Low-Rank Incremental Methods for Computing Dominant Singular Subspaces

C. G. Baker[*] K. A. Gallivan[*] P. Van Dooren[†]

Abstract

This paper describes a generic low-rank incremental method for computing the dominant singular triplets of a matrix via a single pass through the matrix. This work unifies several efforts previously described in the literature. We tie the operation of the proposed method to a particular optimization-based eigensolver. This allows the description of novel methods exploiting multiple passes through the matrix.

1 Introduction

Given a matrix $A \in \mathbb{R}^{m\times n}, m \geq n$, the *Singular Value Decomposition* (SVD) of A is:

$$A = U \begin{bmatrix} \Sigma \\ 0 \end{bmatrix} V^T,$$

where U and V are $m \times m$ and $n \times n$ orthogonal matrices, respectively, and Σ is a diagonal matrix whose elements $\sigma_1, \ldots, \sigma_n$ are real, non-negative and non-increasing. The σ_i are the *singular values* of A, and the first n columns of U and V are the *left and right singular vectors* of A, respectively. Given $k \leq n$, the singular vectors associated with the largest k singular values of A are the *rank-k dominant singular vectors*. The subspaces associated with these vectors are the *rank-k dominant singular subspaces*.

The components of the SVD are optimal in many respects [1], and these properties result in the occurrence of the SVD in numerous analyses, e.g., principal component analysis, proper orthogonal decomposition, the Karhunen-Loeve transform. These applications of the SVD are frequently used in problems related to, e.g., signal processing, model reduction of dynamical systems, image and face recognition, and information retrieval.

A common trait among these problems is the size of the data. Such methods often lead to a matrix that has many more rows than columns. It is matrices like this that are of interest in this paper, and we assume from this point that $m \gg n$. Another similarity, the focus of this paper, is that these methods do not employ all singular triplets of A. Instead, they require only the dominant k singular triplets or dominant rank-k singular subspaces, where $k \ll n$.

One drawback of the SVD is the cost of its computation. For a matrix with more rows than columns, this is usually done by computing the QR factorization of A and the SVD of the R factor:

$$A = QR = Q(\hat{U}\Sigma V^T) = (Q\hat{U})\Sigma V^T \doteq U\Sigma V^T \ .$$

Alternatively, if A has full column-rank, the SVD can be computed via the eigendecomposition of $A^T A = V\Sigma^2 V^T$:

$$A = (AV\Sigma^{-1})\Sigma V^T \doteq U\Sigma V^T \ .$$

[*]Florida State University, USA
[†]Université catholique de Louvain, Belgium

The latter method is less expensive. However, obtaining U in this manner is not as robust as the other approaches [1]. Furthermore, both of these methods require $O(mn^2)$ floating point operations (flops), even if only the dominant SVD is desired. For large matrices, this may not be feasible.

In some applications, the production of the matrix A occurs in an incremental fashion, with new columns being appended to the existing matrix. This has motivated the description of numerous SVD updating methods, e.g., [2, 3]. These methods, while they may suffer from an overall higher complexity, benefit in a number of ways. The latency of producing new columns of A can be hidden by the intermediate computation. Also, some subspace tracking applications (e.g., computer vision, latent semantic indexing) require an "online" computation, i.e., that the current SVD be available between updates. By incorporating the columns into the current decomposition as they are produced, these methods avoid the requirement to store the matrix. However, these same applications often require only the dominant singular subspaces and/or singular values. Incremental methods which compute the full SVD do so at much higher computational expense.

A review of the literature reveals several methods which aim to reduce the complexity of computing the dominant singular triplets [4, 5, 6, 7, 8, 9]. Repeatedly and independently developed, these methods approximate the dominant SVD after a single pass through the matrix. Each group of columns from A is used to update a low-rank factorization which approximates the dominant SVD of A. By incrementally consuming the matrix and maintaining only a low-rank factorization, these methods are able to approximate the dominant SVD using significantly less memory and computation than direct methods. Previous work [5, 8] has shown that the resulting factorization in some cases captures almost all of the information from the dominant SVD.

