Yokohama Publishers
ISSN 1349-8169 ONLINE JOURNAL

Pacific Journal of Optimization
Vol. 9, No. 1, pp.167–180, January 2013

SPARSE AND LOW-RANK MATRIX DECOMPOSITION VIA ALTERNATING DIRECTION METHOD

Xiaoming Yuan* and Junfeng Yang†

Abstract: The problem of recovering the sparse and low-rank components of a given matrix captures a broad spectrum of applications. However, this recovery problem is NP-hard and thus not tractable in general. Recently, it was shown in [3, 6] that this recovery problem can be well approached by solving a convex relaxation problem where the ℓ_1-norm and the nuclear norm are used to induce sparse and low-rank structures, respectively. Generic interior-point solvers can be applied to solve a semi-definite programming reformulation of the convex relaxation problem. However, the interior-point approach can only handle small cases on personal computers. In this paper, we propose to use of the alternating direction method (ADM) to solve the convex relaxation problem. We show by numerical results that the ADM approach is easily implementable and computationally efficient for solving the sparse and low-rank matrix decomposition problem.

Key words: *matrix decomposition, sparse, low-rank, alternating direction method, ℓ_1-norm, nuclear norm*

Mathematics Subject Classification: *90C06, 90C22, 90C25, 90C59, 93B30*

1 Introduction

Matrix representations of complex systems and models arising in various areas often have the character that such a matrix is composed of a sparse component and a low-rank component. Such applications include the model selection in statistics, system identification in engineering, partially coherent decomposition in optical systems, and matrix rigidity in computer science, see e.g. [16, 17, 18, 33, 36, 39, 44]. Practically, it is of significant interest to take advantage of the decomposable character of such a complex system. One necessary way towards this goal is to recover the sparse and low-rank components of a given matrix without prior knowledge about the sparsity pattern or the rank information. The problem of sparse and low-rank matrix decomposition (SLRMD) was highlighted and intensively studied recently in [3, 6].

It is known that the SLRMD problem is NP-hard and thus not trackable. In sprit of the influential work of [4, 5, 10, 11, 12, 38] in compressed sensing and statistics, the authors of [6] proposed the concept of *rank-sparsity incoherence* to characterize the identifiability of the recovery of sparse and low-rank components. Accordingly, a simple deterministic

*Corresponding author. This author was supported by Hong Kong General Research Fund No.202610 and National Science Foundation of China No.10701055.

†This author was supported by National Science Foundation of China No. 11001123 and the Fundamental Research Funds for the Central Universities.

ISSN 1348-9151

condition to ensure exact recovery was derived therein. Note that the concept of rank-sparsity incoherence relates algebraically the sparsity pattern of a matrix and its row or column spaces via an uncertainty principle. Throughout this paper, we assume by default that the matrix under consideration is recoverable for sparse and low-rank components.

The heuristics of using the ℓ_1-norm as the proxy of sparsity and the nuclear norm as the surrogate of low-rank are widely used in many areas such as statistics and image processing (see e.g. [5, 9, 15, 38]). This inspired the authors of [6] to accomplish the sparse and low-rank recovery by solving the following convex programming problem:

$$\min_{A,B} \{\gamma\|A\|_{\ell_1} + \|B\|_* : A + B = C\}, \tag{1.1}$$

where $C \in R^{m\times n}$ is the given matrix to be decomposed; $A, B \in R^{m\times n}$ represent, respectively, the sparse and the low-rank components of C; $\|\cdot\|_{\ell_1}$ is the ℓ_1 norm defined by the component-wise sum of absolute values of all entries; $\|\cdot\|_*$ is the nuclear norm defined by the sum of all singular values; and $\gamma > 0$ is a trade-off constant for the sparse and the low-rank components. Let A^* and B^* be the true sparse and low-rank components of C which are to be recovered. Then, some sufficient conditions on A^* and B^* were proposed in [6] to ensure that the unique solution of (1.1) is exactly (A^*, B^*) for a range of γ. We refer to [6, Theorem 2] for the delineation of these conditions.

Therefore, efficient solvers for (1.1) become crucial for the task of recovering the sparse and the low-rank components of C. In particular, it was suggested in [6] that generic interior-point solvers such as SDPT3 [43] can be applied to solve the semi-definite programming (SDP) reformulation of (1.1). For many cases, however, the matrices to be decomposed are large-scale in dimensions, and thus the problem is beyond the applicable range of generic interior-point codes. In fact, it is well known that on a personal computer, the interior-point methods as implemented in SDPT3 or SeDuMi would have great difficulty in solving a large scale SDP problem whose matrix variable has order larger than 5000 or the number of equality constraints is more than 10000. In particular, for solving (1.1) via the interior-point approach in [6], the dimension of the SDP reformulation is increased to $(m + n) \times (m + n)$ as compared to $m \times n$ in (1.1), see (A.1) in [6].

Hence, just as mentioned in [6], it is of particular interest to develop efficient numerical algorithms for solving (1.1), especially for large-scale cases. We point out that the linearly constrained problem (1.1) is well-structured in the sense that its objective function is the sum of two individual functions without coupled variables. In practice, it is advantageous to exploit this structure in algorithmic design. In fact, the separable structure of (1.1) can be readily exploited by the well known alternating direction method (ADM, [24]). In this paper, we propose to use the ADM approach for solving (1.1) by taking full advantage of its separable structure. As we will analyze in detail, the ADM approach is attractive for (1.1) because the computational of each iteration is dominated by only one singular value decomposition (SVD).

