LOW-RANK APPROXIMATIONS WITH SPARSE FACTORS II: PENALIZED METHODS WITH DISCRETE NEWTON-LIKE ITERATIONS

ZHENYUE ZHANG*, HONGYUAN ZHA† AND HORST SIMON‡

Abstract. In [7], we developed numerical algorithms for computing *sparse* low-rank approximations of matrices and we also provided a detailed error analysis of the proposed algorithms together with some numerical experiments. The low-rank approximations are constructed in a certain factored form with the degree of sparsity of the factors controlled by some user-specified parameters. In this paper, we cast the sparse low-rank approximation problem in the framework of penalized optimization problems. We discuss various approximation schemes for the penalized optimization problem which are more amendable for numerical computations. We also include some analysis to show the relations between the original optimization problem and the reduced one. We then develop a globally convergent *discrete* Newton-like iterative method for solving the approximate penalized optimization problems. We also compare the reconstruction errors of the sparse low-rank approximations computed by our new methods with those obtained using the methods in [7] and several other existing methods for computing sparse low-rank approximations. Numerical examples show that the penalized methods are more robust and produce approximations with lower approximate ranks (less column numbers) and/or sparser factors.

1. Introduction.

Low-rank approximations of matrices have many applications in information retrieval, data mining and solving ill-posed problems, to name a few [5, 7]. The theory of singular value decomposition (SVD) provides the best rank-k approximation $\text{best}_k(A)$ of a given $m \times n$ matrix A in terms of its singular values and singular vectors,

$$\text{best}_k(A) \equiv U_k \Sigma_k V_k^T = [u_1, \dots, u_k] \operatorname{diag}(\sigma_1, \dots, \sigma_k)[v_1, \dots, v_k]^T,$$

where $\sigma_i, i = 1, \dots, k$, are the k largest singular values of A, and u_i and v_i are the corresponding left and right singular vectors [1]. Notice that even when A is sparse, there is in general no guarantee that $\text{best}_k(A)$ will be sparse, not even the factors U_k and V_k. To remedy this drawback of the low-rank approximations computed by SVD, it was proposed to construct low-rank approximations B_k of A in a factored form $B_k = X_k D_k Y_k^T$ without imposing the orthogonality constraints on columns of the left and right factors X_k and Y_k [2, 4, 6]. In [7] we further developed this idea and proposed to compute $B_k = X_k D_k Y_k^T$ with sparse factors X_k and Y_k that solves the following optimization problem,

$$(1.1)\quad \min\{\|A - X_k D_k Y_k^T\|_F \mid D_k \text{ diagonal}, X_k \in \mathcal{R}^{m\times k} \text{ and } Y_k \in \mathcal{R}^{n\times k} \text{ sparse }\}.$$

* Department of Mathematics, Zhejiang University, Yu-Quan Campus, Hangzhou, 310027, P. R. China. zyzhang@zju.edu.cn. The work of this author was supported in part by NSFC (project 19771073), the Special Funds for Major State Basic Research Projects of China (project G19990328), and Foundation for University Key Teacher by the Ministry of Education, China. The work also was supported in part by NSF grants CCR-9619452 and by the Director, Office of Science, Office of Laboratory Policy and Infrastructure Management, of the U.S. Department of Energy under Contract No. DE-AC03-76SF00098.

† Department of Computer Science and Engineering, The Pennsylvania State University, University Park, PA 16802, zha@cse.psu.edu. The work of this author was supported in part by NSF grants CCR-9901986.

‡ National Energy Research Scientific Computing Center, Lawrence Berkeley National Laboratory, One Cyclotron Road, M/S: 50B, Berkeley, CA 94720, HDSimon@lbl.gov. This work was supported by the Director, Office of Science, Office of Laboratory Policy and Infrastructure Management, of the U.S. Department of Energy under Contract No. DE-AC03-76SF00098.

In [7], several algorithms are developed for controlling the degree of sparsity of the factors X_k and Y_k, and a detailed error analysis of our proposed algorithms is given that compares the computed sparse low-rank approximations with those obtained from SVD and some of the previous methods developed in [2, 6]. The basic idea is to use a sequence of rank-one deflation steps to construct the approximation

$$B_k = X_k D_k Y_k^T = [x_1, \dots, x_k]\,\mathrm{diag}(d_1, \dots, d_k)[y_1, \dots, y_k]^T = \sum_{i=1}^{k} x_i d_i y_i^T.$$

At each deflation step, approximations to the largest left and right singular vectors $\hat{u}_i$ and $\hat{v}_i$ of the deflated matrix $A_{i-1} = A - B_{i-1}$ are used to construct a sparse rank-one approximation $x_i d_i y_i^T$ for matrix A_{i-1}, here $A_0 = A$. Specifically, the sparse vectors x_i and y_i are obtained by discarding small components of $\hat{u}_i$ and $\hat{v}_i$. We proved that if the norm of the vector consisting of the discarded components is no greater than $\sqrt{2}\epsilon$ at each step, then the computed sparse low-rank approximation B_k has *reconstruction error*, defined as $\|A - B_k\|_F$, no greater than that of the best rank-k approximation $\mathrm{best}_k(A)$ by a factor $(1 + b_k\epsilon)^{1/2}$, i.e.,

$$\|A - B_k\|_F \le (1 + b_k\epsilon)^{1/2} \|A - \mathrm{best}_k(A)\|_F, \tag{1.2}$$

where $b_1 = \sigma_1^2 / \left(\sigma_2^2 + \cdots \sigma_n^2\right)$ (assuming $n \le m$) and

$$b_k = \frac{\sum_{i=1}^{k} \sigma_i \sigma_{i+1}}{\sum_{i=k+1}^{n} \sigma_i^2} + O(\epsilon), \quad k \ge 2.$$

The tolerance parameter ϵ specified by the user can balance the tradeoff between the degree of sparsity and good reconstruction error of the low-rank approximations. We suggested in [7] that the size of the tolerance ϵ_i used at each deflation step be a variable determined by

$$\epsilon_i = \frac{\|A_{i-1}\|_F}{\|A\|_F}\epsilon, \quad i = 1, \dots, k. \tag{1.3}$$

Numerical results in [7] show that the variable-tolerance scheme (1.3) works better than an alternative constant-tolerance scheme whereby the tolerance is fixed at each step. In general, if we pre-set the desirable reconstruction error, reducing ϵ in (1.3) will yield an approximation with smaller rank k but the degree of sparsity of X_k and Y_k tend to increase while increasing ϵ will have the opposite effect. Moreover, we also observed that the ranks and the degree of sparsity of the low-rank approximations computed by the methods in [7] sometimes can be quite sensitive to the choice of ϵ (ϵ_j), i.e., a slight change of ϵ having little effect on the reconstruction error, can have a much greater effect on the rank of the low-rank approximation and the degree of sparsity of its factors This less robust behavior is rather undesirable.

Our goal is to develop more robust methods for sparse low-rank approximations. Our basic idea for the improved algorithms is to use penalty terms to penalize low-rank approximations with factors X_k and Y_k having large number of nonzeros. In a rather general framework, we can consider the following optimization problem,

$$\min\{\lambda_x \mathtt{nnz}(X_k) + \lambda_y \mathtt{nnz}(Y_k) + \lambda_e \|A - X_k D_k Y_k^T\|_F^2 \mid D_k \text{ diagonal}\}, \tag{1.4}$$

where λ_x, λ_y, and λ_e are user determined penalty parameters, here $\mathtt{nnz}(\cdot)$ denotes the number of nonzeros of a matrix. In essence, we want a low-rank approximation

$B_k = X_k D_k Y_k^T$ to have a small reconstruction error $\|A - X_k D_k Y_k^T\|_F$ and at same time we also penalize those B_k the X_k and Y_k factors of which have large number of nonzeros. In particular, we can use a variation of the deflation technique to reduce the problem (1.4) to the problem of finding a sequence of sparse rank-one approximations, and build the low-rank approximation one rank at a time [2, 4, 6, 7]. Therefore, in this paper we focus on the rank-one case of problem (1.4)

$$\min_{x,d,y}\{\lambda_x \mathbf{nnz}(x) + \lambda_y \mathbf{nnz}(y) + \lambda_e \|A - xdy^T\|_F^2\}. \tag{1.5}$$

This is an optimization problem with an objective function over $(m+n+1)$-dimensional space, assuming the matrix A has m rows and n columns. We can certainly use some general techniques to solve the optimization problem (1.5). Because of the large dimensionality and the un-smooth property of the objective function, general techniques for solving optimization problems will be not efficient and cost much for the optimization problem (1.5). However, the problem itself possesses many useful structures that deserves exploitation. Our approach that will be proposed in this paper is to approximately reduce the $(m+n+1)$-dimensional optimization problem (1.5) to a much simpler one-dimensional discrete problem by several reduction steps each of which involves a certain tight approximation. Indeed, some structures of the 1-D discrete optimization problem can be further exploited by a continuous-discrete relaxation technique (smoothly interpolating and discretely retransforming) so that it can be solved easily by using a discrete Newton-like iteration. We will discuss the continuous-discrete relaxation technique and derive the discrete Newton-like iteration in this paper.