This paper describes this family of low-rank incremental methods. We propose a generic incremental framework which unifies the previous approaches and allows the description of more efficient methods. We present a novel analysis of these methods which relates their operation to a class of iterative methods for approximating the eigenvalues of A^TA, which allows the description of an iterative approach for the scenario where multiple passes through A are permitted.

2 A Generic Low-Rank Incremental SVD

This section outlines a block, incremental technique for estimating the dominant left and right singular subspaces of a matrix. At the heart of this method is the ability to maintain a low-rank factorization from step to step. This is done via the truncation of smaller singular values, which requires decoupling the associated subspaces. Section 2.1 introduces the subspace separation technique. Section 2.2 describes a generic low-rank incremental SVD based on this approach. Section 2.3 discusses previous incremental methods and how they fit into this framework.

2.1 A Generic Separation Technique

Given an $m \times (k+l)$ matrix M, $m \gg k+l$, consider an orthogonal factorization

$$M = QB \ ,$$

where $Q \in \mathbb{R}^{m\times(k+l)}$, $Q^TQ = I$, and $B \in \mathbb{R}^{(k+l)\times(k+l)}$. Consider the SVD of B and partition it conformally as

$$B = U\Sigma V^T = \begin{bmatrix} U_1 & U_2 \end{bmatrix} \begin{bmatrix} \Sigma_1 & 0 \\ 0 & \Sigma_2 \end{bmatrix} \begin{bmatrix} V_1 & V_2 \end{bmatrix}^T \ ,$$

where U_1, Σ_1, and V_1 contain the largest k and U_2, Σ_2 and V_2 contain the smallest l left singular vectors, singular values and right singular vectors of B, respectively. Define *orthogonal* transformations G_u and G_v such that they block diagonalize the singular vectors of B:

$$G_u^TU = \begin{bmatrix} S_u & 0 \\ 0 & T_u \end{bmatrix} \quad \text{and} \quad G_v^TV = \begin{bmatrix} S_v & 0 \\ 0 & T_v \end{bmatrix} \ . \tag{1}$$

Applying these transformations to B yields $B_{new} = G_u^T B G_v$. These transformations rotate B to a coordinate system where its left and right singular bases are block diagonal. It follows that B_{new} has the form

$$B_{new} = G_u^T B G_v = \begin{bmatrix} S_u \Sigma_1 S_v^T & 0 \\ 0 & T_u \Sigma_2 T_v^T \end{bmatrix} .$$

The SVD of the block diagonal matrix B_{new} also has a block diagonal structure. This gives a new factorization of M,

$$\begin{aligned} M &= QB = (QG_u)(G_u^T B G_v) G_v^T \\ &\doteq Q_{new} B_{new} G_v^T = Q_{new} \begin{bmatrix} T_u \Sigma_1 T_v^T & 0 \\ 0 & S_u \Sigma_2 S_v^T \end{bmatrix} G_v^T , \end{aligned}$$

whose partitioning identifies bases for the dominant left and right singular subspaces of M in the first k columns of Q_{new} and G_v.

It should be noted that G_u is not uniquely defined by Equation (1). This definition admits any G_u whose first k columns are some orthonormal basis for the dominant left singular subspace of B and whose last l columns therefore are some orthonormal basis for the dominated left singular subspace of B. This is also the case, *mutatis mutandis*, for G_v. The result is that the choices for G_u and G_v affect the form of B_{new}: it may have structure (e.g., diagonal, triangular) or not.

2.2 An Incremental Method

The technique of the previous section can be used to define a generic method for computing approximate bases for the left and right dominant singular subspaces via a single pass through the columns of an $m \times n$ matrix A. The procedure begins with an orthogonal factorization of the first l_1 columns of A, $Q_1 B_1 = A_{(1:l_1)}$. The right space basis is initialized to $W_1 = I_{l_1}$. At each step j, new columns from A are used to expand the rank of the current factorization $Q_{j-1} B_{j-1} W_{j-1}^T$. Then the technique from Section 2.1 is used to decouple the dominant and dominated subspaces in the new factorization, and the dominated subspaces are truncated to produce a new low-rank factorization $Q_j B_j W_j^T$. This procedure is detailed in Algorithm 1.