2 The ADM Approach

Roughly speaking, ADM is a practical variant of the classical augmented Lagrangian method (ALM, see e.g., [30, 37]) for solving linearly constrained convex programming problem whose objective function is the sum of two individual functions without coupled variables. The ADM has found applications in many areas including convex programming, variational inequalities and image processing, see, e.g. [1, 8, 13, 14, 19, 20, 21, 22, 23, 26, 31, 42]. In

particular, novel applications of ADM for solving some interesting optimization problems have been discovered very recently, see e.g. [7, 14, 27, 35, 40, 46, 47, 48, 49].

The augmented Lagrangian function of (1.1) is

$$L_{\mathcal{A}}(A, B, Z, \beta) := \gamma\|A\|_{\ell_1} + \|B\|_* - \langle Z, A + B - C\rangle + \frac{\beta}{2}\|A + B - C\|_F^2,$$

where $Z \in R^{m\times n}$ is the Lagrange multiplier of the linear constraint; $\beta > 0$ is a penalty parameter for the violation of the linear constraint, $\langle\cdot,\cdot\rangle$ denotes the standard trace inner product, and $\|\cdot\|_F$ is the induced Frobenius norm. Clearly, the classical ALM is applicable, and its iterative scheme starting from Z^k is given as follows:

$$\begin{cases} (A^{k+1}, B^{k+1}) \leftarrow \arg\min_{A,B} L_{\mathcal{A}}(A, B, Z^k, \beta), \\ Z^{k+1} = Z^k - \beta(A^{k+1} + B^{k+1} - C). \end{cases} \tag{2.1}$$

The direct application of ALM, however, treats (1.1) as a generic minimization problem and performs the minimization with respect to A and B simultaneously.

In contrast, ADM splits the minimization task in (2.1) into two smaller and easier subproblems, where A and B are minimized separately. Specifically, the original ADM (see [21]) solves the following problems to generate the new iterate:

$$A^{k+1} = \arg\min_A L_{\mathcal{A}}(A, B^k, Z^k, \beta), \tag{2.2a}$$

$$B^{k+1} = \arg\min_B L_{\mathcal{A}}(A^{k+1}, B, Z^k, \beta), \tag{2.2b}$$

$$Z^{k+1} = Z^k - \beta(A^{k+1} + B^{k+1} - C). \tag{2.2c}$$

By doing so, the subproblems (2.2a) and (2.2b) both have closed-form solutions. Thus, iterative processes for solving the inner subproblems are avoided. This fact contributes significantly to the computational efficiency of ADM for solving (1.1). We now elaborate on strategies of solving the subproblems (2.2a) and (2.2b). First, problem (2.2a) turns out to be a shrinkage problem (see e.g [41]) and its closed-form solution is given by:

$$A^{k+1} = Z^k/\beta - B^k + C - P_{\Omega_\infty^{\gamma/\beta}}(Z^k/\beta - B^k + C),$$

where $P_{\Omega_\infty^{\gamma/\beta}}(\cdot)$ denotes the Euclidean projection onto the set:

$$\Omega_\infty^{\gamma/\beta} := \{X \in R^{m\times n} \mid -\gamma/\beta \le X_{ij} \le \gamma/\beta\}.$$

On the other hand, it is easy to verify that the problem (2.2b) is equivalent to the following minimization problem:

$$B^{k+1} = \arg\min_B \left\{ \|B\|_* + \frac{\beta}{2}\|B - (C - A^{k+1} + Z^k/\beta)\|_F^2 \right\}. \tag{2.3}$$

Then, according to [2, 34], B^{k+1} admits the following explicit form:

$$B^{k+1} = U^{k+1}\mathrm{diag}(\max\{\sigma_i^{k+1} - 1/\beta, 0\})(V^{k+1})^T, \tag{2.4}$$

where $U^{k+1} \in R^{m\times r}$, $V^{k+1} \in R^{n\times r}$ are obtained via the following SVD:

$$C - A^{k+1} + Z^k/\beta = U^{k+1}\Sigma^{k+1}(V^{k+1})^T \quad \text{with} \quad \Sigma^{k+1} = \mathrm{diag}(\{\sigma_i^{k+1}\}_{i=1}^r).$$

Based on above analysis, we now describe the procedure of applying the ADM to solve (1.1). For given (B^k, Z^k), the ADM takes the following steps to generate the new iterate $(A^{k+1}, B^{k+1}, Z^{k+1})$:

Algorithm: the ADM for SLRMD problem:

Step 1. Generate A^{k+1}:

$$A^{k+1} = Z^k/\beta - B^k + C - P_{\Omega_\infty^{\gamma/\beta}}(Z^k/\beta - B^k + C).$$

Step 2 Generate B^{k+1}:

$$B^{k+1} = U^{k+1}\mathrm{diag}(\max\{\sigma_i^{k+1} - 1/\beta, 0\})(V^{k+1})^T,$$

where U^{k+1}, V^{k+1} and $\{\sigma_i^{k+1}\}$ are generated by the following SVD:

$$C - A^{k+1} + Z^k/\beta = U^{k+1}\Sigma^{k+1}(V^{k+1})^T, \text{ with } \Sigma^{k+1} = \mathrm{diag}(\{\sigma_i^{k+1}\}_{i=1}^r).$$

Step 3. Update the multiplier:

$$Z^{k+1} = Z^k - \beta(A^{k+1} + B^{k+1} - C).$$

Remark 1. It is easy to see that when the ADM approach is applied to solve (1.1), the computation of each iteration is dominated by one SVD with the complexity $O(n^3)$, see e.g. [25]. In particular, existing subroutines for efficient computation of SVD (e.g. [32, 34]) guarantees the efficiency of the proposed ADM approach for sparse and low-rank recovery of large-scale matrices.