The rest of the paper is organized as follows. In section 2, we first examine the rank-one case of the penalized optimization problem (1.5) and then, following the idea of constructing sparse rank-one approximation proposed in [7], we discuss the 1-D discrete optimization problem that will be solved eventually. To show the relations between the 1-D problem actually solved and the original problem (1.5), we describe in section 3 the reduction process in several steps, giving the motivation behind each steps and showing of the degree of approximation involved with respect to (1.5) by theoretical analysis together with numerical experiments. The continuous-discrete relaxation technique mentioned above is discussed in section 4. In section 5, we propose a *discrete* globally convergent Newton-like method for solving the resulting discrete optimization problem. Several computational issues arise in the proposed method: We focus on how to compute the Newton-like directions at each iterative step and how to select the next iterate to guarantee the existence of a so-called bracketing interval. In section 6, we present some numerical examples, and make comparison with several existing methods for computing sparse low-rank approximations.

2. Penalized optimization problems for sparse rank-one approximations. In this section, we consider the rank-one version (1.5) of the optimization problem (1.4). To make the penalty parameters clearer, we rewrite (1.5) in the following format,

$$\min_{x,y,d}\left\{\lambda\Big(\mu\frac{\mathbf{nnz}(x)}{m} + (1-\mu)\frac{\mathbf{nnz}(y)}{n}\Big) + (1-\lambda)\frac{\|A - xdy^T\|_F^2}{\|A\|_2^2}\right\}, \tag{2.6}$$

where the parameters λ and μ are two user-specified penalty terms chosen in the interval $(0, 1)$, and have the following interpretations: λ balances the degree of sparsity and the reconstruction error of the rank-one approximation while μ balances the degrees of sparsity of the left and right factors of the rank-one approximation xdy^T. In general, we choose $\mu = 1/2$ to keep the sparsity structure of the approximation *symmetric* if no other reasons dictate us to do otherwise.

It is easy to verify that for fixed vectors x and y,

$$\min_d \|A - xdy^T\|_F^2 = \|A\|_F^2 - d^2(x,y),$$

where

$$d(x,y) = \frac{x^T Ay}{\|x\| \, \|y\|}. \tag{2.7}$$

Therefore, ignoring a constant factor, (2.6) can be written in the form

$$\min_{x,y} \Big\{ \lambda\Big(\mu\frac{\mathtt{nnz}(x)}{m} + (1-\mu)\frac{\mathtt{nnz}(y)}{n}\Big) - (1-\lambda)\frac{d^2(x,y)}{\|A\|_2^2} \Big\}. \tag{2.8}$$

For simplicity, let us introduce

$$\alpha = \frac{\lambda\mu}{(1-\lambda)m}, \quad \beta = \frac{\lambda(1-\mu)}{(1-\lambda)n}, \tag{2.9}$$

and rewrite the optimization problem (2.8) in the following equivalent form

$$G(\alpha,\beta) = \min_{x,\,y} \left\{ \alpha * \mathtt{nnz}(x) + \beta * \mathtt{nnz}(y) - \frac{d^2(x,y)}{\|A\|_2^2} \right\}. \tag{2.10}$$

The objective function of the optimization problem above is a combination of an integer-valued function and a continuous function over a $(m+n)$-dimensional space of continuous variables. Following the idea of constructing sparse rank-one approximation proposed in [7], we consider an approximate solution with x and y constructed from the components of the left and right singular vectors u and v corresponding to the largest singular value of A. To make the presentation self-contained as well as for later reference, we briefly review the basic ideas discussed in [7] for sparse rank-one approximations.

Assume that the largest singular vectors u and v of A are such that $\|u\|_2 = \|v\|_2 = 1$. Let $\tilde{w}$ be the sorted vectors of the vector $w = [u^T, v^T]^T$ in nonincreasing order,

$$|\tilde{w}(1)| \geq |\tilde{w}(2)| \geq \ldots \geq |\tilde{w}(m+n)|.$$

For a given $\xi > 0$, let $k = k(\xi) \leq m+n$ be the smallest integer satisfying

$$\tilde{w}^2(1) + \tilde{w}^2(2) + \ldots + \tilde{w}^2(k) \geq 2(1-\xi),$$

and also let $i = i(\xi)$ and $j = j(\xi) = k(\xi) - i(\xi)$ be, respectively, the numbers of the u-components and the v-components in the subset $\{\tilde{w}(1), \ldots, \tilde{w}(k)\}$. Then denote by x_u the sparse vector consisting of the i u-components in the same component positions and zeros elsewhere. Similarly, denote by y_v the sparse vector consisting of the j v-components in the same component positions and zeros elsewhere. Clearly, if P_u and P_v are the permutations determined by sorting w to $\tilde{w}$ such that $\tilde{u} = P_u u$ and $\tilde{v} = P_v v$ are in nondecreasing orders in absolute value, then

$$x_u = P_u^T \begin{bmatrix} \tilde{u}(1:i) \\ 0 \end{bmatrix}, \quad y_v = P_v^T \begin{bmatrix} \tilde{v}(1:j) \\ 0 \end{bmatrix}. \tag{2.11}$$

In [7], we prove that if $\xi \le 1/3$, then

$$\|A - x_u d(x_u, y_v) y_v^T\|_F^2 \le (1 + b_1\tau)\|A - u\sigma v^T\|_F^2, \tag{2.12}$$

where $b_1 = \sigma_1^2/\left(\sigma_2^2 + \cdots \sigma_n^2\right)$ and $\tau = 4\xi\left(1-\xi^2/(1-\xi)^2\right) < 4\xi$. Note that $\|A - u\sigma v^T\|_F$ is the smallest reconstruction error for any rank-one (sparse or dense) approximation of A. In [7], the degree of sparsity of the rank-one approximation $xd(x,y)y^T$, or the integer k is controlled by the parameter ξ which is closely related to the reconstruction error since

$$\|x_u - u\|_2^2 + \|y_v - v\|_2^2 = \tilde{w}^2(k+1) + \ldots + \tilde{w}^2(m+n) \le 2\xi.$$

In general, if ξ is relatively small, we can not expect to generate rank-one approximations with left and right vectors having many zero entries. On the other hand, if ξ is relatively large, the vectors x_u and y_v can be chosen to be very sparse but $x_u d(x_u, y_v) y_v^T$ may not be a good rank-one approximation, compared with the best rank-one approximation $u\sigma v^T$.

It turns out that the parameter ξ can be approximately determined by the optimal solution to the following penalized optimization problem

$$F(\alpha, \beta, p) = \min_{\xi} \Big\{\alpha * i(\xi) + \beta * j(\xi) - \Big(\frac{|h(\xi)|}{\|A\|_2}\Big)^p\Big\}, \tag{2.13}$$

where $h(\xi) = d(x_u, y_v)$ for the sparsified vectors x_u and y_v that depend on the parameter ξ with $i(\xi)$ and $j(\xi)$ defined as above.

Comparing with the $(n+m)$-dimensional problem (2.10), the one-dimensional optimization problem (2.13) is much simpler. It can be derived from the original optimal problem (2.10) using several reduction steps. In the next section, we will describe the reduction process. To solve the optimization problem (2.13), we will present a discrete Newton-like method in section 5.

3. Relations between $G(\alpha, \beta)$ and $F(\alpha, \beta, p)$. We write (2.10) in a general form

$$\min_{x,y} \left\{\alpha \mathbf{nnz}(x) + \beta \mathbf{nnz}(y) - \left(\frac{|d(x,y)|}{\|A\|_2}\right)^p\right\} \tag{3.14}$$

for $p > 0$. Clearly, the space $\mathcal{R}^m$ of m-dimensional vectors can be divided in to m sets $\mathcal{U}_1, \mathcal{U}_2, \ldots, \mathcal{U}_m$ such that each vector in $\mathcal{U}_i$ has i nonzeros. Similar partition also holds for the space $\mathcal{R}^n$ of n-dimensional vectors. In the case when such a set $\mathcal{U}_i$ of m-dimensional vectors and a set $\mathcal{V}_j$ of n-dimensional vectors are fixed, the objective function of the optimization problem above for $x \in \mathcal{U}_i$ and $y \in \mathcal{V}_j$ is minimized by maximizing $|d(x,y)|$ over the two sets $\mathcal{U}_i$ and $\mathcal{V}_j$. This property explicitly exhibits a *combinatorial* nature of the above optimization problem. To illustrate it in a clearer fashion, for $x \in \mathcal{U}_i$ and $y \in \mathcal{V}_j$, denote by I and J the index sets defined by the corresponding nonzero components of vectors x and y, respectively. Then the size of I is $|I| = i$ and the nonzero section of x can be represented by $x(I)$, similarly with $y(J)$ for y and $|J| = j$. We also use the notation $A(I,J)$ to denote the submatrix of A the elements of which are at the intersection of the rows and columns in I and J. Therefore

$$|d(x,y)| = \left|\frac{x^T(I)A(I,J)y(J)}{\|x(I)\|_2\|y(J)\|_2}\right| \le \|A(I,J)\|_2,$$

and the minimum of the objective function in (3.14) over $\mathcal{U}_i \times \mathcal{V}_j$ is given by $\alpha * i + \beta * j - (s_{i,j}/\|A\|_2)^p$ with

$$s_{ij} = \max_{|I|=i,\,|J|=j} \|A(I,J)\|_2. \tag{3.15}$$

It follows that the optimization problem (3.14) can be written equivalently in the following form

$$G_S(\alpha,\beta,p) = \min_{i,j} \left\{ \alpha * i + \beta * j - \left(\frac{s_{ij}}{\|A\|_2} \right)^p \right\}, \tag{3.16}$$

where $S = (s_{ij}) \in \mathcal{R}^{m\times n}$ and consequently, $G(\alpha,\beta) = G_S(\alpha,\beta,2)$.

