Linearized Alternating Direction Method with Adaptive Penalty for Low-Rank Representation

Zhouchen Lin
Visual Computing Group
Microsoft Research Asia

Risheng Liu **Zhixun Su**
School of Mathematical Sciences
Dalian University of Technology

Abstract

Many machine learning and signal processing problems can be formulated as linearly constrained convex programs, which could be efficiently solved by the alternating direction method (ADM). However, usually the subproblems in ADM are easily solvable only when the linear mappings in the constraints are identities. To address this issue, we propose a linearized ADM (LADM) method by linearizing the quadratic penalty term and adding a proximal term when solving the subproblems. For fast convergence, we also allow the penalty to change adaptively according a novel update rule. We prove the global convergence of LADM with adaptive penalty (LADMAP). As an example, we apply LADMAP to solve low-rank representation (LRR), which is an important subspace clustering technique yet suffers from high computation cost. By combining LADMAP with a skinny SVD representation technique, we are able to reduce the complexity $O(n^3)$ of the original ADM based method to $O(rn^2)$, where r and n are the rank and size of the representation matrix, respectively, hence making LRR possible for large scale applications. Numerical experiments verify that for LRR our LADMAP based methods are much faster than state-of-the-art algorithms.

1 Introduction

Recently, compressive sensing [5] and sparse representation [19] have been hot research topics and also have found abundant applications in signal processing and machine learning. Many of the problems in these fields can be formulated as the following linearly constrained convex programs:

$$\min_{\mathbf{x},\mathbf{y}} f(\mathbf{x}) + g(\mathbf{y}),\ s.t.\ \mathcal{A}(\mathbf{x}) + \mathcal{B}(\mathbf{y}) = \mathbf{c}, \tag{1}$$

where $\mathbf{x}$, $\mathbf{y}$ and $\mathbf{c}$ could be either vectors or matrices, f and g are convex functions (e.g., the nuclear norm $\|\cdot\|_*$ [2], Frobenius norm $\|\cdot\|$, $l_{2,1}$ norm $\|\cdot\|_{2,1}$ [13], and l_1 norm $\|\cdot\|_1$), and $\mathcal{A}$ and $\mathcal{B}$ are linear mappings.

Although the interior point method can be used to solve many convex programs, it may suffer from unbearably high computation cost when handling large scale problems. For example, when using CVX, an interior point based toolbox, to solve nuclear norm minimization (namely, $f(\mathbf{X}) = \|\mathbf{X}\|_*$ in (1)) problems, such as matrix completion [4], robust principal component analysis [18] and their combination [3], the complexity of each iteration is $O(n^6)$, where $n \times n$ is the matrix size. To overcome this issue, first-order methods are often preferred. The accelerated proximal gradient (APG) algorithm [16] is a popular technique due to its guaranteed $O(k^{-2})$ convergence rate, where k is the iteration number. The alternating direction method (ADM) has also regained a lot of attention [11, 15]. It updates the variables alternately by minimizing the augmented Lagrangian function with respect to the variables in a Gauss-Seidel manner. While APG has to convert (1) into an approximate unconstrained problem by adding the linear constraints to the objective function as a penalty, hence only producing an approximate solution to (1), ADM can solve (1) exactly. However, when

$\mathcal{A}$ or $\mathcal{B}$ is not the identity mapping, the subproblems in ADM may not have closed form solutions. So solving them is cumbersome.

In this paper, we propose a linearized version of ADM (LADM) to overcome the difficulty in solving subproblems. It is to replace the quadratic penalty term by linearizing the penalty term and adding a proximal term. We also allow the penalty parameter to change adaptively and propose a novel and simple rule to update it. Linearization makes the auxiliary variables unnecessary, hence saving memory and waiving the expensive matrix inversions to update the auxiliary variables. Moreover, without the extra constraints introduced by the auxiliary variables, the convergence is also faster. Using a variable penalty parameter further speeds up the convergence. The global convergence of LADM with adaptive penalty (LADMAP) is also proven.

As an example, we apply our LADMAP to solve the low-rank representation (LRR) problem [12][1]:

$$\min_{\mathbf{Z},\mathbf{E}} \|\mathbf{Z}\|_* + \mu\|\mathbf{E}\|_{2,1},\ s.t.\ \mathbf{X} = \mathbf{X}\mathbf{Z} + \mathbf{E}, \tag{2}$$

where $\mathbf{X}$ is the data matrix. LRR is an important robust subspace clustering technique and has found wide applications in machine learning and computer vision, e.g., motion segmentation, face clustering, and temporal segmentation [12, 14, 6]. However, the existing LRR solver [12] is based on ADM, which suffers from $O(n^3)$ computation complexity due to the matrix-matrix multiplications and matrix inversions. Moreover, introducing auxiliary variables also slows down the convergence, as there are more variables and constraints. Such a heavy computation load prevents LRR from large scale applications. It is LRR that motivated us to develop LADMAP. We show that LADMAP can be successfully applied to LRR, obtaining faster convergence speed than the original solver. By further representing $\mathbf{Z}$ as its skinny SVD and utilizing an advanced functionality of the PROPACK [9] package, the complexity of solving LRR by LADMAP becomes only $O(rn^2)$, as there is no full sized matrix-matrix multiplications, where r is the rank of the optimal $\mathbf{Z}$. Numerical experiments show the great speed advantage of our LADMAP based methods for solving LRR.

Our work is inspired by Yang et al. [20]. Nonetheless, the difference of our work from theirs is distinct. First, they only proved the convergence of LADM for a specific problem, namely nuclear norm regularization. Their proof utilized some special properties of the nuclear norm, while we prove the convergence of LADM for general problems in (1). Second, they only proved in the case of fixed penalty, while we prove in the case of variable penalty. Although they mentioned the dynamic updating rule proposed in [8], their proof cannot be straightforwardly applied to the case of variable penalty. Moreover, that rule is for ADM only. Third, the convergence speed of LADM heavily depends on the choice of penalty. So it is difficult to choose an optimal fixed penalty that fits for different problems and problem sizes, while our novel updating rule for the penalty, although simple, is effective for different problems and problem sizes. The linearization technique has also been used in other optimization methods. For example, Yin [22] applied this technique to the Bregman iteration for solving compressive sensing problems and proved that the linearized Bregman method converges to an exact solution *conditionally*. In comparison, LADM (and LADMAP) always converges to an exact solution.