Remark 2. Some more general ADM methods are easy to be extended to solve the SLRMD problem. For example, the ADM-based descent method developed in [45]. Since the convergence of ADM type methods have been well studied in the literatures (see e.g. [20, 21, 22, 23, 28, 45]), we omit the convergence analysis of the proposed ADM for (1.1).

3 Numerical Results

In this section, we report some preliminary numerical results to show the efficiency of the proposed ADM for solving (1.1). All codes were written in MATLAB v7.8 (R2009a) and all experiments were performed on a Lenovo laptop with Windows Vista Premium and Intel Core 2 Duo CPU at 1.8 GHz and 2 GB of memory.

Let $C = A^* + B^*$ be the available data, where A^* and B^* are, respectively, the original sparse and low-rank matrices that we wish to recover. For convenience, we let $\gamma = t/(1-t)$ so that problem (1.1) can be equivalently transformed to

$$\min_{A,B} \{t\|A\|_{\ell_1} + (1-t)\|B\|_* : A + B = C\}. \tag{3.1}$$

The advantage of using the reformulation (3.1) is that the varying range of the parameter t in (3.1) can be restricted into the finite interval $(0, 1)$, while the parameter γ should be within $(0, +\infty)$ in (1.1). Let $(\hat{A}_t, \hat{B}_t)$ be an approximate solution of (A^*, B^*) obtained by the proposed ADM, and the quality of recovery be measured by

$$\mathrm{RelErr} := \frac{\|(\hat{A}_t, \hat{B}_t) - (A^*, B^*)\|_F}{\|(A^*, B^*)\|_F}. \tag{3.2}$$

3.1 Experimental Settings

Given a small constant $\tau > 0$, we define

$$\text{diff}_t := \|\hat{A}_t - \hat{A}_{t-\tau}\|_F + \|\hat{B}_t - \hat{B}_{t-\tau}\|_F. \tag{3.3}$$

It is clear that $\hat{B}_t$ approaches the zero matrix (and thus $\hat{A}_t$ approaches C) as $t \downarrow 0$. On the other hand, $\hat{A}_t$ approaches the zero matrix (and thus $\hat{B}_t$ approaches C) as $t \uparrow 1$. Therefore, diff_t is approximately 0 if $\tau > 0$ is sufficiently small and t is close to the boundary of $(0, 1)$. As suggested in [6], a suitable value of t should result in a recovery such that diff_t is stabilized while t stays away from both 0 and 1. To determine a suitable value of t, we set $\tau = 0.01$ in (3.3), which, based on our experimental results, is sufficiently small for measuring the sensitivity of $(\hat{A}_t, \hat{B}_t)$ with respect to t, and ran a set of experiments with different combinations of sparsity ratios of A^* and ranks of B^*. In the following, `r` and `spr` represent, respectively, matrix rank and sparsity ratio. We tested two types of sparse matrices: impulsive and Gaussian sparse matrices. The MATLAB scripts for generating matrix C are as follows:

- `B = randn(m,r)*randn(r,n); mgB = max(abs(B(:)));`
- `A = zeros(m,n); p = randperm(m*n); L = round(spr*m*n);`

  Impulsive sparse matrix: `A(p(1:L)) = mgB.*sign(randn(L,1));`

  Gaussian sparse matrix: `A(p(1:L)) = randn(L,1);`

- $C = A + B$.

More specifically, we set $m = n = 100$ and tested the following cases:

$$(\texttt{r}, \texttt{spr}) = (1, 1\%), (2, 2\%), \ldots, (20, 20\%),$$

for which the decomposition problem is gradually more and more difficult.

Based on our experimental results, suitable values of t shrinks from $[0.05, 0.2]$ to $[0.09, 0.1]$ as (`r, spr`) increases from $(1, 1\%)$ to $(20, 20\%)$. Therefore, we set $t = 0.1$ in our experiments for $m = n = 100$. For $m, n > 100$, we selected t via trial and error. In the following, we present experimental results for the mentioned two types of sparse matrices. In all experiments, we simply set $\beta = 0.25mn/\|C\|_1$ and terminated the iteration whenever the relative change is less then the give tolerance $tol > 0$, i.e.,

$$\text{RelChg} := \frac{\|(A^{k+1}, B^{k+1}) - (A^k, B^k)\|_F}{\|(A^k, B^k)\|_F + 1} \leq tol. \tag{3.4}$$

3.2 Exact Recoverability

In this section, we present numerical results to demonstrate the exact recoverability of the model (3.1). We set $m = n = 100$ and $tol = 10^{-6}$. For each pair (`r, spr`) we ran 50 trials and claimed successful recovery when $\text{RelErr} \leq \epsilon$ for some $\epsilon > 0$. The probability of exact recovery is defined as the successful recovery rate. The following results show the recovery of (3.1) when `r` varies from 1 to 40 and `spr` varies from 1% to 35%.

