: $t^* = \begin{cases} -\mathbf{w}_j/\mathbf{B}_{j,j}, & \mathbf{w}_j^2 > 2\lambda\mathbf{B}_{j,j} \\ 0, & \mathbf{w}_j^2 \leq 2\lambda\mathbf{B}_{j,j} \end{cases}$; when $h(t) = \lambda|t|^p$ and $p < 1$, it admits a closed form solution for some special values [36], such as $p = \frac{1}{2}$ or $\frac{2}{3}$.

Our matrix splitting method is guaranteed to converge even when $h(\cdot)$ is nonconvex. Specifically, we present the following theorem.

Theorem 4. *(Proof of Global Convergence when $h(\cdot)$ is Nonconvex) Assume the nonconvex one-dimensional subproblem in (7) can be solved globally and analytically. We define $\delta \triangleq \min(\theta/\omega + (1-\omega)/\omega \cdot diag(\mathbf{D}))$. If we choose ω and θ such that $\delta \in (0, \infty)$, we have: (i)*

$$f(\mathbf{x}^{k+1}) - f(\mathbf{x}^k) \leq -\tfrac{\delta}{2}\|\mathbf{x}^{k+1} - \mathbf{x}^k\|_2^2 \leq 0 \qquad (18)$$

(ii) Algorithm 1 is globally convergent.

Proof. (i) Due to the optimality of the one-dimensional subproblem in (7), for all $j = 1, 2, ..., n$, we have:

$$\begin{aligned}
&\tfrac{1}{2}\mathbf{B}_{j,j}(\mathbf{x}_j^{k+1})^2 + (\mathbf{u}_j + \textstyle\sum_{i=1}^{j-1}\mathbf{B}_{j,i}\mathbf{x}_i^{k+1})\mathbf{x}_j^{k+1} + h(\mathbf{x}_j^{k+1}) \\
&\leq \tfrac{1}{2}\mathbf{B}_{j,j}t_j^2 + (\mathbf{u}_j + \textstyle\sum_{i=1}^{j-1}\mathbf{B}_{j,i}\mathbf{x}_i^{k+1})t_j + h(t_j),\ \forall t_j
\end{aligned}$$

Letting $t_1 = \mathbf{x}_1^k,\ t_2 = \mathbf{x}_2^k,\ ...,\ t_n = \mathbf{x}_n^k$, we obtain:

$$\begin{aligned}
&\tfrac{1}{2}\textstyle\sum_i^n \mathbf{B}_{i,i}(\mathbf{x}_i^{k+1})^2 + \langle \mathbf{u} + \mathbf{L}\mathbf{x}^{k+1}, \mathbf{x}^{k+1} \rangle + h(\mathbf{x}^{k+1}) \\
&\leq \tfrac{1}{2}\textstyle\sum_i^n \mathbf{B}_{i,i}(\mathbf{x}_i^k)^2 + \langle \mathbf{u} + \mathbf{L}\mathbf{x}^{k+1}, \mathbf{x}^k \rangle + h(\mathbf{x}^k)
\end{aligned}$$

Since $\mathbf{u} = \mathbf{b} + \mathbf{C}\mathbf{x}^k$, we obtain the following inequality:

$$\begin{aligned}
&f^{k+1} + \tfrac{1}{2}\langle \mathbf{x}^{k+1}, (\tfrac{1}{\omega}(\mathbf{D} + \theta\mathbf{I}) + 2\mathbf{L} - \mathbf{A})\mathbf{x}^{k+1} + 2\mathbf{C}\mathbf{x}^k \rangle \\
&\leq f^k + \tfrac{1}{2}\langle \mathbf{x}^k, (\tfrac{1}{\omega}(\mathbf{D} + \theta\mathbf{I}) + 2\mathbf{C} - \mathbf{A})\mathbf{x}^k + 2\mathbf{L}\mathbf{x}^{k+1} \rangle
\end{aligned}$$

By denoting $\mathbf{S} \triangleq \mathbf{L} - \mathbf{L}^T$ and $\mathbf{T} \triangleq ((\omega - 1)\mathbf{D} - \theta\mathbf{I})/\omega$, we have: $\frac{1}{\omega}(\mathbf{D} + \theta\mathbf{I}) + 2\mathbf{L} - \mathbf{A} = \mathbf{T} - \mathbf{S}$, $\frac{1}{\omega}(\mathbf{D} + \theta\mathbf{I}) + 2\mathbf{C} - \mathbf{A} = \mathbf{S} - \mathbf{T}$, and $\mathbf{L} - \mathbf{C}^T = -\mathbf{T}$. Therefore, we have the following inequalities:

$$\begin{aligned}
&f^{k+1} - f^k \leq \tfrac{1}{2}\langle \mathbf{x}^{k+1}, (\mathbf{T} - \mathbf{S})\mathbf{x}^{k+1} \rangle - \langle \mathbf{x}^k, \mathbf{T}\mathbf{x}^{k+1} \rangle \\
&+ \tfrac{1}{2}\langle \mathbf{x}^k, (\mathbf{T} - \mathbf{S})\mathbf{x}^k \rangle = \tfrac{1}{2}\langle \mathbf{x}^k - \mathbf{x}^{k+1}, \mathbf{T}(\mathbf{x}^k - \mathbf{x}^{k+1}) \rangle \\
&\leq -\tfrac{\delta}{2}\|\mathbf{x}^{k+1} - \mathbf{x}^k\|_2^2
\end{aligned}$$

where the first equality uses $\langle \mathbf{x}, \mathbf{Sx} \rangle = 0 \ \forall \mathbf{x}$, since $\mathbf{S}$ is a Skew-Hermitian matrix. The last step uses $\mathbf{T} + \delta \mathbf{I} \preceq \mathbf{0}$, since $\mathbf{x} + \min(-\mathbf{x}) \leq \mathbf{0} \ \forall \mathbf{x}$. Thus, we obtain the sufficient decrease inequality in (18).

(ii) Based on the sufficient decrease inequality in (18), we have: $f(\mathbf{x}^k)$ is a non-increasing sequence, $\|\mathbf{x}^k - \mathbf{x}^{k+1}\| \to 0$, and $f(\mathbf{x}^{k+1}) < f(\mathbf{x}^k)$ if $\mathbf{x}^k \neq \mathbf{x}^{k+1}$. We note that (9) can be still applied even $h(\cdot)$ is nonconvex. Using the same methodology as in the second part of Theorem 1, we obtain that $\partial f(\mathbf{x}^k) \to \mathbf{0}$, which implies the convergence of the algorithm.

Note that guaranteeing $\delta \in (0, \infty)$ can be achieved by simply choosing $\omega \in (0, 1)$ and setting θ to a small number. $\square$

3.2. When x is a Matrix

In many applications (e.g. nonegative matrix factorization and sparse coding), the solutions exist in the matrix form as follows: $\min_{\mathbf{X} \in \mathbb{R}^{n \times r}} \ \frac{1}{2} tr(\mathbf{X}^T \mathbf{AX}) + tr(\mathbf{X}^T \mathbf{R}) + h(\mathbf{X})$, where $\mathbf{R} \in \mathbb{R}^{n \times r}$. Our matrix splitting algorithm can still be applied in this case. Using the same technique to decompose $\mathbf{A}$ as in (4): $\mathbf{A} = \mathbf{B} + \mathbf{C}$, one needs to replace (6) to solve the following nonlinear equation: $\mathbf{BZ}^* + \mathbf{U} + \partial h(\mathbf{Z}^*) \in 0$, where $\mathbf{U} = \mathbf{R} + \mathbf{CX}^k$. It can be decomposed into r independent components. By updating every column of $\mathbf{X}$, the proposed algorithm can be used to solve the matrix problem above. Thus, our algorithm can also make good use of existing parallel architectures to solve the matrix optimization problem.

3.3. When q is not Quadratic

Since the second-order Taylor series is a majorizer of any twice differentiable convex function $q(\mathbf{x})$ (not necessarily quadratic), the following inequality always holds [15]: $q(\mathbf{y}) \leq Q(\mathbf{y}, \mathbf{x}) \triangleq q(\mathbf{x}) + \langle \nabla q(\mathbf{x}), \mathbf{y} - \mathbf{x} \rangle + \frac{1}{2}(\mathbf{y} - \mathbf{x})^T \mathbf{M} (\mathbf{y} - \mathbf{x})$ for any $\mathbf{x}$ and $\mathbf{y}$ such that $\mathbf{M} \succ \nabla q^2(\mathbf{x})$, where $\nabla q(\mathbf{x})$ and $\nabla^2 q(\mathbf{x})$ denote the gradient and Hessian of $q(\mathbf{x})$ at $\mathbf{x}$, respectively. By minimizing (approximately) the upper bound of $q(\mathbf{y})$ (i.e. the quadratic surrogate function) at the current estimate $\mathbf{x}$, *i.e.* $\mathbf{x}^{k+1} = \arg\min_{\mathbf{y}} \ Q(\mathbf{y}, \mathbf{x}^k) + h(\mathbf{y})$, we can drive the objective downward until a stationary point is reached. In addition, one can perform line search by the update [35, 40]: $\mathbf{x}^{k+1} \Leftarrow \mathbf{x}^k + \beta(\mathbf{x}^{k+1} - \mathbf{x}^k)$ for the greatest descent in objective (as in the damped Newton method). Here, $\beta \in (0, 1]$ is the step-size selected by backtracking line search. Note that classical proximal point method considers $\mathbf{M} = ||\nabla q^2(\mathbf{x})|| \cdot \mathbf{I}$, while we can consider the lower triangle matrix of $\nabla q^2(\mathbf{x})$ in our algorithm.