2 Linearized Alternating Direction Method with Adaptive Penalty

2.1 The Alternating Direction Method

ADM is now very popular in solving large scale machine learning problems [1]. When solving (1) by ADM, one operates on the following augmented Lagrangian function:

$$L(\mathbf{x}, \mathbf{y}, \lambda) = f(\mathbf{x}) + g(\mathbf{y}) + \langle \lambda, \mathcal{A}(\mathbf{x}) + \mathcal{B}(\mathbf{y}) - \mathbf{c} \rangle + \frac{\beta}{2}\|\mathcal{A}(\mathbf{x}) + \mathcal{B}(\mathbf{y}) - \mathbf{c}\|^2, \tag{3}$$

where λ is the Lagrange multiplier, $\langle \cdot, \cdot \rangle$ is the inner product, and $\beta > 0$ is the penalty parameter. The usual augmented Lagrange multiplier method is to minimize L w.r.t. $\mathbf{x}$ and $\mathbf{y}$ simultaneously. This is usually difficult and does not exploit the fact that the objective function is separable. To remedy this issue, ADM decomposes the minimization of L w.r.t. $(\mathbf{x}, \mathbf{y})$ into two subproblems that

[1] Here we switch to bold capital letters in order to emphasize that the variables are matrices.

minimize w.r.t. $\mathbf{x}$ and $\mathbf{y}$, respectively. More specifically, the iterations of ADM go as follows:

$$\begin{aligned}
\mathbf{x}_{k+1} &= \arg\min_{\mathbf{x}} L(\mathbf{x}, \mathbf{y}_k, \lambda_k) \\
&= \arg\min_{\mathbf{x}} f(\mathbf{x}) + \frac{\beta}{2}\|\mathcal{A}(\mathbf{x}) + \mathcal{B}(\mathbf{y}_k) - \mathbf{c} + \lambda_k/\beta\|^2, \qquad (4) \\
\mathbf{y}_{k+1} &= \arg\min_{\mathbf{y}} L(\mathbf{x}_{k+1}, \mathbf{y}, \lambda_k) \\
&= \arg\min_{\mathbf{y}} g(\mathbf{y}) + \frac{\beta}{2}\|\mathcal{B}(\mathbf{y}) + \mathcal{A}(\mathbf{x}_{k+1}) - \mathbf{c} + \lambda_k/\beta\|^2, \qquad (5) \\
\lambda_{k+1} &= \lambda_k + \beta[\mathcal{A}(\mathbf{x}_{k+1}) + \mathcal{B}(\mathbf{y}_{k+1}) - \mathbf{c}]. \qquad (6)
\end{aligned}$$

In many machine learning problems, as f and g are matrix or vector norms, the subproblems (4) and (5) usually have closed form solutions when $\mathcal{A}$ and $\mathcal{B}$ are identities [2, 12, 21]. In this case, ADM is appealing. However, in many problems $\mathcal{A}$ and $\mathcal{B}$ are not identities. For example, in matrix completion $\mathcal{A}$ can be a selection matrix, and in LRR and 1D sparse representation $\mathcal{A}$ can be a general matrix. In this case, there are no closed form solutions to (4) and (5). Then (4) and (5) have to be solved iteratively. To overcome this difficulty, a common strategy is to introduce auxiliary variables [12, 1] $\mathbf{u}$ and $\mathbf{v}$ and reformulate problem (1) into an equivalent one:

$$\min_{\mathbf{x},\mathbf{y},\mathbf{u},\mathbf{v}} f(\mathbf{x}) + g(\mathbf{y}),\ s.t.\ \mathcal{A}(\mathbf{u}) + \mathcal{B}(\mathbf{v}) = \mathbf{c}, \mathbf{x} = \mathbf{u}, \mathbf{y} = \mathbf{v}, \qquad (7)$$

and the corresponding ADM iterations analogous to (4)-(6) can be deduced. With more variables and more constraints, more memory is required and the convergence of ADM also becomes slower. Moreover, to update $\mathbf{u}$ and $\mathbf{v}$, whose subproblems are least squares problems, expensive matrix inversions are often necessary. Even worse, the convergence of ADM with more than two variables is not guaranteed [7].

To avoid introducing auxiliary variables and still solve subproblems (4) and (5) efficiently, inspired by Yang et al. [20], we propose a linearization technique for (4) and (5). To further accelerate the convergence of the algorithm, we also propose an adaptive rule for updating the penalty parameter.

2.2 Linearized ADM

By linearizing the quadratic term in (4) at $\mathbf{x}_k$ and adding a proximal term, we have the following approximation:

$$\begin{aligned}
&\mathbf{x}_{k+1} = \arg\min_{\mathbf{x}} f(\mathbf{x}) + \langle \mathcal{A}^*(\lambda_k) + \beta\mathcal{A}^*(\mathcal{A}(\mathbf{x}_k) + \mathcal{B}(\mathbf{y}_k) - \mathbf{c}), \mathbf{x} - \mathbf{x}_k\rangle + \tfrac{\beta\eta_A}{2}\|\mathbf{x} - \mathbf{x}_k\|^2 \\
&= \arg\min_{\mathbf{x}} f(\mathbf{x}) + \frac{\beta\eta_A}{2}\|\mathbf{x} - \mathbf{x}_k + \mathcal{A}^*(\lambda_k + \beta(\mathcal{A}(\mathbf{x}_k) + \mathcal{B}(\mathbf{y}_k) - \mathbf{c}))/(\beta\eta_A)\|^2,
\end{aligned} \qquad (8)$$

where $\mathcal{A}^*$ is the adjoint of $\mathcal{A}$ and $\eta_A > 0$ is a parameter whose proper value will be analyzed later. The above approximation resembles that of APG [16], but we do not use APG to solve (4) iteratively.