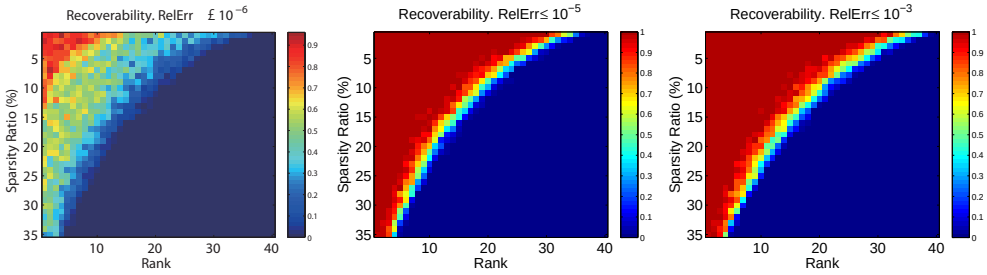

Figure 1: Recoverability results from impulsive sparse matrices. Number of trials is 50.

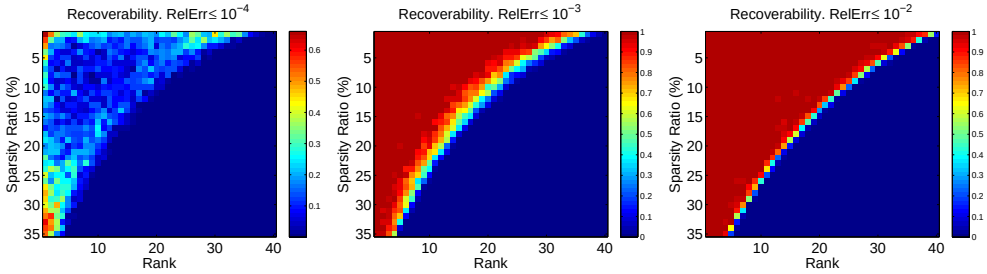

Figure 2: Recoverability results from Gaussian sparse matrices. Number of trials is 50.

It can be seen from Figures 1 and 2 that the model (3.1) shows exact recoverability when either the sparsity ratio of A^* or the rank of B^* is suitably small. More specifically, for impulsive sparse matrices, the resulting relative errors are as small as 10^{-5} for a large number of test problems. When `r` is small, say `r` ≤ 5, ADM applied to (3.1) results in faithful recoveries with `spr` as high as 35%. On the other hand, high accuracy recovery is attainable for `r` as high as 30 when `spr` $\leq 5\%$. Comparing Figures 1 and 2, it is clear that, under the same conditions of sparsity ratio and matrix rank, the probability of high accuracy recovery is smaller when A^* is a Gaussian sparse matrix. This is reasonable since in general impulsive errors are easier to be eliminated than Gaussian errors.

3.3 Recovery Results

In this section, we present two classes of recovery results for both impulsive and Gaussian sparse matrices. The first class of results are illustrative examples with $m = n = 100$, while the the second class contains results of larger problems.

Figure 3 shows the average results of 100 trials for several pairs of (`r, spr`), where x-axes represent the resulting relative errors (average those relative errors fall into $[10^{-(i+1)}, 10^{-i}]$) and y-axes represent the number of trials ADM obtains relative errors in a corresponding interval. It can be seen from Figure 3 that, for impulsive sparse matrices (left-hand side plot), the resulting relative errors are mostly quite small and low-quality recovery appears for less than 20% of the 100 random trials. In comparison, for Gaussian sparse matrices (right-hand side plot), the resulting relative errors are mostly between 10^{-3} and 10^{-4}, and it is generally rather difficult to obtain higher accuracy unless both `r` and `spr` are quite small.

Some test results on larger problems are given in Tables 1 and 2. In these tests, we set $m = n = 100, 500, 1000$ with both low-rank ratio (the rank of B^* divided by $\min(m, n)$) and sparsity ratio (the number of nonzero elements of A^* divided by mn) as large as 10%. The

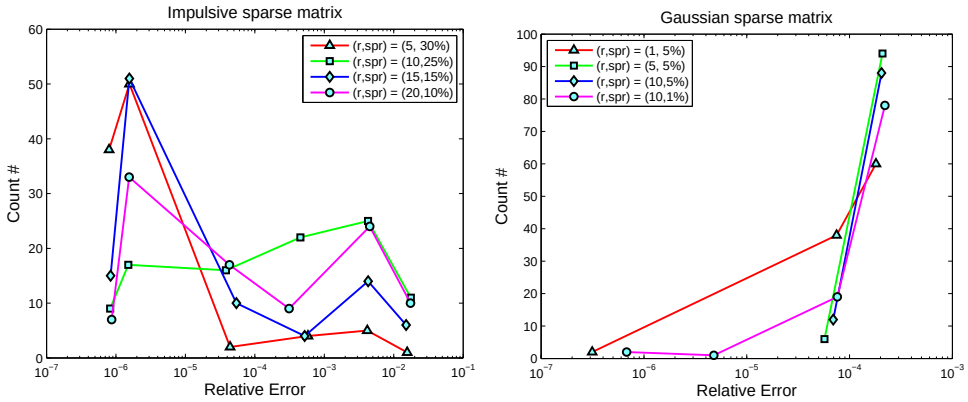

Figure 3: Recovery results of 100 trials. Left: results recovered from impulsive sparse matrices; Right: results recovered from Gaussian sparse matrices. In both plots, x-axes represent relative error of recovery (average those relative errors fall into $[10^{-(i+1)}, 10^{-i})$) and y-axes represent the number of trials ADM results to a relative error in a corresponding interval.