4. Experiments

This section demonstrates the efficiency and efficacy of the proposed Matrix Splitting Method (MSM) by considering two important applications: nonnegative matrix factorization (NMF) [16, 18] and cardinality regularized sparse coding [26, 29, 17]. We implement our method in MATLAB on an Intel 2.6 GHz CPU with 8 GB RAM. Only our generalized Gaussian elimination procedure is developed in C and wrapped into the MATLAB code, since it requires an elementwise loop that is quite inefficient in native MATLAB. We report our results with the choice $\theta = 0.01$ and $\omega = 1$ in all our experiments. Some Matlab code can be found in the authors' research webpages.

4.1. Nonnegative Matrix Factorization (NMF)

Nonnegative matrix factorization [16] is a very useful tool for feature extraction and identification in the fields of text mining and image understanding. It is formulated as the following optimization problem:

$$\min_{\mathbf{W}, \mathbf{H}} \ \frac{1}{2} \|\mathbf{Y} - \mathbf{WH}\|_F^2, \ \ s.t. \ \ \mathbf{W} \geq 0, \ \mathbf{H} \geq 0 \qquad (19)$$

where $\mathbf{W} \in \mathbb{R}^{m \times n}$ and $\mathbf{H} \in \mathbb{R}^{n \times d}$. Following previous work [14, 9, 18, 12], we alternatively minimize the objective while keeping one of the two variables fixed. In each alternating subproblem, we solve a convex nonnegative least squares problem, where our MSM method is used. We conduct experiments on three datasets [4] 20news, COIL, and TDT2. The size of the datasets are 18774×61188, 7200×1024, 9394×36771, respectively. We compare MSM against the following state-of-the-art methods: (1) Projective Gradient (PG) [18, 5] that updates the current solution via steep gradient descent and then maps a point back to the bounded feasible region [5]; (2) Active Set (AS) method [14]; (3) Block Principal Pivoting (BPP) method [14] [6] that iteratively identifies an active and passive set by a principal pivoting procedure and solves a reduced linear system; (4) Accelerated Proximal Gradient (APG) [9] [7] that applies Nesterov's momentum strategy with a constant step size to solve the convex sub-problems; (5) Coordinate Gradient Descent (CGD) [12] [8] that greedily selects one coordinate by measuring the objective reduction and optimizes for a single variable via closed-form update. Similar to our method, the core procedure of CGD is developed in C and wrapped into the MATLAB code, while all other methods are implemented using builtin MATLAB functions.

We use the same settings as in [18]. We compare objective values after running t seconds with t varying from 20 to 50. Table 1 presents average results of using 10 random initial points, which are generated from a standard normal distribution. While the other methods may quickly lower

[4] http://www.cad.zju.edu.cn/home/dengcai/Data/TextData.html
[5] https://www.csie.ntu.edu.tw/~cjlin/libmf/
[6] http://www.cc.gatech.edu/~hpark/nmfsoftware.php
[7] https://sites.google.com/site/nmfsolvers/
[8] http://www.cs.utexas.edu/~cjhsieh/nmf/