Similarly, subproblem (5) can be approximated by

$$\mathbf{y}_{k+1} = \arg\min_{\mathbf{y}} g(\mathbf{y}) + \frac{\beta\eta_B}{2}\|\mathbf{y} - \mathbf{y}_k + \mathcal{B}^*(\lambda_k + \beta(\mathcal{A}(\mathbf{x}_{k+1}) + \mathcal{B}(\mathbf{y}_k) - \mathbf{c}))/(\beta\eta_B)\|^2. \qquad (9)$$

The update of Lagrange multiplier still goes as (6)[2].

2.3 Adaptive Penalty

In previous ADM and LADM approaches [15, 21, 20], the penalty parameter β is fixed. Some scholars have observed that ADM with a fixed β can converge very slowly and it is nontrivial to choose an optimal fixed β. So is LADM. Thus a dynamic β is preferred in real applications. Although Tao et al. [15] and Yang et al. [20] mentioned He et al.'s adaptive updating rule [8] in their papers, the rule is for ADM only. We propose the following adaptive updating strategy for the penalty parameter β:

$$\beta_{k+1} = \min(\beta_{\max}, \rho\beta_k), \qquad (10)$$

[2] As in [20], we can also introduce a parameter γ and update λ as $\lambda_{k+1} = \lambda_k + \gamma\beta[\mathcal{A}(\mathbf{x}_{k+1}) + \mathcal{B}(\mathbf{y}_{k+1}) - \mathbf{c}]$. We choose not to do so in this paper in order not to make the exposition of LADMAP too complex. The readers can refer to Supplementary Material for full details.

where $\beta_{\max}$ is an upper bound of $\{\beta_k\}$. The value of ρ is defined as

$$\rho = \begin{cases} \rho_0, & \text{if } \beta_k \max(\sqrt{\eta_A}\|\mathbf{x}_{k+1}-\mathbf{x}_k\|, \sqrt{\eta_B}\|\mathbf{y}_{k+1}-\mathbf{y}_k\|)/\|\mathbf{c}\| < \varepsilon_2, \\ 1, & \text{otherwise,} \end{cases} \tag{11}$$

where $\rho_0 \geq 1$ is a constant. The condition to assign $\rho = \rho_0$ comes from the analysis on the stopping criteria (see Section 2.5). We recommend that $\beta_0 = \alpha\varepsilon_2$, where α depends on the size of $\mathbf{c}$. Our updating rule is fundamentally different from He et al.'s for ADM [8], which aims at balancing the errors in the stopping criteria and involves several parameters.

2.4 Convergence of LADMAP

To prove the convergence of LADMAP, we first have the following propositions.

Proposition 1

$$-\beta_k\eta_A(\mathbf{x}_{k+1}-\mathbf{x}_k)-\mathcal{A}^*(\tilde{\lambda}_{k+1}) \in \partial f(\mathbf{x}_{k+1}),\ -\beta_k\eta_B(\mathbf{y}_{k+1}-\mathbf{y}_k)-\mathcal{B}^*(\hat{\lambda}_{k+1}) \in \partial g(\mathbf{y}_{k+1}), \tag{12}$$

where $\tilde{\lambda}_{k+1} = \lambda_k + \beta_k[\mathcal{A}(\mathbf{x}_k)+\mathcal{B}(\mathbf{y}_k)-\mathbf{c}]$, $\hat{\lambda}_{k+1} = \lambda_k + \beta_k[\mathcal{A}(\mathbf{x}_{k+1})+\mathcal{B}(\mathbf{y}_k)-\mathbf{c}]$, *and* ∂f *and* ∂g *are subgradients of* f *and* g, *respectively.*

This can be easily proved by checking the optimality conditions of (8) and (9).

Proposition 2 *Denote the operator norms of* $\mathcal{A}$ *and* $\mathcal{B}$ *as* $\|\mathcal{A}\|$ *and* $\|\mathcal{B}\|$, *respectively. If* $\{\beta_k\}$ *is non-decreasing and upper bounded,* $\eta_A > \|\mathcal{A}\|^2$, $\eta_B > \|\mathcal{B}\|^2$, *and* $(\mathbf{x}^*, \mathbf{y}^*, \lambda^*)$ *is any Karush-Kuhn-Tucker (KKT) point of problem (1) (see (13)-(14)), then:* ***(1).*** $\{\eta_A\|\mathbf{x}_k-\mathbf{x}^*\|^2 - \|\mathcal{A}(\mathbf{x}_k-\mathbf{x}^*)\|^2 + \eta_B\|\mathbf{y}_k-\mathbf{y}^*\|^2 + \beta_k^{-2}\|\lambda_k-\lambda^*\|^2\}$ *is non-increasing.* ***(2).*** $\|\mathbf{x}_{k+1}-\mathbf{x}_k\| \to 0$, $\|\mathbf{y}_{k+1}-\mathbf{y}_k\| \to 0$, $\|\lambda_{k+1}-\lambda_k\| \to 0$.

The proof can be found in Supplementary Material. Then we can prove the convergence of LADMAP, as stated in the following theorem.

Theorem 3 *If* $\{\beta_k\}$ *is non-decreasing and upper bounded,* $\eta_A > \|\mathcal{A}\|^2$, *and* $\eta_B > \|\mathcal{B}\|^2$, *then the sequence* $\{(\mathbf{x}_k, \mathbf{y}_k, \lambda_k)\}$ *generated by LADMAP converges to a KKT point of problem (1).*

The proof can be found in Appendix A.