Compared with (2.10), (3.16) is only an optimization problem over a finite 2-D index set. However, the corresponding objective function of (3.16) is more difficult to evaluate when the size of matrix A is large. In the next subsections, we will derive a sequence of nonnegative matrices $H = (h_{i,j})$ that are easier to evaluate than S and additionally satisfy $S \geq H \geq 0$, i.e.,

$$s_{i,j} \geq h_{i,j} \geq 0, \quad \text{for all } i \text{ and } j.$$

This will lead to a sequence of upper bounds for $G_S(\alpha,\beta,p)$ defined by the optimization problem,

$$G_H(\alpha,\beta,p) = \min_{i\,j} \left\{ \alpha * i + \beta * j - \left(\frac{h_{ij}}{\|A\|_2} \right)^p \right\}. \tag{3.17}$$

This sequence of upper-bounding will lead to the problem (2.13) which we will eventually solve approximately.

3.1. Approximation using truncated Rayleigh quotients. As denoted in the previous section, let $\{u,\sigma,v\}$ be the largest singular triplet of matrix A. A lower bound of S is the matrix $R = (r_{ij})$ of Rayleigh quotients

$$r_{ij} = \max_{|I|=i,\,|J|=j} \left| \frac{u(I)^T A(I,J) v(J)}{\|u(I)\|\,\|v(J)\|} \right|$$

which gives rise to the following optimization problem

$$G_R(\alpha,\beta,p) = \min_{i\,j} \left\{ \alpha * i + \beta * j - \left(\frac{r_{ij}}{\|A\|_2} \right)^p \right\}.$$

This problem can be further reduced by choosing the index sets of I and J in a specific way determined by the *truncated* Rayleigh quotients as follows. Let $\tilde{u}$ and $\tilde{v}$ be the sorted versions of u and v, respectively, such that

$$|\tilde{u}(1)| \geq \cdots \geq |\tilde{u}(m)|, \quad \text{and} \quad |\tilde{v}(1)| \geq \cdots \geq |\tilde{v}(n)|,$$

and let π_u and π_v be the permutation index vectors satisfying $\tilde{u} = u(\pi_u)$ and $\tilde{v} = v(\pi_v)$. With I and J defined by

$$I = \pi_u(1:i),\; J = \pi_v(1:j), \tag{3.18}$$

we obtain a lower bound of r_{ij} which we denote as

$$t_{ij} = \left| \frac{u(I)^T A(I,J) v(J)}{\|u(I)\|\,\|v(J)\|} \right|.$$

Setting $T = (t_{t_i j})$ yields the following optimization problem

$$G_T(\alpha, \beta, p) = \min_{ij} \left\{ \alpha * i + \beta * j - \left(\frac{t_{ij}}{\|A\|_2} \right)^p \right\}. \tag{3.19}$$

Obviously, $s_{ij} \geq r_{ij} \geq t_{ij}$ and

$$G_S(\alpha, \beta, p) \leq G_R(\alpha, \beta, p) \leq G_T(\alpha, \beta, p).$$

Now back to $F(\alpha, \beta, p)$ defined in (2.13). It is not difficult to verify that the vectors x_u and y_v defined in (2.11) satisfy

$$x_u(\pi_u) = \begin{bmatrix} \tilde{u}(1:i) \\ 0 \end{bmatrix} = \begin{bmatrix} u(I) \\ 0 \end{bmatrix}, \quad y_v(\pi_v) = \begin{bmatrix} \tilde{v}(1:j) \\ 0 \end{bmatrix} = \begin{bmatrix} v(J) \\ 0 \end{bmatrix},$$

where I and J are those defined in (3.18). By setting $i = i(\xi)$ and $j = j(\xi)$, we have $|d(x_u, y_v)| = t_{ij}$. Therefore, $F(\alpha, \beta, p)$ is the minimal value of the objective function defined in (3.19) over the subset $\{\, (i(\xi),\, j(\xi)) \,|\, \xi > 0 \,\}$.

In the same way, $F(\alpha, \beta, p)$ can be reviewed as a reduced form of the original problem $G_S(\alpha, \beta, p)$ by several reduction steps and satisfy

$$G_S(\alpha, \beta, p) \leq G_R(\alpha, \beta, p) \leq G_T(\alpha, \beta, p) \leq F(\alpha, \beta, p).$$

The difference between the optimal solutions corresponding to $F(\alpha, \beta, p)$ and $G_S(\alpha, \beta, p)$ will naturally be nonzero. In the next subsection, we will illustrate the quality of $G_R(\alpha, \beta, p)$ and $G_T(\alpha, \beta, p)$ as approximations to $G_S(\alpha, \beta, p)$, and give some numerical results to compare the sparsities of the low-rank approximations obtained based on $G_T(\alpha, \beta, p)$ and $F(\alpha, \beta, p)$. In section 4, we will concentrate on computing $F(\alpha, \beta, p)$ to find approximations to $G_S(\alpha, \beta, p)$.

3.2. How good are the approximations. In the above subsection, we showed that G_R and G_T can be used to provide upper bounds of G_S. The following theorem shows they can also be used to give lower bounds of G_S.

THEOREM 3.1. *Let c_{RS}, c_{TR}, and c_{TS} be*

$$c_{RS} = \max_{ij} \frac{s_{ij}^p - r_{ij}^p}{s_{ij}^p}, \quad c_{TR} = \max_{ij} \frac{r_{ij}^p - t_{ij}^p}{r_{ij}^p}, \quad c_{TS} = \max_{ij} \frac{s_{ij}^p - t_{ij}^p}{s_{ij}^p}.$$

Then we have $0 \leq G_R - G_S \leq c_{RS}, \quad 0 \leq G_T - G_R \leq c_{TR}, \quad 0 \leq G_T - G_S \leq c_{TS}$.

Proof. Denote by (i_S, j_S) the optimal index-pair for the problem (3.16). Then

$$\begin{aligned}
0 &\leq G_R(\alpha, \beta, p) - G_S(\alpha, \beta, p) \\
&\leq \left(\alpha * i_S + \beta * i_S - \left(\frac{r_{i_S, j_S}}{\|A\|_2} \right)^p \right) - \left(\alpha * i_S + \beta * i_S - \left(\frac{s_{i_S, j_S}}{\|A\|_2} \right)^p \right) \\
&= \left(\frac{s_{i_S, j_S}}{\|A\|_2} \right)^p \left(1 - \left(\frac{r_{i_S, j_S}}{s_{i_S, j_S}} \right)^p \right) \\
&\leq \left(\frac{s_{i_S, j_S}}{\|A\|_2} \right)^p c_{RS}.
\end{aligned}$$

The result follows from the inequalities $s_{i,j} \leq \|A\|_2$ for all i and j. The proofs for other inequalities are similar. □

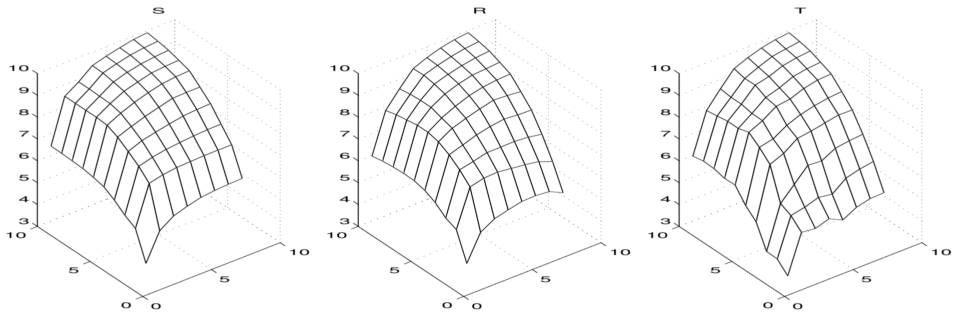

FIG. 1. *Plots of the matrices S, R, and T.*

The theorem above shows that $G_T(\alpha, \beta, p)$ will be a good approximate to $G_S(\alpha, \beta, p)$ if c_{TS} is small. We are, however, unable to provide a useful bound for c_{TS}. In general T seems to differ from S only slightly. To illustrate this, let us consider a small numerical example. (We have also tested other small matrices and found similar behavior.[1])

EXAMPLE. 1. Let $m = 10$, and $n = 8$, and $l = \min(m, n)$. We construct A as (using the notation of MATLAB)

```
[U, r] = qr(rand(m,l));
[V, r] = qr(rand(n,l));
A = U*diag(10*rand(l,1))*V';
```

We take $\mu = 0.5$ and $\lambda = 0.3$ and then construct the three matrices S, R, and T Figure 1 plots the matrices S, R, and T. In general, s_{ij}, r_{ij}, and t_{ij} are close to each other if i and j are not small, i.e., large discrepancy in s_{ij}, r_{ij} and t_{ij} may occur only when the indexes i and j are small. Below we list the average values and the maximums of the relative errors between s_{ij}, r_{ij}, and t_{ij}. Note that the maximums occur generally with small indexes i and j.

relative error	average	max
$(s_{ij} - r_{ij})/s_{ij}$	2.5809e-02	1.0719e-01
$(r_{ij} - t_{ij})/r_{ij}$	4.3807e-02	2.8485e-01

On the other hand, for the tested matrices, the index-pair (i_H, j_H) of the problem (3.17) and corresponding values h_{i_H,j_H} change only slightly for different choices of $H = S$, R, T and $p = 1$, 2, respectively. Below we list the computed results.