Algorithm 1 Low-Rank Incremental SVD

```
Input: $m \times n$ matrix $A = \begin{bmatrix} A_1 & \dots & A_f \end{bmatrix}$, $A_j \in \mathbb{R}^{m \times l_j}$
 1: Compute orthogonal factorization $Q_1 B_1 = A_1$
 2: Set $W_1 = I_{l_1}$, $k_1 = l_1$ and $s_1 = l_1$
 3: for $j = 2, \dots, f$ do
 4:     Compute a rank-$(k_{j-1} + l_j)$ orthogonal factorization:
            $\hat{Q}_j \hat{B}_j = \begin{bmatrix} Q_{j-1} B_{j-1} & A_j \end{bmatrix}$
 5:     Set $\hat{W}_j = \begin{bmatrix} W_{j-1} & 0 \\ 0 & I_{l_j} \end{bmatrix}$
 6:     Set $s_j = s_{j-1} + l_j$
 7:     Choose $k_j \in (0, k_{j-1} + l_j]$ and set $d_j = k_{j-1} + l_j - k_j$
 8:     Apply the technique in Section 2.1 to construct transformations $G_u$ and $G_v$ which decouple the dom-
        inant rank-$k_j$ singular subspaces in $\hat{B}_j$ from the dominated singular subspaces
 9:     Set $\bar{B}_j = G_u^T \hat{B}_j G_v$, $\bar{Q}_j = \hat{Q}_j G_u$, and $\bar{W}_j = \hat{W}_j G_v$
10:     Truncate the last $d_j$ columns of $\bar{Q}_j$ and $\bar{W}_j$ and the last $d_j$ columns and rows of $\bar{B}_j$ to produce $Q_j$,
        $W_j$ and $B_j$
11: end for
Output: Rank-$k_f$ factorization $Q_f B_f W_f^T$ approximating the dominant SVD of $A$
```

The previous literature proposed computing the orthogonal factorization in step 6 of Algorithm 1 via a Gram-Schmidt procedure:

$$C = Q_{j-1}^T A_j$$
$$Q_\perp B_\perp = A_j - Q_{j-1} C \ .$$

This produces a new factorization

$$\left[\begin{array}{cc} Q_{j-1} B_{j-1} W_{j-1}^T & A_j \end{array} \right] = \hat{Q}_j \hat{B}_j \hat{W}_j^T, \tag{2}$$

the structure of which is shown in Figure 1.

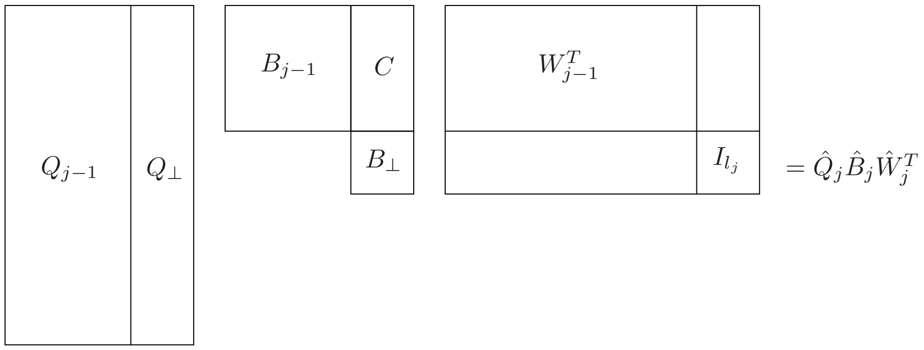

Figure 1: The structure of the update step.

Transformations G_u and G_v are constructed as in Section 2.1. These transformations are applied to put the block triangular matrix $\hat{B}$ into a block diagonal form that isolates the dominant singular subspaces from the dominated subspaces, as follows:

$$\begin{aligned} \hat{Q}_j \hat{B}_j \hat{W}_j^T &= \hat{Q}_j (G_u G_u^T) \hat{B}_j (G_v G_v^T) \hat{W}_j^T \\ &= (\hat{Q}_j G_u)(G_u^T \hat{B}_j G_v)(G_v^T \hat{W}_j^T) \\ &= \bar{Q}_j \bar{B}_j \bar{W}_j^T . \end{aligned}$$

The structure of $\bar{Q}_j \bar{B}_j \bar{W}_j^T$ is shown in Figure 2.