2.5 Stopping Criteria

The KKT conditions of problem (1) are that there exists a triple $(\mathbf{x}^*, \mathbf{y}^*, \lambda^*)$ such that

$$\mathcal{A}(\mathbf{x}^*) + \mathcal{B}(\mathbf{y}^*) - \mathbf{c} = \mathbf{0}, \tag{13}$$

$$-\mathcal{A}^*(\lambda^*) \in \partial f(\mathbf{x}^*), -\mathcal{B}^*(\lambda^*) \in \partial g(\mathbf{y}^*). \tag{14}$$

The triple $(\mathbf{x}^*, \mathbf{y}^*, \lambda^*)$ is called a KKT point. So the first stopping criterion is the feasibility:

$$\|\mathcal{A}(\mathbf{x}_{k+1}) + \mathcal{B}(\mathbf{y}_{k+1}) - \mathbf{c}\|/\|\mathbf{c}\| < \varepsilon_1. \tag{15}$$

As for the second KKT condition, we rewrite the second part of Proposition 1 as follows

$$-\beta_k[\eta_B(\mathbf{y}_{k+1}-\mathbf{y}_k) + \mathcal{B}^*(\mathcal{A}(\mathbf{x}_{k+1}-\mathbf{x}_k))] - \mathcal{B}^*(\tilde{\lambda}_{k+1}) \in \partial g(\mathbf{y}_{k+1}). \tag{16}$$

So for $\tilde{\lambda}_{k+1}$ to satisfy the second KKT condition, both $\beta_k\eta_A\|\mathbf{x}_{k+1}-\mathbf{x}_k\|$ and $\beta_k\|\eta_B(\mathbf{y}_{k+1}-\mathbf{y}_k) + \mathcal{B}^*(\mathcal{A}(\mathbf{x}_{k+1}-\mathbf{x}_k))\|$ should be small enough. This leads to the second stopping criterion:

$$\beta_k \max(\eta_A\|\mathbf{x}_{k+1}-\mathbf{x}_k\|/\|\mathcal{A}^*(\mathbf{c})\|, \eta_B\|\mathbf{y}_{k+1}-\mathbf{y}_k\|/\|\mathcal{B}^*(\mathbf{c})\|) \leq \varepsilon_2'. \tag{17}$$

By estimating $\|\mathcal{A}^*(\mathbf{c})\|$ and $\|\mathcal{B}^*(\mathbf{c})\|$ by $\sqrt{\eta_A}\|\mathbf{c}\|$ and $\sqrt{\eta_B}\|\mathbf{c}\|$, respectively, we arrive at the second stopping criterion in use:

$$\beta_k \max(\sqrt{\eta_A}\|\mathbf{x}_{k+1}-\mathbf{x}_k\|, \sqrt{\eta_B}\|\mathbf{y}_{k+1}-\mathbf{y}_k\|)/\|\mathbf{c}\| \leq \varepsilon_2. \tag{18}$$

Finally, we summarize our LADMAP algorithm in Algorithm 1.

Algorithm 1 LADMAP for Problem (1)

Initialize: Set $\varepsilon_1 > 0, \varepsilon_2 > 0, \beta_{\max} \gg \beta_0 > 0, \eta_A > \|\mathcal{A}\|^2, \eta_B > \|\mathcal{B}\|^2, \mathbf{x}_0, \mathbf{y}_0, \lambda_0$, and $k \leftarrow 0$.
while (15) or (18) is not satisfied **do**
  Step 1: Update $\mathbf{x}$ by solving (8).
  Step 2: Update $\mathbf{y}$ by solving (9).
  Step 3: Update λ by (6).
  Step 4: Update β by (10) and (11).
  Step 5: $k \leftarrow k + 1$.
end while

3 Applying LADMAP to LRR

In this section, we apply LADMAP to solve the LRR problem (2). We further introduce acceleration tricks to reduce the computation complexity of each iteration.

3.1 Solving LRR by LADMAP

As the LRR problem (2) is a special case of problem (1), PADM can be directly applied to it. The two subproblems both have closed form solutions. In the subproblem for updating $\mathbf{E}$, one may apply the $l_{2,1}$-norm shrinkage operator [12], with a threshold β_k^{-1}, to matrix $\mathbf{M}_k = -\mathbf{X}\mathbf{Z}_k + \mathbf{X} - \mathbf{\Lambda}_k/\beta_k$. In the subproblem for updating $\mathbf{Z}$, one has to apply the singular value shrinkage operator [2], with a threshold $(\beta_k\eta_X)^{-1}$, to matrix $\mathbf{N}_k = \mathbf{Z}_k - \eta_X^{-1}\mathbf{X}^T(\mathbf{X}\mathbf{Z}_k + \mathbf{E}_{k+1} - \mathbf{X} + \mathbf{\Lambda}_k/\beta_k)$, where $\eta_X > \sigma_{\max}^2(\mathbf{X})$. If $\mathbf{N}_k$ is formed explicitly, the usual technique of partial SVD, using PROPACK [9] and rank prediction[3], can be utilized to compute the leading r singular values and associated vectors of $\mathbf{N}_k$ efficiently, making the complexity of SVD computation $O(rn^2)$, where r is the predicted rank of $\mathbf{Z}_{k+1}$ and n is the column number of $\mathbf{X}$. Note that as β_k is non-decreasing, the predicted rank is almost non-decreasing, making the iterations computationally efficient.

3.2 Acceleration Tricks for LRR

Up to now, LADMAP for LRR is still of complexity $O(n^3)$, although partial SVD is already used. This is because forming $\mathbf{M}_k$ and $\mathbf{N}_k$ requires full sized matrix-matrix multiplications, e.g., $\mathbf{X}\mathbf{Z}_k$. To break this complexity bound, we introduce a decomposition technique to further accelerate LADMAP for LRR. By representing $\mathbf{Z}_k$ as its skinny SVD: $\mathbf{Z}_k = \mathbf{U}_k\mathbf{\Sigma}_k\mathbf{V}_k^T$, some of the full sized matrix-matrix multiplications are gone: they are replaced by successive reduced sized matrix-matrix multiplications. For example, when updating $\mathbf{E}$, $\mathbf{X}\mathbf{Z}_k$ is computed as $((\mathbf{X}\mathbf{U}_k)\mathbf{\Sigma}_k)\mathbf{V}_k^T$, reducing the complexity to $O(rn^2)$. When computing the partial SVD of $\mathbf{N}_k$, things are more complicated. If we form $\mathbf{N}_k$ explicitly, we will face with computing $\mathbf{X}^T(\mathbf{X} + \mathbf{\Lambda}_k/\beta_k)$, which is neither low-rank nor sparse[4]. Fortunately, in PROPACK the bi-diagonalizing process of $\mathbf{N}_k$ is done by the Lanczos procedure [9], which only requires to compute matrix-vector multiplications $\mathbf{N}_k\mathbf{v}$ and $\mathbf{u}^T\mathbf{N}_k$, where $\mathbf{u}$ and $\mathbf{v}$ are some vectors in the Lanczos procedure. So we may compute $\mathbf{N}_k\mathbf{v}$ and $\mathbf{u}^T\mathbf{N}_k$ by multiplying the vectors $\mathbf{u}$ and $\mathbf{v}$ successively with the component matrices in $\mathbf{N}_k$, rather than forming $\mathbf{N}_k$ explicitly. So the computation complexity of partial SVD of $\mathbf{N}_k$ is still $O(rn^2)$. Consequently, with our acceleration techniques, the complexity of our accelerated LADMAP (denoted as LADMAP(A) for short) for LRR is $O(rn^2)$. LADMAP(A) is summarized in Algorithm 2.