	$p = 1$			$p = 2$		
H	i_H	j_H	h_{i_H,j_H}	i_H	j_H	h_{i_H,j_H}
S	6	3	8.72648	7	6	9.50874
R	5	3	8.35768	8	6	9.52802
T	6	4	8.79437	8	6	9.51918

It seems that in general, we can expect the following,

$$i_S + j_S \lesssim i_R + j_R \lesssim i_T + j_T, \quad \text{and} \quad s_{i_S,j_S} \lesssim r_{i_R,j_R} \lesssim t_{i_T,j_T}.$$

[1] Only small matrices are used in our examples since computing S, R, and T involves exhaustive search.

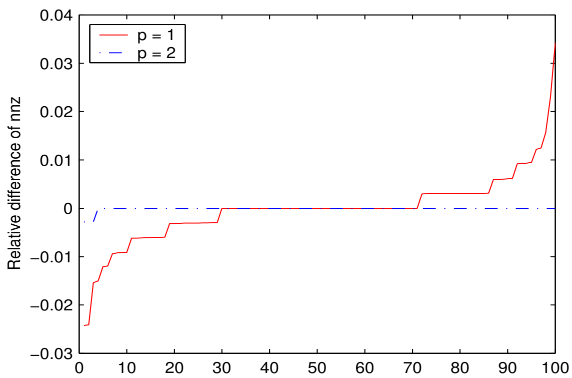

FIG. 2. *Plots of the difference in sparsity of the optimal solutions based on G_T and F for 100 test matrices.*

It is difficult to prove the above assertion in a rigorous fashion. We now, however, provide a proof for a weaker form.

THEOREM 3.2. *Using the notations as above, we have*

$$(\alpha * i_T + \beta * j_T) - (\alpha * i_S + \beta * j_S) \le \frac{t_{i_T,j_T} - t_{i_s,j_s}}{s_{i_S,j_S}}.$$

Proof. Notice that

$$\begin{aligned} & (\alpha * i_T + \beta * j_T) - (\alpha * i_S + \beta * j_S) \\ = \; & G_T(\alpha,\beta,p) - G_S(\alpha,\beta,p) - (t^p_{i_T,j_T} - s^p_{i_S,j_S})/\|A\|_2^p \\ \le \; & (t^p_{i_T,j_T} - t^p_{i_S,j_S})/\|A\|_2^p \\ \le \; & (t^p_{i_T,j_T} - t^p_{i_S,j_S})/s^p_{i_S,j_S}, \end{aligned}$$

completing the proof. □

Similar results can also be proved for r_{i_R,j_R} and t_{i_T,j_T}.

Finally, we show some numerical results to illustrate the difference in sparsity of the low-rank approximations obtained based on the optimal solutions of $G_T(\alpha,\beta,p)$ and $F(\alpha,\beta,p)$. The tested matrices A we considered are constructed as those in Example 1 with $m = 200$ and $n = 160$. We test 100 matrices and compare the total number of nonzeros in the optimal solutions x and y for $G_T(\alpha,\beta,p)$ and $F(\alpha,\beta,p)$ with $\alpha = 1/(2m)$ and $\beta = 1/(2n)$ corresponding to $\lambda = 0.2$ and $\mu = 0.5$. The difference is quite small. Figure 2 plots the sorted relative difference of the total number of nonzeros for the 100 tested matrices.

4. Continuous relaxation for $F(\alpha,\beta,p)$. Now we discuss how to solve the discrete optimization problem (2.13). Clearly the objective function $f(\xi)$ of problem $F(\alpha,\beta,p)$ has at most $m+n$ different values at $\xi = \xi_k$ defined by

$$\xi_k = 1 - \frac{1}{2}\big(\tilde{w}(1)^2 + \ldots + \tilde{w}(k)^2\big), \quad k = 1, 2, \ldots, (m+n) \tag{4.20}$$

for the sorted vector $\tilde{w}$ of $w = (u^T, v^T)^T$ with computed singular vectors u and v.[2] Therefore, the discrete problem $F(\alpha,\beta,p)$ can be solved by a direct method shown as

[2] It is not required to evaluate $f(\xi_k)$ if $\tilde{w}(1:k)$ has only u-components or v-components.

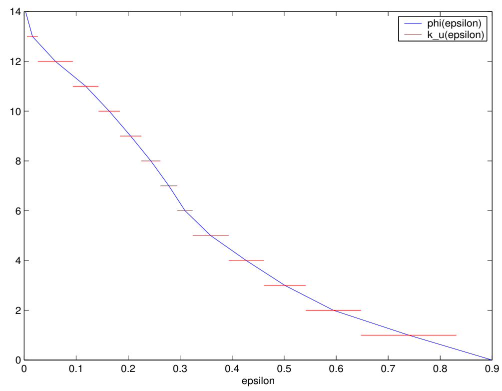

FIG. 3. *Plot for the piece-constant function $i(\xi)$ and its interpolation $\phi(\xi)$.*

the following two steps: 1) evaluate the $m+n$ function values $f(\xi_k)$, $k = 1, 2, \ldots, (m+n)$, and 2) find the minimum and the optimal ξ_{k^*} or k^* for constructing the sparse vectors $x_u(\xi_{k^*})$ and $y_v(\xi_{k^*})$. Clearly, to evaluate the $m+n$ function values $f(\xi_k)$, $k = 1, 2, \ldots, (m+n)$, $m+n$ vector inner-products are required if some updating techniques are used. Roughly speaking, it needs the same cost as that for $x^T Ay$,[3] together with the cost of checking the u-components or v-components for the components of w.

To solve $F(\alpha, \beta, p)$ efficiently, we will reduce $F(\alpha, \beta, p)$ to a discrete form which does not need the matrix-vector product and can be solved easily using a Newton-like iterative method discussed in the next section. Our strategy is to first implicitly construct a continuous approximation of the discrete optimization problem by smoothly interpolating the discrete objective function in (2.13) and finding the conditions that characterize the solution to the continuous problem. The conditions are then transformed back to the discrete problem that we will solve.

To this end, let $\phi(\xi)$, $\psi(\xi)$, and $\omega(\xi)$ be approximations of the piecewise constant functions $i(\xi)$, $j(\xi)$, and $|h(\xi)|/\|A\|_2$ with $h(\xi) = d(x_u, y_v)$ defined as before, respectively,

$$\phi(\xi) \approx i(\xi), \quad \psi(\xi) \approx j(\xi), \quad \omega(\xi) \approx \frac{|h(\xi)|}{\|A\|_2}.$$

Then we seek to solve the corresponding approximate continuous optimization problem,

$$C(\alpha, \beta) = \min_{\xi} \Big\{ \alpha\phi(\xi) + \beta\psi(\xi) - \omega^p(\xi) \Big\}. \tag{4.21}$$

Obviously, the optimal ξ^* satisfies

$$\alpha\phi'(\xi) + \beta\psi'(\xi) - p\omega^{p-1}(\xi)\omega'(\xi) = 0, \tag{4.22}$$

provided that ϕ, ψ, and ω are differentiable.

Now we show how to construct *smooth* functions that interpolate $i(\xi)$ and $j(\xi)$. For simplicity, we set $p = 1$. We choose functions $\phi(\xi)$ and $\psi(\xi)$ defined by the

[3] In the $(i+1)$-th rank-one iteration of our sparse low-rank approximation method, the matrix corresponding to $F(\alpha, \beta, p)$ is the reconstruction error matrix $A_i = A - X_i D_i Y_i^T$.

following integral equations,

$$\int_0^{\phi(\xi)} f_u(t)dt = 1-\xi, \quad \int_0^{\psi(\xi)} f_v(t)dt = 1-\xi,$$

where $f_u(t)$ and $f_v(t)$ are continuous functions interpolating $\{\tilde{u}^2(i)\}$ and $\{\tilde{v}^2(i)\}$, respectively, see Figure 3 for an illustration.