algorithm was terminated by relative change defined in (3.4) with $tol = 10^{-4}$. Let $(\hat{A}_t, \hat{B}_t)$ be the recovered results from (3.1) with parameter t. We define the relative error to the original sparse matrix A^* by

$$\text{ErrSP} = \|\hat{A}_t - A^*\|_F / \|A^*\|_F.$$

Similarly, the relative error to the original low-rank matrix B^* is defined by

$$\text{ErrLR} = \|\hat{B}_t - B^*\|_F / \|B^*\|_F.$$

The total relative error (RelErr) between $(\hat{A}_t, \hat{B}_t)$ and (A^*, B^*) is as defined in (3.2). The recovered results from both impulsive and Gaussian sparse matrices are, respectively, given in Tables 1 and 2, where the number of iteration used by ADM (Iter), the computing time in seconds (Time) along with the resulting relative errors (ErrSP, ErrLR and RelErr) are given.

Table 1: Recovered results: A^* is impulsive sparse.

$m = n$	t	r	spr	ErrSP	ErrLR	RelErr	Iter	Time
100	0.1	10	5%	5.8e-5	2.4e-4	1.6e-4	13	0.12
			10%	1.4e-4	3.1e-4	2.1e-4	17	0.17
500	0.05	10	5%	3.3e-5	1.9e-5	2.9e-5	10	2.7
			10%	5.0e-5	5.1e-5	5.0e-5	11	2.8
		50	5%	4.4e-5	2.7e-4	3.8e-5	13	3.9
			10%	5.2e-5	5.8e-5	5.4e-5	15	3.8
1000	0.05	50	5%	2.8e-5	1.8e-5	2.4e-5	12	23
			10%	5.5e-5	5.4e-5	5.5e-5	13	25
		100	5%	4.6e-5	3.6e-5	4.2e-5	14	27
			10%	7.7e-5	7.4e-5	7.6e-5	16	31

Table 2: Recovered results: A^* is Gaussian sparse.

$m = n$	t	r	spr	ErrSP	ErrLR	RelErr	Iter	Time
100	0.1	10	5%	3.5e-2	1.3e-3	2.9e-3	48	0.44
			10%	1.5e-2	8.3e-4	1.8e-3	82	0.77
500	0.05	10	5%	6.2e-3	9.3e-5	4.5e-4	55	14
			10%	6.4e-3	1.5e-4	6.7e-4	57	15
		50	5%	2.2e-2	3.3e-4	7.7e-4	71	18
			10%	1.7e-2	3.8e-4	8.4e-4	89	23
1000	0.05	50	5%	1.8e-2	2.0e-4	6.1e-4	69	135
			10%	1.6e-2	3.2e-4	7.7e-4	82	165
		100	5%	2.8e-2	3.0e-4	7.0e-4	84	165
			10%	2.5e-2	5.0e-4	9.3e-4	107	212

It can be seen from the results given in Tables 1 and 2 that the ADM algorithm performs very stably. More specifically, for the impulsive noise case, the iteration numbers required by ADM to attain the preset stopping tolerance is at most 17, while the resulting relative errors to the true sparse and low-rank components are of the orders $10^{-4} \sim 10^{-5}$. The consumed CPU time is about 30 seconds for m and n as large as 1000. For the Gaussian noise case, the corresponding results are generally worse than those of the impulsive noise case. More specifically, the required iteration numbers are increased to about 100, and the resulting relative errors are larger than those for the impulsive noise. These results also imply the fact that the problem (3.1) becomes harder when the sparse component is subjected to Gaussian distribution, rather than Bernoulli distribution.

3.4 Other Experimental Results

In the ALM scheme (2.1), the penalty parameter β plays an important role. Theoretically, larger β leads to faster convergence. In particular, superlinear convergence of the ALM can be achieved if $\beta \to +\infty$, even though this scenario should be avoided in practice due to that extremely large values of β lead to numerical difficulty empirically. Since the ADM framework (2.2a)-(2.2c) is a practical variant of the ALM, its practical performance is also dependent on the value of β. In this subsection, we conduct some experiments to illustrate the performance of ADM with different values of β. Specifically, we tested two types of data generated as in subsection 3.1. For illustrative purpose, we only tested the case $m = n = 500$ with ADM terminated by (3.4). The test results of ADM with different values of β are given in Table 3.

It can be seen from Table 3 that ADM performs very well with the data dependent choice $\beta_0 = 0.25mn/\|C\|_1$. Specifically, ADM obtained solutions with relative errors to the true sparse and low-rank matrices (defined in (3.2)) in the order of $10^{-5} \sim 10^{-7}$ for both types of tested data, and it usually requires more iterations when more extreme values of β are used. These results also indicate that the ADM for the sparse and low-rank matrix decomposition problem (1.1) is somehow sensitive to the choice of β. We point out that this data-dependent choice of β is by no means optimal due to our preliminary experiments. For various problems, the optimal choice of β could be different, see e.g. [49] for intensive discussions on low-rank matrix completion problems. In fact, the value of β can be determined by some self-adaptive rules, see e.g. [26, 28, 29, 31].

Next, we report some experimental results of the model (3.1) with various values of t. i.e., t= 0.03, 0.06, 0.09 and 0.12. We set $\beta = 0.25mn/\|C\|_1$ and tested the same sets of data

Table 3: Test results of different β values. RelErr is defined in (3.2).