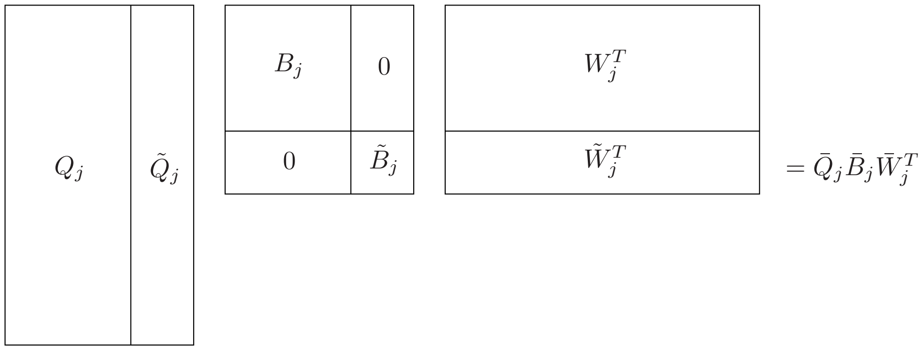

Figure 2: The result of the separation step.

The selection of k_j (line 7 in Algorithm 1) can be performed in a variety of ways. One commonly described technique maintains a constant rank at each step. Another common technique involves choosing k_j to retain all singular values of $\hat{B}_j$ satisfying some threshold (absolute or relative), this approach being constrained by the memory allocated for the factorization.

This technique produces at each step j a rank-k_j factorization $Q_j B_j W_j^T$ that best approximates the dominant SVD of $\begin{bmatrix} Q_{j-1} B_{j-1} W_{j-1}^T & A_j \end{bmatrix}$. Applying this heuristic recursively, we view $Q_j B_j W_j^T$ as an approximation to the columns of the matrix seen through step j,

$$Q_j B_j W_j^T \approx \begin{bmatrix} A_1 & \dots & A_j \end{bmatrix} .$$

The result of the algorithm is a factorization $Q_f B_f W_f^T$ that serves as an approximation for the whole of A.

The output at step j includes:

- Q_j - an approximate basis for the dominant left singular space of $A_{(1:s_j)}$,
- W_j - an approximate basis for the dominant right singular space of $A_{(1:s_j)}$, and
- B_j - a $k_j \times k_j$ matrix whose SVD contains the transformations that rotate Q_j and W_j into approximate singular vectors. The singular values of B_j are estimates for the singular values of $A_{(1:s_j)}$. These singular value estimates are necessarily non-decreasing from step $j-1$ to step j [8].

A useful result is that after each step j, there exists an orthogonal matrix embedding W_j and relating the first s_j columns of A to the current approximation and the discarded data up to this point:

$$A_{(1:s_j)} \left[\overbrace{W_j}^{k_j} \quad \overbrace{W_j^\perp}^{s_j - k_j} \right] = \left[\overbrace{Q_j B_j}^{k_j} \quad \overbrace{\tilde{Q}_1 \tilde{B}_1}^{d_1} \quad \cdots \quad \overbrace{\tilde{Q}_j \tilde{B}_j}^{d_j} \right] . \tag{3}$$

In particular, after the final step f of the algorithm, this factorization takes the form

$$A \left[W_f \quad W_f^\perp \right] = \left[Q_f B_f \quad \tilde{Q}_1 \tilde{B}_1 \quad \cdots \quad \tilde{Q}_f \tilde{B}_f \right] ,$$

yielding the following additive decomposition:

$$A = Q_f B_f W_f^T + \left[\tilde{Q}_1 \tilde{B}_1 \quad \cdots \quad \tilde{Q}_f \tilde{B}_f \right] {W_f^\perp}^T .$$

This property is proven in [8, Appendix A] and is used to construct bounds on the error of the computed factorization [10].