By the proof of Theorem 3.1 in [7], we have

$$1 \geq \frac{|h(\xi)|}{\|A\|_2} \geq 1 - \frac{2\xi}{1-\xi}.$$

Therefore we can choose $\omega(\xi)$ to be the mean of the lower bound and the upper bound,[4]

$$\omega(\xi) = 1 - \frac{\xi}{1-\xi}. \tag{4.23}$$

The functions $\phi(\xi)$ and $\psi(\xi)$ are differentiable because f_u and f_v are continuous, and

$$\phi'(\xi) = -\frac{1}{f_u(\phi(\xi))} \approx -\frac{1}{\tilde{u}^2(i(\xi))}, \quad \psi'(\xi) = -\frac{1}{f_v(\psi(\xi))} \approx -\frac{1}{\tilde{v}^2(j(\xi))},$$

in the above we have used $f_u(\phi(\xi)) \approx \tilde{u}^2(i(\xi))$ and $f_u(\psi(\xi)) \approx \tilde{v}^2(j(\xi))$. On the other hand, $\omega'(\xi) = -(1-\xi)^{-2}$. Hence the optimal ξ approximately satisfies

$$\frac{\alpha}{\tilde{u}^2(i(\xi))} + \frac{\beta}{\tilde{v}^2(j(\xi))} - \frac{1}{(1-\xi)^2} = 0,$$

or equivalently,

$$1-\xi = \left(\alpha\tilde{u}^{-2}(i(\xi)) + \beta\tilde{v}^{-2}(j(\xi))\right)^{-1/2}.$$

Note that if the optimal ξ satisfies $\xi_k \leq \xi < \xi_{k+1}$ where ξ_k is defined in (4.20), then $i(\xi) = i(\xi_k)$, $j(\xi_k) = j(\xi_k)$, and

$$\frac{1}{2}\left(\tilde{w}^2(1) + \ldots + \tilde{w}^2(k)\right) = 1 - \xi_k \approx 1 - \xi,$$

For simplicity, we denote $i_k = i(\xi_k)$ and $j_k = j(\xi_k)$. We conclude that the integer k such that the interval $[\xi_k, \xi_{k+1})$ contains the optimal ξ approximately satisfies

$$\frac{1}{2}\left(\tilde{w}^2(1) + \ldots + \tilde{w}^2(k)\right) = \left(\alpha\tilde{u}^{-2}(i_k) + \beta\tilde{v}^{-2}(j_k)\right)^{-1/2}.$$

Now we can transform the continuous form (4.22) back to the following discrete optimization problem,

$$D(\alpha, \beta) = \min_k |\Phi(k) - \Psi(k)|, \tag{4.24}$$

where

$$\Phi(k) = \frac{1}{2}\left(\tilde{w}^2(1) + \ldots + \tilde{w}^2(k)\right), \quad \Psi(k) = \left(\frac{\alpha}{\tilde{u}^2(i_k)} + \frac{\beta}{\tilde{v}^2(j_k)}\right)^{-1/2}.$$

[4] One can in general choose $\omega(\xi) = 1 - c_0\xi/(1-\xi)$ with a constant $c_0 \in (0, 2)$.

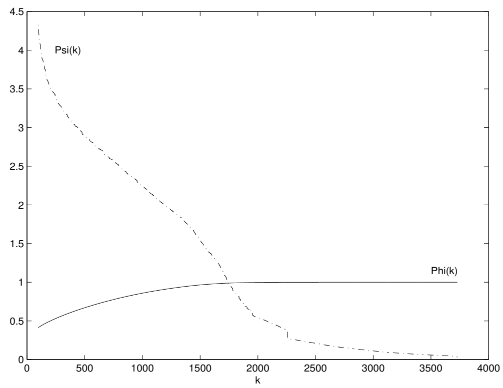

FIG. 4. *Plot for the discrete curves $\Phi(k)$ (solid line) and $\Psi(k)$ (dashdot line).*

It is not difficult to verify that the integers $i_k \geq 1$ and $j_k \geq 1$ determined by k satisfy the following equations

$$\left\{ \begin{array}{rcl} i_k + j_k &=& k \\ \min\left\{\tilde{u}^2(i_k), \tilde{v}^2(j_k)\right\} &=& \tilde{w}^2(k) \end{array} \right. . \tag{4.25}$$

It should be pointed out that the indexes i_k and j_k may not be uniquely determined by k if $|\tilde{w}_k| = |\tilde{w}_{k+1}|$. We will have a detailed discussion about this in the next section. Ignoring, for the moment, the possibility of those indexes being multi-valued, we can easily see that the discrete function $\Psi(k)$ is *decreasing* while $\Phi(k)$ is *increasing*. Figure 4 plots the graphs of $\Psi(k)$ and $\Phi(k)$ for a matrix of order 2331×1398 Obviously, the monotonicity of $\Phi(k)$ and $\Psi(k)$ is very helpful for constructing a *globally* convergent method to solve the minimization problem (4.24). Such an iterative method will be considered in the next section.

5. Discrete Newton-like iteration. In this section, we discuss a Newton-like iteration for solving the optimization problem (4.24).

Assume we have an approximation k_ℓ to the optimal k^*. Denote $\Psi_\ell = \Psi(k_\ell)$ and $\Phi_\ell = \Phi(k_\ell)$. We choose suitably two secants $\delta\Psi_\ell$ and $\delta\Phi_\ell$ for $\Psi(k)$ and $\Phi(k)$, respectively. Replacing $\Psi(k)$ and $\Phi(k)$ in (4.24) by the secant lines $\Psi_\ell + (k - k_\ell)\delta\Psi_\ell$ and $\Phi_\ell + (k - k_\ell)\delta\Phi_\ell$, and solving the problem

$$\min_k |(\Psi_\ell + (k - k_\ell)\delta\Psi_\ell) - (\Phi_\ell + (k - k_\ell)\delta\Phi_\ell)| \tag{5.26}$$

yields the following discrete Newton-like iteration

$$k_{l+1} = k_\ell + \left\lfloor \frac{\Psi_\ell - \Phi_\ell}{\delta\Phi_\ell - \delta\Psi_\ell} \right\rfloor, \tag{5.27}$$

where $\lfloor a \rfloor$ is the floor function giving the largest integer no greater than a. The monotonicity of $\Phi(k)$ and $\Psi(k)$ also insures the uniqueness of the solution to (4.24).

As mentioned at the end of the last section, we need to investigate whether the objective function for (4.24) is well-defined or not Note that $\Phi(k)$ is always well defined.

If $|\tilde{w}(k)| > |\tilde{w}(k+1)|$, i_k and j_k are uniquely determined by k; i_k is the number of the u-components of subvector $\tilde{w}(1:k)$ and j_k is the number of the v-components

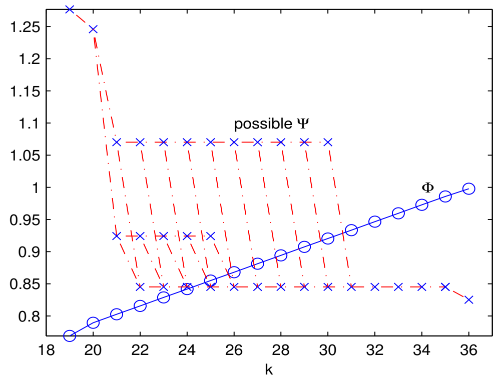

FIG. 5. *The curves $\Phi(k)$ (solid circle line) and the possible curve $\Psi(k)$ (dashed x-mark lines) in a bracketing interval.*

of $\tilde{w}(1:k)$. In this case, $\Psi(k)$ is also well-defined. If $|\tilde{w}(k)| = |\tilde{w}(k+1)|$, i_k and j_k may be not unique and then $\Psi(k)$ may have different values depending on the choices of i_k and j_k, see Figure 5 for a detailed illustration. There are several important computational details need to discuss for the discrete Newton-like iteration.

w-CONSTANT INTERVALS. We call $(a,\, b)$ a w-constant interval if $b-a>1$ and

$$|\tilde{w}(a)| > |\tilde{w}(a+1)| = \cdots = |\tilde{w}(b)| > |\tilde{w}(b+1)|.$$

Obviously, if all components in subvector $\tilde{w}(a+1:b)$ are u-components or all components are v-components, $\Psi(k)$ is well defined for $k \in [a+1,\, b-1]$. Otherwise, if $\tilde{w}(a+1:b)$ has both u-components and v-components, then for any integer $k \in \big[a+1, b-1\big]$, i_k can be any integer in the interval $\big[i_a, \min(i_b, k-j_a)\big]$ and $j_k = k - i_k \in \big[\max(j_a, k-i_b),\, k-i_a\big]$. This will lead to ill-defining of $\Psi(k)$ because $\Psi(k)$ may have multiple values depending on the choice of i_k and j_k if $\alpha \neq \beta$. However, we can ignore the multi-definition of $\Psi(k)$ if $k^* \notin \big[a,\, b\big]$ because in this case the choice of pair (i_k, j_k) does not affect the convergence of the discrete Newton-like iteration.