data	n	[18] PG	[14] AS	[14] BPP	[9] APG	[12] CGD	[ours] MSM	data	n	[18] PG	[14] AS	[14] BPP	[9] APG	[12] CGD	[ours] MSM
time limit=20								time limit=40							
20news	20	5.001e+06	2.762e+07	8.415e+06	4.528e+06	4.515e+06	4.506e+06	20news	20	4.622e+06	2.762e+07	7.547e+06	4.495e+06	4.500e+06	4.496e+06
20news	50	5.059e+06	2.762e+07	4.230e+07	3.775e+06	3.544e+06	3.467e+06	20news	50	4.386e+06	2.762e+07	1.562e+07	3.564e+06	3.478e+06	3.438e+06
20news	100	6.955e+06	5.779e+06	4.453e+07	3.658e+06	3.971e+06	2.902e+06	20news	100	6.486e+06	2.762e+07	4.223e+07	3.128e+06	2.988e+06	2.783e+06
20news	200	7.675e+06	3.036e+06	1.023e+08	4.431e+06	3.573e+07	2.819e+06	20news	200	6.731e+06	1.934e+07	1.003e+08	3.304e+06	5.744e+06	2.407e+06
20news	300	1.997e+07	2.762e+07	1.956e+08	4.519e+06	4.621e+07	3.202e+06	20news	300	1.041e+07	2.762e+07	1.932e+08	3.621e+06	4.621e+07	2.543e+06
COIL	20	2.004e+09	5.480e+09	2.031e+09	1.974e+09	1.976e+09	1.975e+09	COIL	20	1.987e+09	5.141e+09	2.010e+09	1.974e+09	1.975e+09	1.975e+09
COIL	50	1.412e+09	1.516e+10	6.962e+09	1.291e+09	1.256e+09	1.252e+09	COIL	50	1.308e+09	2.403e+10	5.032e+09	1.262e+09	1.250e+09	1.248e+09
COIL	100	2.960e+09	2.834e+10	3.222e+10	9.919e+08	8.745e+08	8.510e+08	COIL	100	2.922e+09	2.834e+10	2.086e+10	9.161e+08	8.555e+08	8.430e+08
COIL	200	3.371e+09	2.834e+10	5.229e+10	8.495e+08	5.959e+08	5.600e+08	COIL	200	3.361e+09	2.834e+10	4.116e+10	7.075e+08	5.584e+08	5.289e+08
COIL	300	3.996e+09	2.834e+10	1.017e+11	8.493e+08	5.002e+08	4.956e+08	COIL	300	3.920e+09	2.834e+10	7.040e+10	6.221e+08	4.384e+08	4.294e+08
TDT2	20	1.597e+06	2.211e+06	1.688e+06	1.591e+06	1.595e+06	1.592e+06	TDT2	20	1.595e+06	2.211e+06	1.643e+06	1.591e+06	1.594e+06	1.592e+06
TDT2	50	1.408e+06	2.211e+06	2.895e+06	1.393e+06	1.390e+06	1.385e+06	TDT2	50	1.394e+06	2.211e+06	1.933e+06	1.392e+06	1.388e+06	1.384e+06
TDT2	100	1.300e+06	2.211e+06	6.187e+06	1.222e+06	1.224e+06	1.214e+06	TDT2	100	1.229e+06	2.211e+06	5.259e+06	1.213e+06	1.216e+06	1.211e+06
TDT2	200	1.628e+06	2.211e+06	1.791e+07	1.119e+06	1.227e+06	1.079e+06	TDT2	200	1.389e+06	1.547e+06	1.716e+07	1.046e+06	1.070e+06	1.041e+06
TDT2	300	1.915e+06	1.854e+06	3.412e+07	1.172e+06	7.902e+06	1.066e+06	TDT2	300	1.949e+06	1.836e+06	3.369e+07	1.008e+06	1.155e+06	9.776e+05
time limit=30								time limit=50							
20news	20	4.716e+06	2.762e+07	7.471e+06	4.510e+06	4.503e+06	4.500e+06	20news	20	4.565e+06	2.762e+07	6.939e+06	4.488e+06	4.498e+06	4.494e+06
20news	50	4.569e+06	2.762e+07	5.034e+07	3.628e+06	3.495e+06	3.446e+06	20news	50	4.343e+06	2.762e+07	1.813e+07	3.525e+06	3.469e+06	3.432e+06
20news	100	6.639e+06	2.762e+07	4.316e+07	3.293e+06	3.223e+06	2.817e+06	20news	100	6.404e+06	2.762e+07	3.955e+07	3.046e+06	2.878e+06	2.765e+06
20news	200	6.991e+06	2.762e+07	1.015e+08	3.609e+06	7.676e+06	2.507e+06	20news	200	5.939e+06	2.762e+07	9.925e+07	3.121e+06	4.538e+06	2.359e+06
20news	300	1.354e+07	2.762e+07	1.942e+08	4.519e+06	4.621e+07	3.097e+06	20news	300	9.258e+06	2.762e+07	1.912e+08	3.621e+06	2.323e+07	2.331e+06
COIL	20	1.992e+09	4.405e+09	2.014e+09	1.974e+09	1.975e+09	1.975e+09	COIL	20	1.982e+09	7.136e+09	2.033e+09	1.974e+09	1.975e+09	1.975e+09
COIL	50	1.335e+09	2.420e+10	5.772e+09	1.272e+09	1.252e+09	1.250e+09	COIL	50	1.298e+09	2.834e+10	4.365e+09	1.258e+09	1.248e+09	1.248e+09
COIL	100	2.936e+09	2.834e+10	1.814e+10	9.422e+08	8.623e+08	8.458e+08	COIL	100	1.945e+09	2.834e+10	1.428e+10	9.014e+08	8.516e+08	8.414e+08
COIL	200	3.362e+09	2.834e+10	4.627e+10	7.614e+08	5.720e+08	5.392e+08	COIL	200	3.362e+09	2.834e+10	3.760e+10	6.771e+08	5.491e+08	5.231e+08
COIL	300	3.946e+09	2.834e+10	7.417e+10	6.734e+08	4.609e+08	4.544e+08	COIL	300	3.905e+09	2.834e+10	6.741e+10	5.805e+08	4.226e+08	4.127e+08
TDT2	20	1.595e+06	2.211e+06	1.667e+06	1.591e+06	1.594e+06	1.592e+06	TDT2	20	1.595e+06	2.211e+06	1.622e+06	1.591e+06	1.594e+06	1.592e+06
TDT2	50	1.397e+06	2.211e+06	2.285e+06	1.393e+06	1.389e+06	1.385e+06	TDT2	50	1.393e+06	2.211e+06	1.875e+06	1.392e+06	1.386e+06	1.384e+06
TDT2	100	1.241e+06	2.211e+06	5.702e+06	1.216e+06	1.219e+06	1.212e+06	TDT2	100	1.223e+06	2.211e+06	4.831e+06	1.212e+06	1.214e+06	1.210e+06
TDT2	200	1.484e+06	1.878e+06	1.753e+07	1.063e+06	1.104e+06	1.049e+06	TDT2	200	1.267e+06	2.211e+06	1.671e+07	1.040e+06	1.054e+06	1.036e+06
TDT2	300	1.879e+06	2.211e+06	3.398e+07	1.060e+06	1.669e+06	1.007e+06	TDT2	300	1.903e+06	2.211e+06	3.328e+07	9.775e+05	1.045e+06	9.606e+05