[3]The current PROPACK can only output a given number of singular values and vectors. So one has to predict the number of singular values that are greater than a threshold [11, 20, 16]. See step 3 of Algorithm 2. Recently, we have modified PROPACK so that it can output the singular values that are greater than a threshold and their corresponding singular vectors. See [10].

[4]When forming $\mathbf{N}_k$ explicitly, $\mathbf{X}^T\mathbf{X}\mathbf{Z}_k$ can be computed as $((\mathbf{X}^T(\mathbf{X}\mathbf{U}_k))\mathbf{\Sigma}_k)\mathbf{V}_k^T$, whose complexity is still $O(rn^2)$, while $\mathbf{X}^T\mathbf{E}_{k+1}$ could also be accelerated as $\mathbf{E}_{k+1}$ is a column-sparse matrix.

Algorithm 2 Accelerated LADMAP for LRR (2)

Input: Observation matrix $\mathbf{X}$ and parameter $\mu > 0$.
Initialize: Set $\mathbf{E}_0$, $\mathbf{Z}_0$ and $\mathbf{\Lambda}_0$ to zero matrices, where $\mathbf{Z}_0$ is represented as $(\mathbf{U}_0, \mathbf{\Sigma}_0, \mathbf{V}_0) \leftarrow (\mathbf{0}, \mathbf{0}, \mathbf{0})$. Set $\varepsilon_1 > 0$, $\varepsilon_2 > 0$, $\beta_{\max} \gg \beta_0 > 0$, $\eta_X > \sigma^2_{\max}(\mathbf{X})$, $r = 5$, and $k \leftarrow 0$.
while (15) or (18) is not satisfied **do**
Step 1: Update $\mathbf{E}_{k+1} = \arg\min_{\mathbf{E}} \mu\|\mathbf{E}\|_{2,1} + \frac{\beta_k}{2}\|\mathbf{E} + (\mathbf{X}\mathbf{U}_k)\mathbf{\Sigma}_k\mathbf{V}_k^T - \mathbf{X} + \mathbf{\Lambda}_k/\beta_k\|^2$. This subproblem can be solved by using Lemma 3.2 in [12].
Step 2: Update the skinny SVD $(\mathbf{U}_{k+1}, \mathbf{\Sigma}_{k+1}, \mathbf{V}_{k+1})$ of $\mathbf{Z}_{k+1}$. First, compute the partial SVD $\tilde{\mathbf{U}}_r\tilde{\mathbf{\Sigma}}_r\tilde{\mathbf{V}}_r^T$ of the *implicit* matrix $\mathbf{N}_k$, which is bi-diagonalized by the successive matrix-vector multiplication technique described in Section 3.1. Second, $\mathbf{U}_{k+1} = \tilde{\mathbf{U}}_r(:, 1 : r')$, $\mathbf{\Sigma}_{k+1} = \tilde{\mathbf{\Sigma}}_r(1 : r', 1 : r') - (\beta_k\eta_X)^{-1}\mathbf{I}$, $\mathbf{V}_{k+1} = \tilde{\mathbf{V}}_r(:, 1 : r')$, where r' is the number of singular values in $\mathbf{\Sigma}_r$ that are greater than $(\beta_k\eta_X)^{-1}$.
Step 3: Update the predicted rank r:
If $r' < r$, then $r = \min(r' + 1, n)$; otherwise, $r = \min(r' + \text{round}(0.05n), n)$.
Step 4: Update $\mathbf{\Lambda}_{k+1} = \mathbf{\Lambda}_k + \beta_k((\mathbf{X}\mathbf{U}_{k+1})\mathbf{\Sigma}_{k+1}\mathbf{V}_{k+1}^T + \mathbf{E}_{k+1} - \mathbf{X})$.
Step 5: Update β_{k+1} by (10)-(11).
Step 6: $k \leftarrow k + 1$.
end while

4 Experimental Results

In this section, we report numerical results on LADMAP, LADMAP(A) and other state-of-the-art algorithms, including APG[5], ADM[6] and LADM, for LRR based data clustering problems. APG, ADM, LADM and LADMAP all utilize the Matlab version of PROPACK [9]. For LADMAP(A), we provide two function handles to PROPACK which fulfils the successive matrix-vector multiplications. All experiments are run and timed on a PC with an Intel Core i5 CPU at 2.67GHz and with 4GB of memory, running Windows 7 and Matlab version 7.10.

We test and compare these solvers on both synthetic multiple subspaces data and the real world motion data (Hopkin155 motion segmentation database [17]). For APG, we set the parameters $\beta_0 = 0.01$, $\beta_{\min} = 10^{-10}$, $\theta = 0.9$ in its continuation technique and the Lipschitz constant $\tau = \sigma^2_{\max}(\mathbf{X})$. The parameters of ADM and LADM are the same as those in [12] and [20], respectively. In particular, for LADM the penalty is fixed at $\beta = 2.5/\min(m, n)$, where $m \times n$ is the size of $\mathbf{X}$. For LADMAP, we set $\varepsilon_1 = 10^{-4}$, $\varepsilon_2 = 10^{-5}$, $\beta_0 = \min(m, n)\varepsilon_2$, $\beta_{\max} = 10^{10}$, $\rho_0 = 1.9$, and $\eta_X = 1.02\sigma^2_{\max}(\mathbf{X})$. As the code of ADM was downloaded, its stopping criteria, $\|\mathbf{X}\mathbf{Z}_k + \mathbf{E}_k - \mathbf{X}\|/\|\mathbf{X}\| \le \varepsilon_1$ and $\max(\|\mathbf{E}_k - \mathbf{E}_{k-1}\|/\|\mathbf{X}\|, \|\mathbf{Z}_k - \mathbf{Z}_{k-1}\|/\|\mathbf{X}\|) \le \varepsilon_2$, are used in all our experiments[7].