2.3 Implementing an Incremental SVD

The generic algorithm from the previous section leaves unspecified any structure imposed on Q_j, B_j and W_j, as well as the choice of G_u and G_v used to decouple the singular subspaces at each step. These decisions affect the overall efficiency of the method, and they constitute most of the variation in the previous work on this class of methods. This section briefly describes the previous work and summarizes the consequences of the various approaches.

In [11], Gu and Eisenstat propose a stable and fast algorithm for updating the SVD when appending a single column or row to a matrix with a known SVD. In this manner, they propose computing the SVD of A by incrementally updating the full SVD (up to the current point). The kernel step in their algorithm is the efficient tridiagonalization of a "broken arrowhead" matrix, requiring only $O(n^2)$ flops to tridiagonalize a broken arrowhead matrix of order n, compared to the $O(n^3)$ flops required for a general $n \times n$ matrix.

Manjunath and Chandrasekaran [4] propose an algorithm for tracking the dominant singular subspaces and singular values, called the *Eigenspace Update Algorithm* (EUA). Their method chooses for G_u and G_v the singular vectors of $\hat{B}_j$. The consequence of this is that the matrix $\bar{B}_j$ is a diagonal matrix whose non-zero elements are the current approximate singular values. Performing the Gram-Schmidt update (2) on a single vector from A produces a broken arrowhead matrix in $\hat{B}_j$. This allows the application of the Gu and Eisenstat approach to compute the SVD of $\hat{B}_j$ in $O(k_{j-1}^2 + l_j^2)$ and the computation of $\hat{Q}_j G_u$ and $\hat{W}_j G_v$ in $O(mk_j)$ and $O(nk_j)$ flops, respectively. Unfortunately, the overhead of this approach is such that it is only worthwhile for large values of k. Otherwise, it is more appropriate to use a dense SVD, requiring $O(mk_j^2)$ and $O(nk_j^2)$ flops to form $\hat{Q}_j G_u$ and $\hat{W}_j G_v$, respectively. Note also that the arrowhead-based method is only possible if a single column is used to update the SVD at each step. The formation of the intermediate matrices in the algorithms discussed is rich in block matrix operations whose exploitation makes efficient use of modern memory hierarchies.

In [5], Levy and Lindenbaum independently propose an approach for incrementally computing a basis for the dominant left singular subspace. Their algorithm, the *Sequential Karhunen-Loeve* (SKL), describes updating the current factorization at each step with l new columns from A. They explicitly compute the SVD of $\hat{B} = \hat{U}\hat{S}\hat{V}^T$ and choose $G_u = \hat{U}$ and $G_v = \hat{V}$. Computing the first block of $\hat{Q}G_u$ at each step requires $O(mk(k+l))$ flops. The authors suggest a value $l = \sqrt{k}/2$ for the block size, as this choice for l minimizes the overall complexity of the algorithm to approximately $12mnk$. The work of Levy and Lindenbaum focused on computing only the left dominant singular basis (the Karhunen-Loeve basis). However, for $m \gg n$, computing the right dominant singular basis does not add significant cost. Their block algorithm is rich in level 3 BLAS operations, although the naive choice of G_u and G_v results in a higher operation count than some of the following methods.

In [10], Chahlaoui, Gallivan and Van Dooren independently propose yet another algorithm for incrementally tracking dominant singular subspaces. Their algorithm approximates the left singular subspace in $8mnk$ flops. Computing approximations to the left and right singular subspaces requires $10mnk$ flops. This efficiency over the earlier algorithms is a result of a more efficient decoupling step. Their method proceeds using a URV form, where the middle matrix is maintained in a triangular form. The Gram-Schmidt expansion preserves the triangular structure, which is exploited to reduce the cost of computing $\hat{Q}G_u$. This work also presents an error analysis that addresses the effect of truncation at each step. Error bounds are derived that are essentially independent of the problem size, suggesting that the method is robust even for very large problems. Also, to quell concerns about numerical problems associated with the Gram-Schmidt procedure used in the update step, they present an error analysis that bounds the loss of orthogonality in the computed basis vectors.