$m = n = 500$, $t = 0.05$, $tol = 10^{-6}$, $\beta_0 = 0.25mn/\|C\|_1$									
β		$0.1\beta_0$		β_0		$10\beta_0$		$50\beta_0$	
Sparse Matrix Type: Impulsive									
r	spr	Iter	RelErr	Iter	RelErr	Iter	RelErr	Iter	RelErr
10	5%	51	4.58e-6	14	3.70e-7	24	6.32e-7	133	2.68e-7
	10%	69	6.06e-6	16	2.95e-7	25	5.18e-7	114	6.87e-7
50	5%	130	1.42e-5	20	8.61e-7	32	4.72e-7	147	4.76e-7
	10%	190	2.39e-5	26	2.16e-6	36	8.40e-7	140	1.60e-6
Sparse Matrix Type: Gaussian									
r	spr	Iter	RelErr	Iter	RelErr	Iter	RelErr	Iter	RelErr
10	5%	204	1.70e-4	80	3.97e-5	164	3.80e-6	864	6.98e-7
	10%	275	1.65e-4	82	4.46e-5	165	8.62e-6	931	2.25e-7
50	5%	371	1.57e-4	129	3.92e-5	171	7.09e-6	824	9.81e-7
	10%	465	1.87e-4	188	2.97e-5	177	7.72e-6	834	5.59e-7

with $m = n = 500$. The ADM is terminated by (3.4) with $tol = 10^{-6}$, and the tested results are given in Table 4.

Table 4: Test results of different t values. RelErr is defined in (3.2).

$m = n = 500$, $tol = 10^{-6}$, $\beta = 0.25mn/\|C\|_1$									
t		0.03		0.06		0.09		0.12	
Sparse Matrix Type: Impulsive									
r	spr	Iter	RelErr	Iter	RelErr	Iter	RelRelErr	Iter	RelErr
10	5%	36	2.33e-7	22	4.13e-7	20	9.02e-7	31	1.27e-6
	10%	37	3.68e-7	24	3.27e-7	28	1.53e-6	108	1.19e-2
50	5%	52	5.79e-7	30	6.44e-7	34	7.47e-7	134	6.62e-7
	10%	56	8.96e-7	35	1.13e-6	77	1.63e-3	197	1.20e-2
Sparse Matrix Type: Gaussian									
r	spr	Iter	RelErr	Iter	RelErr	Iter	RelErr	Iter	RelErr
10	5%	85	1.67e-5	75	4.15e-5	94	4.72e-5	188	4.80e-5
	10%	76	3.08e-5	111	3.64e-5	223	2.90e-5	231	6.63e-2
50	5%	137	2.25e-5	149	3.51e-5	195	4.89e-5	452	5.04e-4
	10%	151	2.74e-5	228	3.01e-5	371	4.34e-3	503	1.21e-2

It can be seen from these results that the model (3.1) ensures faithful recovery for a wide range values of t. Specifically, for the tested cases where the matrices to be decomposed have highly sparse and low rank components, high-quality reconstruction can be ensured for t around $0.03 \sim 0.06$. We also observed that ADM works quite well for the tested values of t, and ADM usually takes more iterations to achieve the prescribed accuracy when t becomes larger or the sparse component becomes less sparse. Via numerical experiments, we also find that the model (3.1) becomes not suitable for high-quality recovery when t is beyond the interval $[0.03, 0.12]$.

Clearly, the main cost at each iteration of the ADM framework (2.2a)-(2.2c) for solving (1.1) lies in the computation of B in (2.4). In our implementation, we simply computed the

full SVD of the matrix $C - A^{k+1} + Z^k/\beta$ at each iteration. However, by taking a closer examination on the formula (2.4), we found that only those singular values of the matrix $C - A^{k+1} + Z^k/\beta$ bigger than $1/\beta$ and the corresponding singular vectors are utilized in the computation of B^{k+1}. Therefore, it is helpful if this property can be utilized in the algorithmic design. Next we examine the variation of the rank of B with respect to the iterative procedure of ADM. We set $m = n = 1000$ and tested impulsive sparse matrices data. The same as before, we set $\beta = 0.25mn/\|C\|_1$ and terminated ADM by (3.4) with $tol = 10^{-6}$. The test results are plotted in Figure 4.

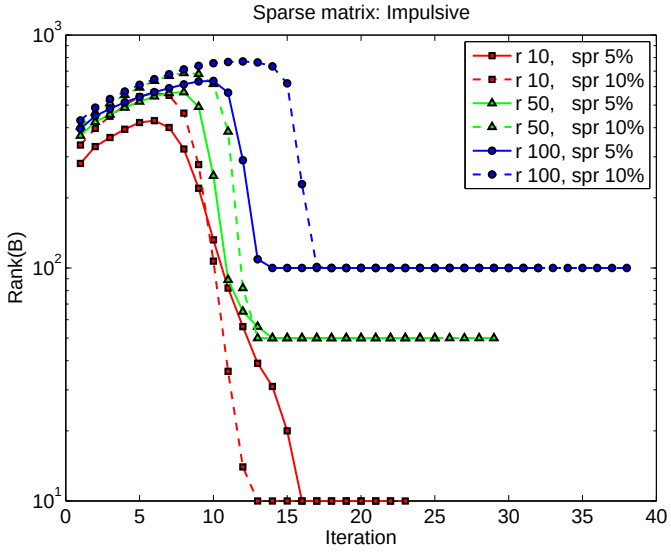

Figure 4: Changing behavior of the rank of B.