BRACKETING INTERVALS. We should pay more attention to w-constant interval $\big(a,\, b\big]$ if the optimal $k^* \in \big[a+1,\, b-1\big]$. Such a w-constant interval is called a bracketing interval. For a bracketing interval $\big(a,\, b\big]$, the optimal k^* should be $k^* = i^* + j^*$ with i^* and j^* solving

$$\min_{i_a \le i \le i_b,\, j_a \le j \le j_b} \big|\Phi(i+j) - \Psi(i+j)\big|. \tag{5.28}$$

It is easy to see that $\Phi(k)$ is uniquely defined and is a linear function in the interval $\big(a,\, b\big)$ while $\Psi(k)$ will have multiple values depending on i_k and j_k. Figure 5 plots for the graph of $\Phi(k)$ and many possible graphs for $\Psi(k)$ in a bracketing interval.

The existence of bracketing intervals make the problem (4.24) more complicated because it is necessary to solve (5.28) with possible multi-valued $\Psi(k)$ for each $k \in [a+1,\, b-1]$. Note that there are no easy methods except for exhaustive enumeration for solving (5.28). Fortunately, bracketing intervals seldom occur and the length of the occurred bracketing interval is generally very small. For all the numerical experiments we did, the largest length of the occurred bracketing interval is $b-a=3$.

CHOOSING THE SECANTS $\delta\Phi_\ell$ AND $\delta\Psi_\ell$. There are many ways to choose the secants $\delta\Phi_\ell$ and $\delta\Psi_\ell$ which are to be used to construct the next iterate. Let k_0 be the smallest integer such that $\min(i_{k_0}, j_{i_0}) = 1$. For the initial $\delta\Phi_0$ and $\delta\Psi_0$, we set

$$\delta\Phi_0 = (\Phi(k_0+d) - \Phi(k_0))/d, \quad \text{and} \quad \delta\Psi_0 = (\Psi(k_0+d) - \Psi(k_0))/d.$$

with a step size $d \le \frac{1}{2}(m+n-k_0)$, for example, we have used $d = \min(100, \lfloor\frac{1}{2}(m+n-k_0)\rfloor)$. In general, one can choose $\delta\Phi_\ell = (\Phi_\ell - \Phi_{l-1})/(k_\ell - k_{l-1})$ and $\delta\Psi_\ell = (\Psi_\ell - \Psi_{l-1})/(k_\ell - k_{l-1})$. However a better way is to compute $\delta\Phi_\ell$ and $\delta\Psi_\ell$ by

$$\delta\Phi_\ell = \frac{\Phi_{\max} - \Phi_{\min}}{k_{\max} - k_{\min}} \quad \text{and} \quad \delta\Psi_\ell = \frac{\Psi_{\max} - \Psi_{\min}}{k_{\max} - k_{\min}},$$

if we know an interval $[k_{\min}, k_{\max}]$ that contains the optimal k^*, where

$$\begin{aligned} \Phi_{\min} &= \Phi(k_{\min}), \quad \Phi_{\max} = \Phi(k_{\max}), \\ \Psi_{\min} &= \Psi(k_{\min}), \quad \Psi_{\max} = \Psi(k_{\max}). \end{aligned}$$

Initially, we can set $k_{\min} = k_0$ and $k_{\max} = m+n$. The interval $[k_{\min}, k_{\max}]$ will be updated at each iteration step as follows If $\Psi_\ell < \Phi_\ell$, we then decrease $k_{\max}$ and reset $k_{\max} = k_\ell$, otherwise if $\Psi_\ell \le \Phi_\ell$, we increase $k_{\min}$ and reset $k_{\min} = k_\ell$.

AVOIDING INFINITE LOOP. To avoid infinite loop of the Newton-like iteration in the case when $\left\lfloor \frac{\Psi_\ell - \Phi_\ell}{\delta\Phi_\ell - \delta\Psi_\ell} \right\rfloor = 0$, we need to slightly modify k_{l+1} at the l-th iteration so that

$$\begin{array}{rcccll} k_\ell + 1 & \le & k_{l+1} & \le & k_{\max} - 1, & \text{if} \quad k_\ell < k^*, \quad \text{or} \\ k_{\min} + 1 & \le & k_{l+1} & \le & k_\ell - 1, & \text{if} \quad k_\ell > k^*. \end{array} \tag{5.29}$$

It is not difficult to check if the inequality $k_\ell \le k^*$ holds or not for given k_ℓ by comparing Φ_ℓ and Ψ_ℓ. Now we are ready to present our algorithm for solving the discrete optimization problem (4.24).

Algorithm DNI (Discrete Newton-like Iteration) Given two vectors u and v, this algorithm solves the minimization problem (4.24).

1. [Initialization]
- **1.1** Sort $\tilde{w} = Pw$, $\tilde{w}(k) \leftarrow \tilde{w}(k)^2$, $\tilde{u}(i) \leftarrow \tilde{u}(i)^2$, $\tilde{v}(j) \leftarrow \tilde{v}(j)^2$.
- **1.2** Determine the smallest k_0 such that $i_{k_0} \ge 1$, $j_{k_0} \ge 1$, and set an initial trust interval $[k_{\min}, k_{\max}]$.

2. For $\ell = 0, 1, 2, \cdots$, until convergence,
- **2.1** Compute $\Phi_\ell = \Phi(k_\ell)$ and $\Psi_\ell = \Psi(k_\ell)$.
- **2.2** Modify the trust interval $[k_{\min}, k_{\max}]$ and check convergence. If $k_{\min} = k_{\max}$, or $|\Psi_\ell - \Phi_\ell| < \tau$, stop.
- **2.3** Determine the secants $\delta\Phi_0$ and $\delta\Psi_0$
- **24** One Newton iteration. $k_{\ell+1} = k_\ell + \left\lfloor \frac{\Psi_\ell - \Phi_\ell}{\delta\Phi_\ell - \delta\Psi_\ell} \right\rfloor$. Slightly modify k_{l+1} as in (5.29) if necessary.
- **2.5** Determine the w-constant interval $(a, b]$ of which covers k, and compute $\Psi_a = \Psi(a)$, $\Psi_b = \Psi(b)$, $\Phi_a = \Phi(a)$, $\Phi_b = \Phi(b)$.
- **2.6** If $[a, b]$ does not cover the optimal k^*, compute $k_{\ell+1}$ by

$$k_{\ell+1} = \begin{cases} a & \text{if } \Psi_a \le \Phi_a \\ b & \text{if } \Psi_b > \Phi_b \end{cases},$$

otherwise turn to **step 3**.

3 Compute $k^* \in [a, b]$.

TABLE 1
Computation cost of DNI and the direct method for matrix `cisi`.

	min flops	max flops	mean flops	total flops	total CPU(s)
DNI	9.1500e+3	4.1130e+4	2.6966e+4	2.2382e+6	1.7600e+0
Direct Method	2.4500e+5	2.4546e+5	2.4525e+5	2.0110e+7	9.3340e+2

The convergence property of DNI is relatively easy to establish. Notice that the length of the interval $[k_{\min}, k_{\max}]$ will be reduced at least one at each iteration. Therefore the above algorithm is guaranteed to converge in at most $m + n - k_0$ iterations. For all the numerical experiments we did, DNI converges within about 10 iterations averagely.

The iterative method DNI is very efficient. In Table 1, we list the computation cost of DNI and the direct method for solving (2.13) with $\lambda = 0.1$ when they were used, respectively, together with the sparse low-rank approximation algorithm (SLRA) for computing sparse low-rank approximations of the matrix A. Each sparse low-rank approximation consists of 83 rank-one sparse approximations to $A_0 = A$ and the reconstruction error matrices $A_i = A - X_i D_i Y_i^T$ of the previously computed sparse approximations $X_i D_i Y_i^T$, $i = 1, 2, \ldots, 82$.

Combining with SLRA, we have the following overall algorithm for computing spare low-rank approximations.

Algorithm Penalized SLRA with DNI. Given a matrix A, penalty parameters α, β, and tolerance τ, this algorithm produces a positive diagonal matrix D_k, and sparse matrices X_k and Y_k such that $B_k \equiv X_k D_k Y_k^T$ is a low-rank approximation of A with relative reconstruction error τ.

1. Set $A_0 = A$ and $\eta_0 = \|A\|_F^2$.
2. For $i = 1, 2, \cdots$, until convergence
  2.1 Compute approximately the largest singular triplet (σ, u, v) of matrix A_{i-1}.
  2.2 Determine integer k^* and corresponded $i^* = i_{k^*}$ and $j^* = j_{k^*}$, using Algorithm DNI.
  2.3 Construct the sparse vectors x_i and y_i as defined in (2.11) with i^* and j^*, and determine the optimal d_i for the minimization problem $\min_d \|A_{i-1} - x_i d y_i^T\|$.
  2.4 Set $A_i = A_{i-1} - x_i d_i y_i^T$ and $\eta_i = \eta_{i-1} - d_i^2$.
  2.5 Check convergence: if $\eta_i < (\tau\|A\|_F)^2$, then set $k = i$ and turn to Step 3.
3. Set $D_k = \text{diag}(d_1, \cdots, d_k)$, $X_k = [x_1, \cdots, x_k]$ and $Y_k = [y_1, \cdots, y_k]$.