In [7], Brand independently proposes an algorithm similar to that of Levy and Lindenbaum. By employing identical update and decoupling steps as those of the SKL, the algorithm has a similarly high complexity. However, the main concern of this work was the handling of missing or uncertain values in the input data; he was not concerned with the complexity of the method, aside from the reduction in cost associated with tracking a low-rank subspace. More recently, Brand [9] presents an SVD updating strategy for computing exactly the thin SVD of a matrix via a single pass through the columns. This method enjoys linear complexity when the rank of the matrix is less than the square root of the number of columns. For cases where the rank of the matrix is much larger, Brand suggests truncation in order to maintain the low computational costs of the method. This results in an algorithm which is similar to his previous work and which falls in the framework described in this paper.

In [8], Baker presented the generic separation technique described in Section 2.1 in order to unify the previous works. He presents an efficient block implementation which minimizes the computational complexity to $10mnk$. This work illustrated that the limited freedom in choosing G_u and G_v must be balanced between lowering the complexity of the method (primarily, computing $\hat{Q}G_u$) and specifying the structure of the resulting factorization. That work also illustrated the importance of block methods for exploiting a memory hierarchy.

3 Relationship to Iterative Eigensolvers

This section relates the mechanisms of Algorithm 1 to a class of optimizing eigensolvers on $A^T A$. This new analysis describes the workings of the incremental method and sets the stage for the iterative methods that follow.

Assume that we are given an orthogonal matrix D, $DD^T = D^TD = I_n$, and that we are to compute a low-rank incremental SVD of AD using Algorithm 1. Partition the matrix D according to the block updates,

$$D = \begin{bmatrix} D_1 & \cdots & D_f \end{bmatrix} ,$$

so that the algorithm is initialized with AD_1 and the factorization at step j is updated using the columns AD_j.

Note first that recurrence (3) grants us the following at each step j:

$$A \begin{bmatrix} D_1 & \cdots & D_j \end{bmatrix} W_j = Q_j B_j \ .$$

The matrix W_j approximates the right singular subspace of $A \begin{bmatrix} D_1 & \cdots & D_j \end{bmatrix}$, and the matrix $X_j \doteq \begin{bmatrix} D_1 & \cdots & D_j \end{bmatrix} W_j$ approximates the right singular subspace for A. It is easily verified that X_j has orthonormal columns of the proper dimension.

Then note the following:

$$\text{trace}\left(X_j^T A^T A X_j\right) = \text{trace}\left(B_j^T Q_j^T Q_j B_j\right) = \text{trace}\left(B_j^T B_j\right) = \sum \sigma^2(B_j) \ ,$$

where $\sigma(B_j)$ denotes the singular values of B_j. This identifies the current singular values of B_j as the Ritz values [1] of A^TA with respect to the subspace spanned by X_j. Furthermore, it can be shown that incremental algorithm performs a search at step j that maximizes the Ritz values along a "search direction" given by D_j. A proof follows.

Recall from Algorithm 1 (line 10) that the low-rank incremental SVD selects W_j as the first k_j columns of $\hat{W}_j G_v$, where $\hat{W}_j = \begin{bmatrix} W_{j-1} & 0 \\ 0 & I_{l_j} \end{bmatrix}$ is the right basis after the expansion step (line 5). Equation (1) requires that the first k_j columns of G_v are a subspace for the dominant right singular vectors of $\hat{B}_j$. Consequently, they are a global maximizer for the *Rayleigh quotient* of $\hat{B}_j^T \hat{B}_j$:

$$RayQuo(Y) \doteq \text{trace}\left(Y^T \hat{B}_j^T \hat{B}_j Y\right) \ \text{with } Y^TY = I_{k_j} \ .$$

This results from the relationship between the dominant right singular subspace of $\hat{B}_j$ and the dominant eigenspace of the symmetric matrix $\hat{B}_j^T \hat{B}_j$ (see, for example, [1]).