4.1 On Synthetic Data

The synthetic test data, parameterized as (s, p, d, $\tilde{r}$), is created by the same procedure in [12]. s independent subspaces $\{\mathcal{S}_i\}_{i=1}^s$ are constructed, whose bases $\{\mathbf{U}_i\}_{i=1}^s$ are generated by $\mathbf{U}_{i+1} = \mathbf{T}\mathbf{U}_i,\ 1 \le i \le s-1$, where $\mathbf{T}$ is a random rotation and $\mathbf{U}_1$ is a $d \times \tilde{r}$ random orthogonal matrix. So each subspace has a rank of $\tilde{r}$ and the data has an ambient dimension of d. Then p data points are sampled from each subspace by $\mathbf{X}_i = \mathbf{U}_i\mathbf{Q}_i,\ 1 \le i \le s$, with $\mathbf{Q}_i$ being an $\tilde{r} \times p$ i.i.d. zero mean unit variance Gaussian matrix $\mathcal{N}(0, 1)$. 20% samples are randomly chosen to be corrupted by adding Gaussian noise with zero mean and standard deviation $0.1\|\mathbf{x}\|$. We empirically find that LRR achieves the best clustering performance on this data set when $\mu = 0.1$. So we test all algorithms with $\mu = 0.1$ in this experiment. To measure the relative errors in the solutions, we run LADMAP 2000 iterations with $\beta_{\max} = 10^3$ to establish the ground truth solution $(\mathbf{E}_0, \mathbf{Z}_0)$.

The computational comparison is summarized in Table 1. We can see that the iteration numbers and the CPU times of both LADMAP and LADMAP(A) are much less than those of other methods, and

[5]Please see Supplementary Material for the detail of solving LRR by APG.

[6]We use the Matlab code provided online by the authors of [12].

[7]Note that the second criterion differs from that in (18). However, this does not harm the convergence of LADMAP because (18) is always checked when updating β_{k+1} (see (11)).

LADMAP(A) is further much faster than LADMAP. Moreover, the advantage of LADMAP(A) is even greater when the ratio $\tilde{r}/p$, which is roughly the ratio of the rank of $\mathbf{Z}_0$ to the size of $\mathbf{Z}_0$, is smaller, which testifies to the complexity estimations on LADMAP and LADMAP(A) for LRR. It is noteworthy that the iteration numbers of ADM and LADM seem to grow with the problem sizes, while that of LADMAP is rather constant. Moreover, LADM is not faster than ADM. In particular, on the last data we were unable to wait until LADM stopped. Finally, as APG converges to an approximate solution to (2), its relative errors are larger and its clustering accuracy is lower than ADM and LADM based methods.

Table 1: Comparison among APG, ADM, LADM, LADMAP and LADMAP(A) on the synthetic data. For each quadruple (s, p, d, $\tilde{r}$), the LRR problem, with $\mu = 0.1$, was solved for the same data using different algorithms. We present typical running time (in $\times 10^3$ seconds), iteration number, relative error (%) of output solution $(\hat{\mathbf{E}}, \hat{\mathbf{Z}})$ and the clustering accuracy (%) of tested algorithms, respectively.

Size (s, p, d, $\tilde{r}$)	Method	Time	Iter.	$\frac{\|\hat{\mathbf{Z}}-\mathbf{Z}_0\|}{\|\mathbf{Z}_0\|}$	$\frac{\|\hat{\mathbf{E}}-\mathbf{E}_0\|}{\|\mathbf{E}_0\|}$	Acc.
(10, 20,200, 5)	APG	0.0332	110	2.2079	1.5096	81.5
	ADM	0.0529	176	0.5491	0.5093	**90.0**
	LADM	0.0603	194	**0.5480**	**0.5024**	**90.0**
	LADMAP	0.0145	**46**	**0.5480**	**0.5024**	**90.0**
	LADMAP(A)	**0.0010**	**46**	**0.5480**	**0.5024**	**90.0**
(15, 20,300, 5)	APG	0.0869	106	2.4824	1.0341	80.0
	ADM	0.1526	185	0.6519	0.4078	83.7
	LADM	0.2943	363	**0.6518**	**0.4076**	**86.7**
	LADMAP	0.0336	**41**	**0.6518**	**0.4076**	**86.7**
	LADMAP(A)	**0.0015**	**41**	**0.6518**	**0.4076**	**86.7**
(20, 25, 500, 5)	APG	1.8837	117	2.8905	2.4017	72.4
	ADM	3.7139	225	1.1191	1.0170	80.0
	LADM	8.1574	508	**0.6379**	**0.4268**	80.0
	LADMAP	0.7762	**40**	**0.6379**	**0.4268**	**84.6**
	LADMAP(A)	**0.0053**	**40**	**0.6379**	**0.4268**	**84.6**
(30, 30, 900, 5)	APG	6.1252	116	3.0667	0.9199	69.4
	ADM	11.7185	220	0.6865	0.4866	**76.0**
	LADM	N.A.	N.A.	N.A.	N.A.	N.A.
	LADMAP	2.3891	**44**	**0.6864**	**0.4294**	**80.1**
	LADMAP(A)	**0.0058**	**44**	**0.6864**	**0.4294**	**80.1**

Table 2: Comparison among APG, ADM, LADM, LADMAP and LADMAP(A) on the Hopkins155 database. We present their average computing time (in seconds), average number of iterations and average classification errors (%) on all 156 sequences.