It can be seen form Figure 4 that, for impulsive sparse matrix data, the ADM is able to recover the true rank of B^* within a few iterations (about 16 iterations for the tested cases). However, the results for Gaussian sparse matrices data (which are not plotted here due to reasons given below) are generally different. For Gaussian sparse matrices data, the solutions of model (3.1) do not have identical ranks to the true low-rank matrices, even though they have small relative errors to the true low-rank matrices (RelErr is in the order of $10^{-5} \sim 10^{-6}$). This is imaginable: small Gaussian errors are more difficult to be eliminated than large impulsive noise, and furthermore, the rank of a matrix is very sensitive to small perturbations of its entries. In practical applications, some post processing techniques to numerical solutions of (3.1) are highly recommended if the sparse component have many entries corrupted by Gaussian noise. It is also preferable to develop such an algorithm that is capable of estimating the true ranks of the low-rank matrices approximately based on the information accumulated during iterations.

4 Conclusions

This paper mainly emphasizes the applicability of the alternating direction method (ADM) for solving the sparse and low-rank matrix decomposition (SLRMD) problem, and thus numerically reinforces the work [3, 6]. It has been shown that the existing ADM type methods can be extended to solve the SLRMD problem, and the implementation is pretty

easy since all the resulting subproblems have closed-form solutions. Preliminary numerical results exhibit affirmatively the efficiency of ADM for the SLRMD problem.

Acknowledgement

We are grateful to two anonymous referees for their valuable comments and suggestions which have helped us improve the presentation of this paper. We also thank Professor Kim-Chuan Toh of National University of Singapore for providing us an efficient MATLAB Mex interface for computing SVD via a divide-and-conquer routine (`dgesdd`) implemented in LAPACK.

References

[1] D.P. Bertsekas and J.N. Tsitsiklis, *Parallel and Distributed Computation: Numerical Methods*, Prentice-Hall, Englewood Cliffs, NJ, 1989.

[2] J.F. Cai, E.J. Candès and Z.W. Shen, A singular value thresholding algorithm for matrix completion, *SIAM J. Optim.* 20 (2010) 1956–1982.

[3] E.J. Candès, X.D. Li, Y. Ma and J. Wright, Robust principal component analysis?, *ACM* 58 (2011) 11.

[4] E.J. Candès, J. Romberg and T. Tao, Robust uncertainty principles: exact signal reconstruction from highly incomplete frequency information, *IEEE Trans. Inform. Theory* 52 (2006) 489–509.

[5] E.J. Candès and B. Recht, Exact matrix completion via convex optimization, *Found. Comput. Math.* 9 (2008) 717–772.

[6] V. Chandrasekaran, S. Sanghavi, P.A. Parrilo and A.S. Willskyc, Rank-sparsity incoherence for matrix decomposition, *SIAM J. Optim.* 21 (2011) 572–596.

[7] C.H. Chen, B.S. He and X.M. Yuan, Matrix completion via alternating direction method, *IMA J. Numer. Anal.* 32 (2012) 227–245.

[8] G. Chen and M. Teboulle, A proximal-based decomposition method for convex minimization problems, *Math. Program.* 64 (1994) 81–101.

[9] S. Chen, D. Donoho, and M. Saunders, Atomic decomposition by basis pursuit, *SIAM J. Sci. Comput.* 20 (1998) 33–61.

[10] D.L. Donoho and X. Huo, Uncertainty principles and ideal atomic decomposition, *IEEE Trans. Inform. Theory* 47 (2001) 2845–2862.

[11] D.L. Donoho and M. Elad, Optimal sparse representation in general (nonorthogonal) dictionaries via ℓ_1 minimization, *Proc. Natl. Acad. Sci.* 100 (2003) 2197–2202.

[12] D.L. Donoho, Compressed sensing, *IEEE Trans. Inform. Theory* 52 (2006) 1289–1306.

[13] J. Eckstein and M. Fukushima, Some reformulation and applications of the alternating directions method of multipliers, in *Large Scale Optimization: State of the Art*, W.W. Hager (eds), Kluwer Academic Publishers, 1994, pp. 115–134.

[14] E. Esser, Applications of Lagrangian-based alternating direction methods and connections to split Bregman, Manuscript, http://www.math.ucla.edu/applied/cam/.

[15] M. Fazel, *Matrix rank minimization with applications*, *PhD thesis*, Stanford University, 2002.

[16] M. Fazel and J. Goodman, Approximations for partially coherent optical imaging systems, *Technical Report,* Stanford University, 1998.

[17] M. Fazel, H. Hindi and S. Boyd, A rank minimization heuristic with application to minimum order system approximation, *Proceedings American Control Conference* 6 (2001) 4734–4739.

[18] M. Fazel, H. Hindi and S. Boyd, Log-det heuristic for matrix rank minimization with applications to Hankel and Euclidean distance matrices, *Proceedings of the American Control Conference*, 2003.

[19] M. Fukushima, Application of the alternating direction method of multipliers to separable convex programming problems, *Comput. Optim. Appl.* 1 (1992) 93–111.

[20] D. Gabay, Application of the method of multipliers to varuational inequalities, in *Augmented Lagrangian methods: Application to the numerical solution of Boundary-Value Problem*, M. Fortin and R. Glowinski (eds.), North-Holland, Amsterdam, The Netherlands, 1983, pp. 299–331.