Table 1: Comparisons of objective values for non-negative matrix factorization for all the compared methods. The 1^{st}, 2^{nd}, and 3^{rd} best results are colored with red, blue and green, respectively.

objective values when n is small ($n = 20$), MSM catches up very quickly and achieves a faster convergence speed when n is large. It generally achieves the best performance in terms of objective value among all the methods.

4.2. Cardinality Regularized Sparse Coding

Sparse coding is a popular unsupervised feature learning technique for data representation that is widely used in computer vision and medical imaging. Motivated by recent success in ℓ_0 norm modeling [39, 2, 37], we consider the following cardinality regularized (*i.e.* ℓ_0 norm) sparse coding problem:

$$\min_{\mathbf{W},\mathbf{H}} \ \tfrac{1}{2}\|\mathbf{Y}-\mathbf{W}\mathbf{H}\|_F^2+\lambda\|\mathbf{H}\|_0, \ s.t. \ \|\mathbf{W}(:,i)\|_2=1 \ \forall i \quad (20)$$

with $\mathbf{W} \in \mathbb{R}^{m\times n}$ and $\mathbf{H} \in \mathbb{R}^{n\times d}$. Existing solutions for this problem are mostly based on the family of proximal point methods [25, 2]. We compare MSM with the following methods: (1) Proximal Gradient Method (PGM) with constant step size, (2) PGM with line search, (3) accelerated PGM with constant step size, and (4) accelerated PGM with line search.

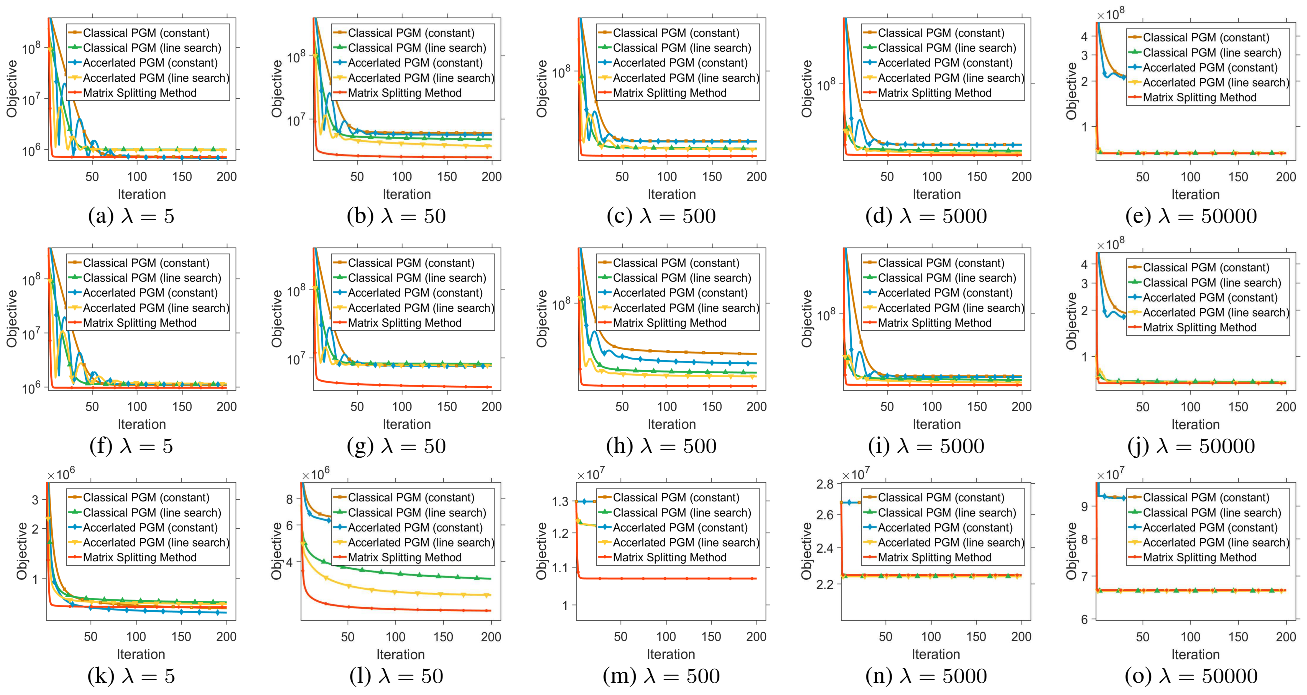
We evaluate all the methods for the application of image denoising. Following [1, 2], we set the dimension of the dictionary to $n = 256$. The dictionary is learned from $m = 1000$ image patches randomly chosen from the noisy input image. The patch size is 8×8, leading to $d = 64$. The experiments are conducted on 16 conventional test images with different noise standard deviations σ. For the regulization parameter λ, we sweep over $\{1, 3, 5, 7, 9, ..., 10000, 30000, 50000, 70000, 90000\}$.