	Two Motion			Three Motion			All		
	Time	Iter.	CErr.	Time	Iter.	CErr.	Time	Iter.	CErr.
APG	15.7836	90	5.77	46.4970	90	**16.52**	22.6277	90	8.36
ADM	53.3470	281	**5.72**	159.8644	284	**16.52**	77.0864	282	**8.33**
LADM	9.6701	110	5.77	22.1467	64	**16.52**	12.4520	99	8.36
LADMAP	3.6964	**22**	**5.72**	10.9438	**22**	**16.52**	5.3114	**22**	**8.33**
LADMAP(A)	**2.1348**	**22**	**5.72**	**6.1098**	**22**	**16.52**	**3.0202**	**22**	**8.33**

4.2 On Real World Data

We further test the performance of these algorithms on the Hopkins155 database [17]. This database consists of 156 sequences, each of which has 39 to 550 data vectors drawn from two or three motions. For computational efficiency, we preprocess the data by projecting it to be 5-dimensional using PCA. As $\mu = 2.4$ is the best parameter for this database [12], we test all algorithms with $\mu = 2.4$.

Table 2 shows the comparison among APG, ADM, LADM, LADMAP and LADMAP(A) on this database. We can also see that LADMAP and LADMAP(A) are much faster than APG, ADM, and

LADM, and LADMAP(A) is also faster than LADMAP. However, in this experiment the advantage of LADMAP(A) over LADMAP is not as dramatic as that in Table 1. This is because on this data μ is chosen as 2.4, which cannot make the rank of the ground truth solution $\mathbf{Z}_0$ much smaller than the size of $\mathbf{Z}_0$.

5 Conclusions

In this paper, we propose a linearized alternating direction method with adaptive penalty for solving subproblems in ADM conveniently. With LADMAP, no auxiliary variables are required and the convergence is also much faster. We further apply it to solve the LRR problem and combine it with an acceleration trick so that the computation complexity is reduced from $O(n^3)$ to $O(rn^2)$, which is highly advantageous over the existing LRR solvers. Although we only present results on LRR, LADMAP is actually a general method that can be applied to other convex programs.

Acknowledgments

The authors would like to thank Dr. Xiaoming Yuan for pointing us to [20]. This work is partially supported by the grants of the NSFC-Guangdong Joint Fund (No. U0935004) and the NSFC Fund (No. 60873181, 61173103). R. Liu also thanks the support from CSC.

A Proof of Theorem 3

Proof By Proposition 2 (1), $\{(\mathbf{x}_k, \mathbf{y}_k, \lambda_k)\}$ is bounded, hence has an accumulation point, say $(\mathbf{x}_{k_j}, \mathbf{y}_{k_j}, \lambda_{k_j}) \to (\mathbf{x}^\infty, \mathbf{y}^\infty, \lambda^\infty)$. We accomplish the proof in two steps.

1. We first prove that $(\mathbf{x}^\infty, \mathbf{y}^\infty, \lambda^\infty)$ is a KKT point of problem (1).

By Proposition 2 (2), $\mathcal{A}(\mathbf{x}_{k+1}) + \mathcal{B}(\mathbf{y}_{k+1}) - \mathbf{c} = \beta_k^{-1}(\lambda_{k+1} - \lambda_k) \to 0$. This shows that any accumulation point of $\{(\mathbf{x}_k, \mathbf{y}_k)\}$ is a feasible solution.

By letting $k = k_j - 1$ in Proposition 1 and the definition of subgradient, we have

$$\begin{aligned} f(\mathbf{x}_{k_j}) + g(\mathbf{y}_{k_j}) \leq \quad & f(\mathbf{x}^*) + g(\mathbf{y}^*) + \langle \mathbf{x}_{k_j} - \mathbf{x}^*, -\beta_{k_j-1}\eta_A(\mathbf{x}_{k_j} - \mathbf{x}_{k_j-1}) - \mathcal{A}^*(\tilde{\lambda}_{k_j})\rangle \\ & + \langle \mathbf{y}_{k_j} - \mathbf{y}^*, -\beta_{k_j-1}\eta_B(\mathbf{y}_{k_j} - \mathbf{y}_{k_j-1}) - \mathcal{B}^*(\hat{\lambda}_{k_j})\rangle. \end{aligned}$$

Let $j \to +\infty$, by observing Proposition 2 (2), we have

$$\begin{aligned} f(\mathbf{x}^\infty) + g(\mathbf{y}^\infty) \quad & \leq f(\mathbf{x}^*) + g(\mathbf{y}^*) + \langle \mathbf{x}^\infty - \mathbf{x}^*, -\mathcal{A}^*(\lambda^\infty)\rangle + \langle \mathbf{y}^\infty - \mathbf{y}^*, -\mathcal{B}^*(\lambda^\infty)\rangle \\ & = f(\mathbf{x}^*) + g(\mathbf{y}^*) - \langle \mathcal{A}(\mathbf{x}^\infty - \mathbf{x}^*), \lambda^\infty\rangle - \langle \mathcal{B}(\mathbf{y}^\infty - \mathbf{y}^*), \lambda^\infty)\rangle \\ & = f(\mathbf{x}^*) + g(\mathbf{y}^*) - \langle \mathcal{A}(\mathbf{x}^\infty) + \mathcal{B}(\mathbf{y}^\infty) - \mathcal{A}(\mathbf{x}^*) - \mathcal{B}(\mathbf{y}^*), \lambda^\infty\rangle \\ & = f(\mathbf{x}^*) + g(\mathbf{y}^*), \end{aligned}$$

where we have used the fact that both $(\mathbf{x}^\infty, \mathbf{y}^\infty)$ and $(\mathbf{x}^*, \mathbf{y}^*)$ are feasible solutions. So we conclude that $(\mathbf{x}^\infty, \mathbf{y}^\infty)$ is an optimal solution to (1).