[21] D. Gabay and B. Mercier, A dual algorithm for the solution of nonlinear variational problems via finite element approximations, *Comput.Math. Appl.* 2 (1976) 17–40.

[22] R. Glowinski, *Numerical Methods for Nonlinear Variational Problems*, Springer-Verlag, New York, Berlin, Heidelberg, Tokyo, 1984.

[23] R. Glowinski and P. Le Tallec, *Augmented Lagrangian and Operator Splitting Methods in Nonlinear Mechanics*, SIAM Studies in Applied Mathematics, Philadelphia, PA, 1989.

[24] R. Glowinski and A. Marrocco, Approximation par éléments finis d'ordre un et résolution par pénalisation-dualité d'une classe de problèmes non linéaires, *R.A.I.R.O.* R2 (1975) 41-76.

[25] G.H. Golub and C. F. van Loan, *Matrix Computation*, third edition, The Johns Hopkins University Press, 1996.

[26] B. S. He, L. Z. Liao, D. Han and H. Yang, A new inexact alternating directions method for monontone variational inequalities, *Math. Program.* 92 (2002) 103–118.

[27] B.S. He, M.H. Xu and X.M. Yuan, Solving large-scale least squares covariance matrix problems by alternating direction methods, *SIAM J. Matrix Anal. Appl.* 32 (2011) 136–152.

[28] B.S. He and H. Yang, Some convergence properties of a method of multipliers for linearly constrained monotone variational inequalities, *Oper. Res. Lett.* 23 (1998) 151–161.

[29] B.S. He and X.M. Yuan, The unified framework of some proximal-based decomposition methods for monotone variational inequalities with separable structure, *Pac. J. Optim.* 8 (2012) 817–844.

[30] M.R. Hestenes, Multiplier and gradient methods, *J. Optim. Theory Appl.* 4 (1969) 303–320.

[31] S. Kontogiorgis and R. R. Meyer, A variable-penalty alternating directions method for convex optimization, *Math. Program.* 83 (1998) 29–53.

[32] R.M. Larsen, *PROPACK-Software for large and sparse SVD calculations*, Available from http://sun.stanford.edu/srmunk/PROPACK/.

[33] S.L. Lauritzen, *Graphical Models*, Oxford University Press, Oxford, 1996.

[34] S.Q. Ma, D. Goldfarb and L.F. Chen, Fixed point and Bregman iterative methods for matrix rank minimization, *Math. Program.* 128 (2011) 321–353.

[35] M.K. Ng, P.A. Weiss and X.M. Yuan, Solving constrained total-variation problems via alternating direction methods, *SIAM J. Sci. Comput.* 32 (2010) 2710–2736.

[36] Y.C. Pati and T. Kailath, Phase-shifting masks for microlithography: Automated design and mask requirements, *J. Opt. Soc. Amer.* 11 (1994) 2438–2452.

[37] M.J.D. Powell, A method for nonlinear constraints in minimization problems, in *Optimization*, R. Fletcher (eds.), Academic Press, New York, 1969, pp. 283–298.

[38] B. Recht, M. Fazel and P. A. Parrilo, Guaranteed minimum rank solutions to linear matrix equations via nuclear norm minimization, *SIAM Rev.* 52 (2010) 471–501.

[39] E. Sontag, *Mathematical Control Theory*, Springer-Verlag, New York, 1998.

[40] J. Sun and S. Zhang, A modified alternating direction method for convex quadratically constrained quadratic semidefinite programs, *European J. Oper. Res.* 207 (2010) 1210–1220.

[41] R. Tibshirani, Regression shrinkage and selection via the LASSO, *J. Roy. Statist. Soc. Series B,* 58 (1996) 267–288.

[42] P. Tseng, Alternating projection-proximal methods for convex programming and variational inequalities, *SIAM J. Optim.* 7 (1997) 951–965.

[43] R.H. Tütüncü, K.C. Toh and M.J. Todd, Solving semidefinite-quadrtic-linear programs using SDPT3, *Math. Program.* 95 (2003) 189–217.

[44] L.G. Valiant, Graph-theoretic arguments in low-level complexity, *6th Symposium on Mathematical Foundations of Computer Science,* 1977, pp. 162-176.

[45] C.H. Ye and X.M. Yuan, A descent method for stuctured monotone variational inequalities, *Optim. Methods Softw.* 22 (2007) 329–338.

[46] Z.W. Wen, D. Goldfarb and W. Yin, Alternating direciotn augmented lagrangina methods for semidefinite programming, *Math. Program. Comput.* 2 (2010) 203–230.

[47] J.F. Yang and Y. Zhang, Alternating direction algorithms for ℓ_1 problems in compressive sensing, *SIAM J. Sci. Comput.* 33 (2011) 250–278.

[48] X.M. Yuan, Alternating direction methods for sparse covariance selection, *J. Sci. Comput.* 51 (2012) 261–273.

[49] S. Zhang, J. Ang and J. Sun, An alternating direction method for solving convex nonlinear semidefinite programming problems, *Optim.*, to appear.

Manuscript received 2 February 2011
revised 3 March 2011
accepted for publication 7 March 2011

Xiaoming Yuan
Department of Mathematics, Hong Kong Baptist University, Hong Kong, China
E-mail address: xmyuan@hkbu.edu.hk

Junfeng Yang
Department of Mathematics, Nanjing University, Nanjing, Jiangsu, China
E-mail address: jfyang@nju.edu.cn