First, we compare the objective values for all methods by *fixing* the variable $\mathbf{W}$ to an over-complete DCT dictionary [1] and *only* optimizing over $\mathbf{H}$. We compare all methods with varying regularization parameter λ and different initial points that are either generated by random Gaussian sampling or the Orthogonal Matching Pursuit (OMP) method [33]. In Figure 2, we observe that MSM converges rapidly in 10 iterations. Moreover, it often generates much better local optimal solutions than the compared methods.

Second, we evaluate the methods according to Signal-to-Noise Ratio (SNR) value wrt the groundtruth denoised image. In this case, we minimize over $\mathbf{W}$ and $\mathbf{H}$ *alternatingly* with different initial points (OMP initialization or standard normal random initialization). For updating $\mathbf{W}$, we use the same proximal gradient method as in [2]. For updating $\mathbf{H}$, since the accelerated PGM does not necessarily present better performance than canonical PGM and since the line search strategy does not present better performance than the simple constant step size, we only compare with PGM, which has been implemented in [2] [9] and KSVD [1] [10]. In our experiments, we find that MSM achieves lower

[9]Code: http://www.math.nus.edu.sg/~matjh/research/research.htm

[10]Code: http://www.cs.technion.ac.il/~elad/software/



(a) $\lambda = 5$ (b) $\lambda = 50$ (c) $\lambda = 500$ (d) $\lambda = 5000$ (e) $\lambda = 50000$

(f) $\lambda = 5$ (g) $\lambda = 50$ (h) $\lambda = 500$ (i) $\lambda = 5000$ (j) $\lambda = 50000$

(k) $\lambda = 5$ (l) $\lambda = 50$ (m) $\lambda = 500$ (n) $\lambda = 5000$ (o) $\lambda = 50000$

Figure 2: Convergence behavior for solving (20) with fixing $\mathbf{W}$ for different λ and initializations. Denoting $\tilde{\mathbf{O}}$ as an arbitrary standard Gaussian random matrix of suitable size, we consider the following three initializations for $\mathbf{H}$. First row: $\mathbf{H} = 0.1 \times \tilde{\mathbf{O}}$. Second row: $\mathbf{H} = 10 \times \tilde{\mathbf{O}}$. Third row: $\mathbf{H}$ is set to the output of the orthogonal matching pursuit.

		OMP init		random init	
img + σ	KSVD	PGM	MSM	PGM	MSM
walkbridge + 5	35.70	35.71	35.71	35.72	35.75
walkbridge + 10	31.07	31.07	31.17	31.07	31.17
walkbridge + 20	27.01	27.11	27.21	27.07	27.23
walkbridge + 30	24.93	25.08	25.21	25.09	25.19
walkbridge + 40	23.71	23.85	23.87	23.84	23.90
mandrill + 5	35.18	35.21	35.20	35.22	35.21
mandrill + 10	30.36	30.38	30.47	30.38	30.47
mandrill + 20	26.02	26.15	26.31	26.12	26.33
mandrill + 30	23.75	23.96	24.19	23.96	24.18
mandrill + 40	22.37	22.61	22.78	22.57	22.81
livingroom + 5	37.00	36.97	37.10	36.94	37.07
livingroom + 10	32.98	33.02	33.19	32.92	33.27
livingroom + 20	29.22	29.16	29.55	29.23	29.57
livingroom + 30	27.04	27.04	27.43	27.06	27.45
livingroom + 40	25.62	25.55	25.78	25.59	25.81
lake + 5	36.77	36.74	36.77	36.73	36.75
lake + 10	32.84	32.75	32.86	32.77	32.90
lake + 20	29.32	29.19	29.36	29.23	29.38
lake + 30	27.32	27.04	27.30	27.10	27.33
lake + 40	25.94	25.66	25.80	25.59	25.80
blonde + 5	36.98	37.00	37.06	37.00	37.08
blonde + 10	33.23	33.31	33.37	33.27	33.43
blonde + 20	29.83	29.78	29.99	29.75	30.00
blonde + 30	28.00	27.77	28.04	27.73	27.95
blonde + 40	26.82	26.33	26.41	26.31	26.63
barbara + 5	37.50	37.46	37.74	37.42	37.71
barbara + 10	33.36	33.37	33.65	33.33	33.67
barbara + 20	29.19	29.22	29.70	29.30	29.63
barbara + 30	26.79	26.84	27.30	26.92	27.36
barbara + 40	25.11	25.02	25.71	25.16	25.83
boat + 5	36.94	36.94	36.98	36.93	37.01
boat + 10	33.10	33.02	33.24	33.05	33.31
boat + 20	29.47	29.40	29.66	29.52	29.65
boat + 30	27.50	27.27	27.50	27.29	27.52
boat + 40	26.16	25.84	26.13	25.86	26.03
pirate + 5	36.49	36.43	36.54	36.42	36.50
pirate + 10	32.19	32.09	32.25	32.10	32.29
pirate + 20	28.33	28.23	28.42	28.24	28.44
pirate + 30	26.31	26.12	26.28	26.13	26.33
pirate + 40	24.94	24.71	24.86	24.72	24.86
jetplane + 5	38.84	38.87	39.06	38.93	39.05
jetplane + 10	34.98	34.99	35.17	35.01	35.22
jetplane + 20	31.30	31.04	31.28	31.01	31.30
jetplane + 30	29.11	28.64	28.87	28.68	28.95
jetplane + 40	27.57	27.05	27.19	26.97	27.26