6. Numerical Experiments. In this section, we will present several numerical experiments to illustrate the penalized SLRA with DNI (SLRA-DNI). We will compare SLRA-DNI with the SLRA using a strategy of mixed-sorting and variable-tolerance (SLRA-MV) proposed in [7]. The test matrices we will use are the same as in [7]. (The reader is referred to [7] for detailed descriptions of those test matrices.) In all of the tests we always use four Lanczos bidiagonalization iterations for computing the approximate largest singular vectors of the deflated matrices $A_i = A - X_i D_i Y_i^T$ at

TABLE 2
Average of the discrete secant iterations of DNI.

λ	0.05	0.10	0.15	0.20	0.25	0.30	0.35	0.40	0.45	0.50
ash958	9.5	8.6	9.2	8.3	8.1	7.9	7.7	7.0	7.2	6.6
illc1033	11.2	10.3	9.2	9.6	9.0	8.7	8.4	8.1	7.7	70
cisi	12.8	11.3	10.4	10.0	9.9	9.4	9.4	8.9	88	8.4
cacm	11.3	10.6	10.4	10.2	9.9	9.8	9.8	97	9.1	9.3
med	14.6	12.8	12.0	11.1	11.0	10.2	97	9.5	9.1	8.9
npl	11.2	11.5	11.9	12.0	11.7	114	11.2	11.0	11.0	10.7
orsirr2	10.3	9.9	9.6	8.7	84	8.4	8.1	7.9	7.6	7.5
e20r1000	12.8	12.1	11.3	111	10.9	10.6	10.4	10.0	9.9	9.6

each iteration. The penalty factors α and β are chosen as in (2.9) with $\mu = 1/2$ and

$$\lambda = \texttt{005:0.05:0.5}. \tag{6.30}$$

In genal, larger values of λ produce sparser factors X_k and Y_k.

Behavior of DNI iterations. Compared with SLRA-MV, the penalized SLRA-DNI requires additional computations for the DNI part. The main cost of DNI is the evaluations of $\Psi(k)$ and $\Phi(k)$ for different k. Clearly they can be easily updated with about $2(m+n)$ flops and the total cost of DNI is about $2k_{DNI}(n+m)$, where k_{DNI} is the iteration number of DNI. In all the numerical experiments we did, the iteration number k_{DNI} required is relatively small. In Table 2 we list the average number of iterations for each matrix and λ, and generally $k_{DNI} \approx 10$. Comparing with the cost for computing approximately the singular vectors with four Lanczos bidiagonalization iterations, the cost for DNI can be negligible. Therefore the SLRA-DNI requires similar cost as SLRA with the variable-mixed sorting technique if the approximate ranks (the number of columns of the block X_k) of the computed low-rank approximations are approximately equal.

Although it is possible that the optimal integer k^* is in a w-constant interval which leads to complication in the algorithm, the bracketing intervals seldom occur and the length $b-a$ of the occurred bracketing intervals are rather small in general. Among all the eight tested matrices, there are no bracketing intervals occurred for `illc1033`, `npl`, `orsirr2`, and `e20r1000` and for the choice of λ = `0.05:0.05:0.50`, while for the other four matrices, `ash958`, `cisi`, `cacm` and `med`, the bracketing intervals are rather small. See the table listed below, where the integer p in the form $p(q)$ is the maximal length of the q bracketing intervals occurred.

λ	0.05	0.10	015	0.20	0.25	0.30	0.35	0.40	0.45	0.50
ash958			2(1)		2(2)				2(1)	
cisi			3(1)			2(1)				
cacm	2(1)	2(3)	2(1)	2(2)	2(1)	2(1)		2(2)	3(1)	2(1)
med										2(1)

Sensitivity of SLRA-DNI to parameter λ. We now show how the parameter λ affects the approximation rank and the storage $\texttt{nnz}(X_k) + \texttt{nnz}(Y_k) + k$ required for the factors X_k, Y_k, and D_k computed by SLRA-DNI. For the eight test matrices we considered, the approximation ranks and storages remain relatively stable with respect to the choice $\lambda = 0.05 : 0.05 : 0.5$. See Figure 6 for the results computed by

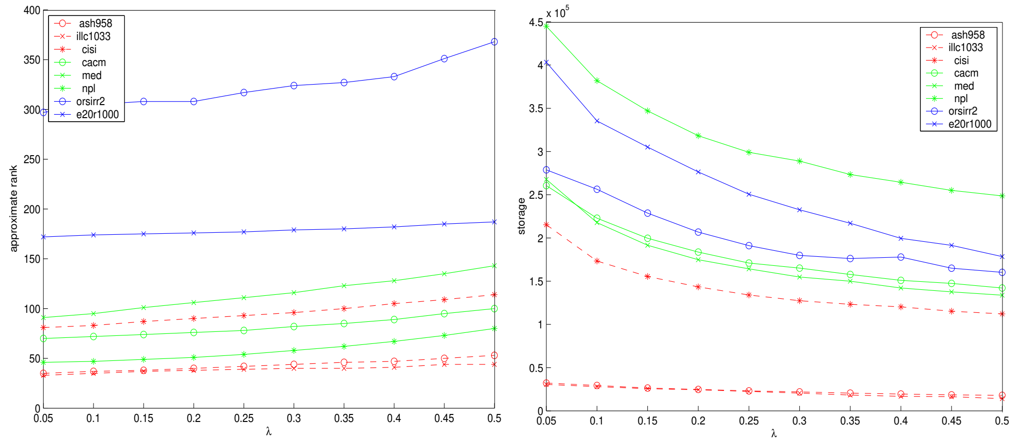

FIG. 6. *Plots for ranks (left) and numbers of nonzeros of X_k and Y_k (right) vs starting epsilon for the penalized SLRA-DNI.*

SLRA-DNI. SLRA-DNI seems to be more stable than SLRA-MV. To show this, we choose the parameter ϵ for SLRA-MV to be

$$\epsilon = 0.05 : 0.05 : 0.5,$$

as we did in in [7]. In Figure 7 we plot the storage versus the approximate rank for the two SLRA algorithms. The results show that SLRA-DNI always gives sparser factors than SLRA-MV when the computed approximations have approximately equal ranks. Figure 8 plots the storages versus reconstruction errors for the four term-document matrices in our test set for the methods TSVD, SLRA-MV, SLRA-DNI, and SDD. The storage requirement of SLRA-DNI grows relatively slowly compared with that of TSVD. Notice that the storage requirement for SDD is accounted as $(\mathtt{nnz}(X_k) + \mathtt{nnz}(Y_k))/32 + \mathtt{nnz}(D_k)$, because SDD gives factors X_k and Y_k with elements chosen from set $\{-1, 0, 1\}$ and the storage for each nonzero in X_k and Y_k is only 2 bits (or 1/32 the storage of a double).

Comparison with other methods. Now we make a comparison of the computational costs for several of the existing methods including ours. Obviously, in the $(j+1)$th rank-one approximation step, the major cost is the matrix-vector product $A_j x = Ax - X_j(D_j(Y_j^T x))$. Seven matrix-vector products are required for four Lanczos bidiagonalization iterations for approximately computing the largest left and right singular vectors of A_j. Let r be the average number of nonzeros for each column in the factor X_k or Y_k. Then each rank-one approximation requires seven matrix-vector of the original matrix A and $7(4r+1)j$ flops (multiplications and divisions count one flop each) and the normalization cost for 4 m-dimensional vectors and 4 n-dimensional vectors. So the total cost for the sparse approximation $B_k = X_k D_k Y_k^T$ is about $7k$ matrix-vector multiplication of the original matrix A and another $7(2r + 1/2)k(k+1) + 36k(m+n)$ flops. Including the cost of DNI that is about $2k_{DNI}(n+m)$ with $k_{DNI} \approx 10$, the total flops required are about

$$(14r + 3.5)k(k+1) + 36k(m+n) + 14k\mathtt{nnz}(A).$$

In Table 3, we list several computation results for several of the existing methods with different parameters. As we did in [7], for each sparse test matrix, we first