Again, let $k = k_j - 1$ in Proposition 1 and by the definition of subgradient, we have

$$f(\mathbf{x}) \geq f(\mathbf{x}_{k_j}) + \langle \mathbf{x} - \mathbf{x}_{k_j}, -\beta_{k_j-1}\eta_A(\mathbf{x}_{k_j} - \mathbf{x}_{k_j-1}) - \mathcal{A}^*(\tilde{\lambda}_{k_j})\rangle, \ \forall \mathbf{x}. \tag{19}$$

Fix $\mathbf{x}$ and let $j \to +\infty$, we see that

$$f(\mathbf{x}) \geq f(\mathbf{x}^\infty) + \langle \mathbf{x} - \mathbf{x}^\infty, -\mathcal{A}^*(\lambda^\infty)\rangle, \ \forall \mathbf{x}.$$

So $-\mathcal{A}^*(\lambda^\infty) \in \partial f(\mathbf{x}^\infty)$. Similarly, $-\mathcal{B}^*(\lambda^\infty) \in \partial g(\mathbf{y}^\infty)$. Therefore, $(\mathbf{x}^\infty, \mathbf{y}^\infty, \lambda^\infty)$ is a KKT point of problem (1).

2. We next prove that the whole sequence $\{(\mathbf{x}_k, \mathbf{y}_k, \lambda_k)\}$ converges to $(\mathbf{x}^\infty, \mathbf{y}^\infty, \lambda^\infty)$.

By choosing $(\mathbf{x}^*, \mathbf{y}^*, \lambda^*) = (\mathbf{x}^\infty, \mathbf{y}^\infty, \lambda^\infty)$ in Proposition 2, we have $\eta_A\|\mathbf{x}_{k_j} - \mathbf{x}^\infty\|^2 - \|\mathcal{A}(\mathbf{x}_{k_j} - \mathbf{x}^\infty)\|^2 + \eta_B\|\mathbf{y}_{k_j} - \mathbf{y}^\infty\|^2 + \beta_{k_j}^{-2}\|\lambda_{k_j} - \lambda^\infty\|^2 \to 0$. By Proposition 2 (1), we readily have $\eta_A\|\mathbf{x}_k - \mathbf{x}^\infty\|^2 - \|\mathcal{A}(\mathbf{x}_k - \mathbf{x}^\infty)\|^2 + \eta_B\|\mathbf{y}_k - \mathbf{y}^\infty\|^2 + \beta_k^{-2}\|\lambda_k - \lambda^\infty\|^2 \to 0$. So $(\mathbf{x}_k, \mathbf{y}_k, \lambda_k) \to (\mathbf{x}^\infty, \mathbf{y}^\infty, \lambda^\infty)$.

As $(\mathbf{x}^\infty, \mathbf{y}^\infty, \lambda^\infty)$ can be an arbitrary accumulation point of $\{(\mathbf{x}_k, \mathbf{y}_k, \lambda_k)\}$, we may conclude that $\{(\mathbf{x}_k, \mathbf{y}_k, \lambda_k)\}$ converges to a KKT point of problem (1).

References

[1] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. In Michael Jordan, editor, *Foundations and Trends in Machine Learning*, 2010.

[2] J. Cai, E. Candès, and Z. Shen. A singular value thresholding algorithm for matrix completion. *preprint*, 2008.

[3] E. Candès, X. Li, Y. Ma, and J. Wright. Robust principal component analysis? *J. ACM*, 2011.

[4] E. Candès and B. Recht. Exact matrix completion via convex optimization. *Foundations of Computational Mathematics*, 2009.

[5] E. J. Candès and M. Wakin. An introduction to compressive sampling. *IEEE Signal Processing Magazine*, 2008.

[6] P. Favaro, R. Vidal, and A. Ravichandran. A closed form solution to robust subspace estimation and clustering. In *CVPR*, 2011.

[7] B. He, M. Tao, and X. Yuan. Alternating direction method with Gaussian back substitution for separable convex programming. *SIAM Journal on Optimization*, accepted.

[8] B. He, H. Yang, and S. Wang. Alternating direction method with self-adaptive penalty parameters for monotone variational inequality. *J. Optimization Theory and Applications*, 106:337–356, 2000.

[9] R. Larsen. Lanczos bidiagonalization with partial reorthogonalization. *Department of Computer Science, Aarhus University, Technical report, DAIMI PB-357*, 1998.

[10] Z. Lin. Some software packages for partial SVD computation. arXiv:1108.1548.

[11] Z. Lin, M. Chen, and Y. Ma. The augmented Lagrange multiplier method for exact recovery of corrupted low-rank matrices. *UIUC Technical Report UILU-ENG-09-2215*, 2009, arXiv:1009.5055.

[12] G. Liu, Z. Lin, and Y. Yu. Robust subspace segmentation by low-rank representation. In *ICML*, 2010.

[13] J. Liu, S. Ji, and J. Ye. Multi-task feature learning via efficient $l_{2,1}$ norm minimization. In *UAI*, 2009.

[14] Y. Ni, J. Sun, X. Yuan, S. Yan, and L. Cheong. Robust low-rank subspace segmentation with semidefinite guarantees. In *ICDM Workshop*, 2010.

[15] M. Tao and X.M. Yuan. Recovering low-rank and sparse components of matrices from incomplete and noisy observations. *SIAM Journal on Optimization*, 21(1):57–81, 2011.

[16] K. Toh and S. Yun. An accelerated proximal gradient algorithm for nuclear norm regularized least sequares problems. *Pacific J. Optimization*, 6:615–640, 2010.

[17] R. Tron and R. Vidal. A benchmark for the comparison of 3D montion segmentation algorithms. In *CVPR*, 2007.

[18] J. Wright, A. Ganesh, S. Rao, and Y. Ma. Robust principal component analysis: Exact recovery of corrupted low-rank matrices via convex optimization. In *NIPS*, 2009.

[19] J. Wright, Y. Ma, J. Mairal, G. Sapirao, T. Huang, and S. Yan. Sparse representation for computer vision and pattern recognition. *Proceedings of the IEEE*, 2010.

[20] J. Yang and X. Yuan. Linearized augmented Lagrangian and alternating direction methods for nuclear norm minimization. *submitted*, 2011.

[21] J. Yang and Y. Zhang. Alternating direction algorithms for l_1 problems in compressive sensing. *SIAM J. Scientific Computing*, 2010.

[22] W. Yin. Analysis and generalizations of the linearized Bregman method. *SIAM Journal on Imaging Sciences*, 2010.