Table 2: Comparisons of SNR values for the sparse coding based image denoising problem with OMP initialization and random initialization. The 1st, 2nd, and 3rd best results are colored with red, blue and green, respectively.

objectives than PGM in all cases. We do not report the objective values here but only the best SNR value, since (i) the best SNR result does not correspond to the same λ for PGM and MSM, and (ii) KSVD does not solve exactly the same problem in (20)[11]. We observe that MSM is generally 4-8 times faster than KSVD. This is not surprising, since KSVD needs to call OMP to update the dictionary $\mathbf{H}$, which involves high computational complexity while our method only needs to call a generalized Gaussian elimination procedure in each iteration. In Table 2, we summarize the results, from which we make two conclusions. (i) The two initialization strategies generally lead to similar SNR results. (ii) Our MSM method generally leads to a larger SNR than PGM and a comparable SNR as KSVD, but in less time.

[11] In fact, it solves an ℓ_0 norm constrained problem using a greedy pursuit algorithm and performs a codebook update using SVD. It may not necessarily converge, which motivates the use of the alternating minimization algorithm in [2].

Acknowledgments. This work was supported by the King Abdullah University of Science and Technology (KAUST) Office of Sponsored Research and, in part, by the NSF-China (61402182, 61472456, 61522115, 61628212).

References

[1] M. Aharon, M. Elad, and A. Bruckstein. K-svd: An algorithm for designing overcomplete dictionaries for sparse representation. *IEEE Transactions on Signal Processing*, 54(11):4311–4322, 2006. 1, 7

[2] C. Bao, H. Ji, Y. Quan, and Z. Shen. Dictionary learning for sparse coding: Algorithms and convergence analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 38(7):1356–1369, 2016. 1, 7, 8

[3] A. Beck and M. Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences (SIIMS)*, 2(1):183–202, 2009. 1, 2

[4] A. Beck and L. Tetruashvili. On the convergence of block coordinate descent type methods. *SIAM journal on Optimization (SIOPT)*, 23(4):2037–2060, 2013. 2

[5] D. P. Bertsekas. *Nonlinear programming*. Athena scientific Belmont, 1999. 6

[6] J. W. Demmel. *Applied numerical linear algebra*. SIAM, 1997. 1, 5

[7] D. L. Donoho. Compressed sensing. *IEEE Transactions on Information Theory*, 52(4):1289–1306, 2006. 1

[8] E. Elhamifar and R. Vidal. Sparse subspace clustering: Algorithm, theory, and applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 35(11):2765–2781, 2013. 1

[9] N. Guan, D. Tao, Z. Luo, and B. Yuan. Nenmf: an optimal gradient method for nonnegative matrix factorization. *IEEE Transactions on Signal Processing*, 60(6):2882–2898, 2012. 1, 6, 7

[10] M. Hong, X. Wang, M. Razaviyayn, and Z.-Q. Luo. Iteration complexity analysis of block coordinate descent methods. *Mathematical Programming*, pages 1–30, 2013. 2

[11] C.-J. Hsieh, K.-W. Chang, C.-J. Lin, S. S. Keerthi, and S. Sundararajan. A dual coordinate descent method for large-scale linear svm. In *International Conference on Machine Learning (ICML)*, pages 408–415, 2008. 1

[12] C.-J. Hsieh and I. S. Dhillon. Fast coordinate descent methods with variable selection for non-negative matrix factorization. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, pages 1064–1072, 2011. 6, 7

[13] R. Johnson and T. Zhang. Accelerating stochastic gradient descent using predictive variance reduction. In *Advances in Neural Information Processing Systems (NIPS)*, pages 315–323, 2013. 2

[14] J. Kim and H. Park. Fast nonnegative matrix factorization: An active-set-like method and comparisons. *SIAM Journal on Scientific Computing (SISC)*, 33(6):3261–3281, 2011. 6, 7