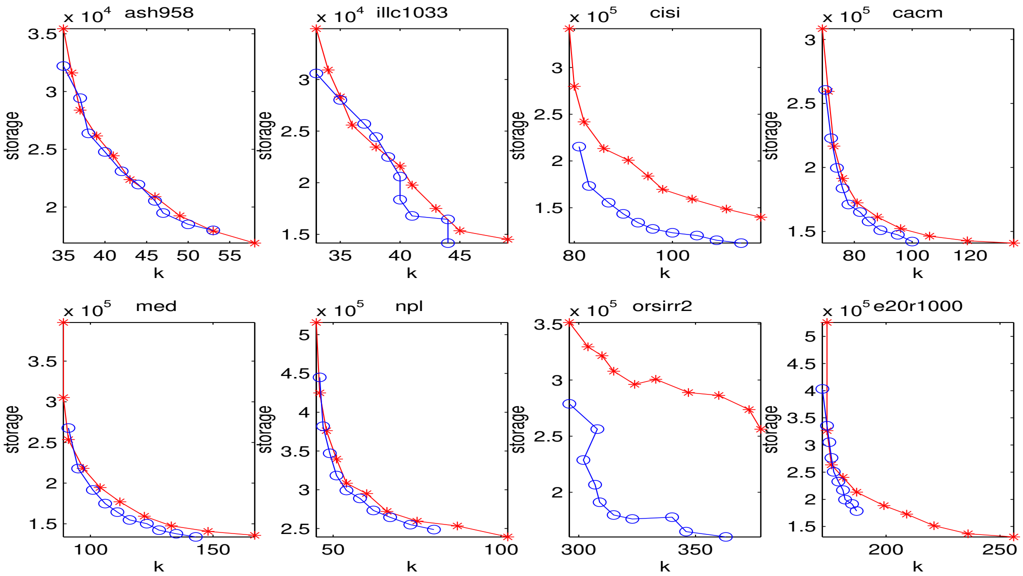

FIG. 7. *Plots storage versus approximate rank for the penalized SLRA-DNI (−o−) and SLRA-MV (− ∗ −).*

compute a sparse approximation using 300 column/row of the test matrix that are determined by SPQR [6]. The reconstruction error then is used as the tolerance for other methods. For SLRA-MV, we take the starting error $\epsilon = 0.1$ and $\epsilon = 0.3$, respectively. For SLRA-DNI, we set $\lambda = 0.1$ and $\lambda = 0.3$. There are several initialization strategies used for the inner loops of SDD. We choose the "threshold" strategy ($choice = 1$) and the "cycling" strategy ($choice = 2$). In the table, column nnz is the number of nonzeros in factors X_k and Y_k for methods except SPQR for which nnz is the number of nonzeros in X_k, Y_k and H_k of the approximation $B_k = X_k H_k Y_k^T$. For the four tested tern-document matrices, penalized SLRA is relatively faster than other methods and has in general less storage requirement than other methods except SDD. As we mentioned before, SDD gives $B_k = X_k D_k Y_k^T$ with factors X_k and Y_k with elements chosen from set $\{-1, 0, 1\}$ and certainly has the smallest storage requirement. However, SDD depends on the initialization strategy used for the inner iterations. For different initialization strategies, SDD tends to give different sparse approximations. At each outer iteration step of SDD, the "threshold" strategy, for example, is to determine a column of the previously constructed matrix $A_k = A - X_k D_k Y_k^T$ such that the column has its squared norm larger than the average of those of all the columns. This process will tend to be expensive if the columns of A_k have almost the same norms. It may also be possible for an infinite-loop to occur if certain method for selecting a column candidate such as that described in SDDPACK is used [3].[5] Such phenomenon seem to have occurred for matrix `npl`. In general, to achieve the same reconstruction error as that of TSVD or SLRA, we need to SDD to compute an approximation with a much larger rank k.

7. Concluding Remarks. Computing low-rank approximations of matrices is a very important matrix computation problem that has a variety of applications. The

[5] the MATLAB implementation is significantly slower than the C implementation.

TABLE 3
Comparisons on cost for different methods

matrix	method		k	nnz	CPUtime(s)	flops
cisi	TSVD		72	471600	668.61	8.6541e+9
	SLRA-MV	$\epsilon = 0.1$	80	279695	33.89	2.4947e+8
		$\epsilon = 0.3$	95	183809	26.25	2.2194e+8
	SLRA-DNI	$\lambda = 01$	83	173233	18.34	1.9706e+8
		$\lambda = 0.3$	96	127298	17.25	1.9711e+8
	SDD	choice = 1	318	88503	254.75	5.1115e+8
		choice = 2	300	123804	664.76	7.2086e+8
	SPQR		300	129480	49.55	3.2661e+8
cacm	TSVD		63	422982	414.30	5.3256e+9
	SLRA-MV	$\epsilon = 01$	71	259321	30.81	2.2781e+8
		$\epsilon = 0.3$	88	161162	24.66	2.1046e+8
	SLRA-DNI	$\lambda = 0.1$	72	222786	20.32	2.1361e+8
		$\lambda = 0.3$	82	165043	17.80	2.0415e+8
	SDD	choice = 1	217	98932	198.34	4.1030e+8
		choice = 2	205	89624	630.60	3.6730e+8
	SPQR		300	134004	51.14	2.9662e+8
med	TSVD		80	522960	726.83	9.3770e+9
	SLRA-MV	$\epsilon = 0.1$	89	305015	42.08	2.9025e+8
		$\epsilon = 0.3$	112	176964	29.33	2.4306e+8
	SLRA-DNI	$\lambda = 0.1$	95	217701	23.68	2.4540e+8
		$\lambda = 0.3$	116	154644	22.14	2.4288e+8
	SDD	choice = 1	335	31357	178.23	2.5482e+8
		choice = 2	338	32368	654.60	2.5075e+8
	SPQR		300	120319	43.06	2.9364e+8
npl	TSVD		41	645791	529.71	58064e+9
	SLRA-MV	$\epsilon = 0.1$	46	424917	56.03	3.3373e+8
		$\epsilon = 0.3$	60	295024	46.86	3.6177e+8
	SLRA-DNI	$\lambda = 0.1$	47	381847	31.04	3.2627e+8
		$\lambda = 0.3$	58	288838	29.49	3.5174e+8
	SDD	choice = 1	—-	—-	—-	—-
		choice = 2	120	460073	1047.93	8.8916e+8
	SPQR		300	227649	116.38	5.8994e+8

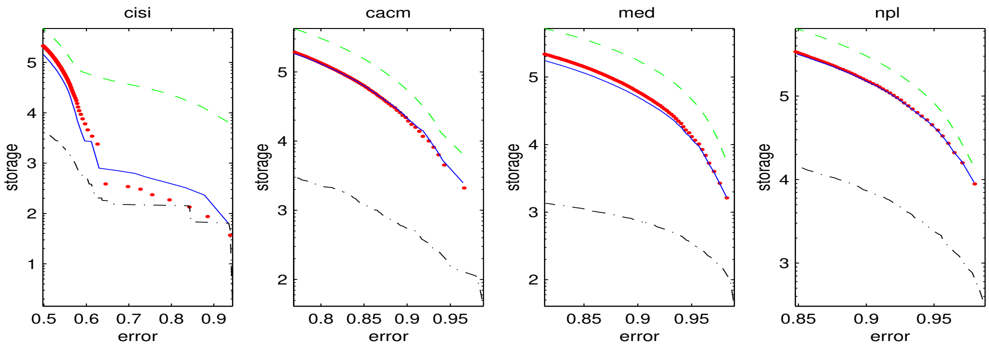

FIG. 8. *Plots storage versus reconstruction error for penalized SLRA-DNI (solid line), SLRA-MV (points), and truncated SVD (dashed line), and SDD (dashdot line).*

large sizes and sparsity properties of the matrices arising from some of the applications entail that we find low-rank approximations that themselves also possess some sparsity properties. We continue our research on this problem following the general framework proposed in [7]: we formulate the sparse low-rank approximation problem as an $(m+n+1)$-dimensional penalized optimization problem, and successfully reduce the penalized optimization problem to a simpler 1-D form that can be solved numerically by a discrete Newton-like iteration method (DNI). Numerical experiments show that our penalized method is more robust and produces approximations with lower ranks (less column numbers) and/or more sparser factors.

Acknowledgment. The authors also want to thank the anonymous referees for their comments and suggestions that greatly improved the presentation of the paper.

REFERENCES

[1] G. H. Golub and C. F. Van Loan. *Matrix Computations.* Johns Hopkins University Press, Baltimore, Maryland, 3nd edition, 1996.
[2] T.G. Kolda and D.P. O'Leary. A semidiscrete matrix decomposition for latent semantic indexing in information retrieval. ACM Trans. Information Systems, 16:322-346, 1998.
[3] T.G. Kolda and D.P. O'Leary. Algorithm 805: Computation and uses of the semidiscrete matrix decomposition. ACM Trans. Mathematical Software, 26:416-437, 2000.
[4] D.P. O'Leary and S. Peleg. Digital image compression by outer product expansion. IEEE Transactions on Communications, 31:441–444, 1983.
[5] H. Simon and H. Zha. Low-rank matrix approximation using the Lanczos bidiagonalization process. *SIAM Journal of Scientific Computing,* 21:2257–2274, 2000.
[6] G.W. Stewart. Four algorithms for the efficient computation of truncated pivoted QR approximation to a sparse matrix. *Numerische Mathematik*, 83:313–323, 1999.
[7] Z. Zhang, H. Zha and H. Simon. Low-rank approximations with sparse factors I: basic algorithms and error analysis. *SIAM Journal on Matrix Analysis and Applications*, 23:706-727, 2002.