[15] K. Lange, D. R. Hunter, and I. Yang. Optimization transfer using surrogate objective functions. *Journal of Computational and Graphical Statistics*, 9(1):1–20, 2000. 6

[16] D. D. Lee and H. S. Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–791, 1999. 1, 6

[17] H. Lee, A. Battle, R. Raina, and A. Y. Ng. Efficient sparse coding algorithms. In *Advances in Neural Information Processing Systems (NIPS)*, pages 801–808, 2006. 1, 6

[18] C.-J. Lin. Projected gradient methods for nonnegative matrix factorization. *Neural Computation*, 19(10):2756–2779, 2007. 1, 6, 7

[19] Q. Lin, Z. Lu, and L. Xiao. An accelerated randomized proximal coordinate gradient method and its application to regularized empirical risk minimization. *SIAM Journal on Optimization (SIOPT)*, 25(4):2244–2273, 2015. 2

[20] J. Liu and S. J. Wright. Asynchronous stochastic coordinate descent: Parallelism and convergence properties. *SIAM Journal on Optimization (SIOPT)*, 25(1):351–376, 2015. 2

[21] Z. Lu and L. Xiao. Randomized block coordinate non-monotone gradient method for a class of nonlinear programming. *arXiv preprint*, 2013. 2

[22] Z. Lu and L. Xiao. On the complexity analysis of randomized block-coordinate descent methods. *Mathematical Programming*, 152(1-2):615–642, 2015. 2

[23] Z.-Q. Luo and P. Tseng. Error bounds and convergence analysis of feasible descent methods: a general approach. *Annals of Operations Research*, 46(1):157–178, 1993. 4

[24] Y. Nesterov. Efficiency of coordinate descent methods on huge-scale optimization problems. *SIAM Journal on Optimization (SIOPT)*, 22(2):341–362, 2012. 2

[25] Y. Nesterov. *Introductory lectures on convex optimization: A basic course*, volume 87. Springer Science & Business Media, 2013. 1, 2, 7

[26] B. A. Olshausen et al. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381(6583):607–609, 1996. 6

[27] N. Parikh, S. P. Boyd, et al. Proximal algorithms. *Foundations and Trends in optimization*, 1(3):127–239, 2014. 2

[28] A. Patrascu and I. Necoara. Iteration complexity analysis of random coordinate descent methods for ℓ_0 regularized convex problems. *arXiv preprint*, 2014. 1

[29] Y. Quan, Y. Xu, Y. Sun, Y. Huang, and H. Ji. Sparse coding for classification via discrimination ensemble. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016. 1, 6

[30] P. Richtárik and M. Takáč. Iteration complexity of randomized block-coordinate descent methods for minimizing a composite function. *Mathematical Programming*, 144(1-2):1–38, 2014. 2

[31] Y. Saad. *Iterative Methods for Sparse Linear Systems*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2nd edition, 2003. 1, 5

[32] R. Sun and M. Hong. Improved iteration complexity bounds of cyclic block coordinate descent for convex problems. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1306–1314, 2015. 1

[33] J. A. Tropp and A. C. Gilbert. Signal recovery from random measurements via orthogonal matching pursuit. *IEEE Transactions on Information Theory*, 53(12):4655–4666, 2007. 7

[34] P. Tseng. Approximation accuracy, gradient methods, and error bound for structured convex optimization. *Mathematical Programming*, 125(2):263–295, 2010. 4

[35] P. Tseng and S. Yun. A coordinate gradient descent method for nonsmooth separable minimization. *Mathematical Programming*, 117(1-2):387–423, 2009. 1, 4, 6

[36] Z. Xu, X. Chang, F. Xu, and H. Zhang. L1/2 regularization: A thresholding representation theory and a fast solver. *IEEE Transactions on Neural Networks and Learning Systems (TNNLS)*, 23(7):1013–1027, 2012. 5

[37] Y. Yang, J. Feng, N. Jojic, J. Yang, and T. S. Huang. ℓ^0-sparse subspace clustering. *European Conference on Computer Vision (ECCV)*, 2016. 7

[38] H.-F. Yu, F.-L. Huang, and C.-J. Lin. Dual coordinate descent methods for logistic regression and maximum entropy models. *Machine Learning*, 85(1-2):41–75, 2011. 1

[39] G. Yuan and B. Ghanem. $\ell_0 tv$: A new method for image restoration in the presence of impulse noise. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5369–5377, 2015. 7

[40] X.-T. Yuan and Q. Liu. Newton greedy pursuit: A quadratic approximation method for sparsity-constrained optimization. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4122–4129, 2014. 1, 6

[41] S. Yun, P. Tseng, and K.-C. Toh. A block coordinate gradient descent method for regularized convex separable optimization and covariance selection. *Mathematical Programming*, 129(2):331–355, 2011. 4

[42] J. Zeng, Z. Peng, and S. Lin. GAITA: A gauss-seidel iterative thresholding algorithm for ℓ_q regularized least squares regression. *Journal of Computational and Applied Mathematics*, 319:220–235, 2017. 2