Note the following, recalling the necessary definitions from Section 2.2:

$$\begin{aligned} RayQuo(Y) &= \text{trace}\left(Y^T \hat{B}_j^T \hat{B}_j Y\right) \\ &= \text{trace}\left(Y^T \hat{B}_j^T \hat{Q}_j^T \hat{Q}_j \hat{B}_j Y\right) \\ &= \text{trace}\left(Y^T \begin{bmatrix} Q_{j-1}B_{j-1} & AD_j \end{bmatrix}^T \begin{bmatrix} Q_{j-1}B_{j-1} & AD_j \end{bmatrix} Y\right) \\ &= \text{trace}\left(Y^T \begin{bmatrix} AX_{j-1} & AD_j \end{bmatrix}^T \begin{bmatrix} AX_{j-1} & AD_j \end{bmatrix} Y\right) \\ &= \text{trace}\left(Y^T \begin{bmatrix} X_{j-1} & D_j \end{bmatrix}^T A^T A \begin{bmatrix} X_{j-1} & D_j \end{bmatrix} Y\right) \ . \end{aligned}$$

Then the minimizer W_j of $RayQuo(Y)$ in effect minimizes the Rayleigh quotient of A^TA subject to the span of $\begin{bmatrix} X_{j-1} & D_j \end{bmatrix}$.

The incremental algorithm can be interpreted as follows: Each step of the algorithm updates the current right basis X_j along the directions prescribed by the orthogonal matrix D, so as to maximize the trace of A^TA. For the specific choice $D = I$, described in Algorithm 1 and all previous literature, the directions take the form $D_j = \begin{bmatrix} 0 & I_{l_j} & 0 \end{bmatrix}^T$. These approaches can thus be characterized as coordinate ascent approaches for maximizing the singular values captured by the factorization. As a result, the singular values are nondecreasing from one step to the next, a fact that has been noted in previous literature. This further implies that if the dominant singular subspaces are discovered by the algorithm, then the subspace will not be discarded. Furthermore, this analysis suggests that the method can be modified to compute the singular subspaces associated with the *smallest* singular values, though this is not considered in this paper.

4 A Multipass Incremental SVD

The previous discussion interpreted the low-rank incremental SVD of AD, where D was an orthogonal matrix. This section proposes some choices for matrices D that allow the low-rank incremental algorithm to exploit multiple passes through A, assuming the availability of A allows this.

Assume we have a rank-k orthonormal basis X_0 whose span approximates the right dominant singular subspace of A. Consider an orthogonal matrix $D = \begin{bmatrix} X_0 & X_\perp \end{bmatrix}$. Then a rank-$k$ incremental SVD of AD will initially produce a factorization with a right basis equivalent X_0. The algorithm will process the rest of the directional information in D, as discussed in the previous section. In this way, we can describe an algorithm which makes multiple passes through A, initializing each new pass with the approximation computed by previous pass, via an appropriate choice of D. Because Algorithm 1 is an ascent method, each successive factorization approximates A at least as well as the preceding factorization. Algorithm 2 details this approach.

Algorithm 2 Coordinate Ascent Multipass Incremental SVD.

```
Input: Rank-$k$ orthonormal basis $X_0$ approximating dominant right singular subspace.
1: for $j = 1, 2, \ldots$ until $Q_{j-1}$, $B_{j-1}$, $X_{j-1}$ satisfy some convergence criterion do
2:     Compute orthogonal matrix $D$
           $$D = \begin{bmatrix} X_{j-1} & D_2 & \ldots & D_f \end{bmatrix} \quad (4)$$
3:     Compute rank-$k$ factorization $Q_j B_j W_j^T$ of $AD$ via Algorithm 1.
4:     Set $X_j = DW_j$
5: end for
```

The analysis in Section 3 showed that if X_0 is a basis for the dominant right singular subspace of A, then each X_j is also a basis for the dominant right singular subspaces. Section 3 explained that the columns of D act as prescribed search directions in the optimization for the dominant SVD of A. Algorithm 2 specified only the first k directions, in order to initialize the search with the output of the previous iteration. It is possible that specifying additional columns of D might improve the convergence of the algorithm. It is common in optimization methods to exploit gradient information to increase the efficiency of a search. Algorithm 2 was shown in Section 3 to implement a maximization of the Rayleigh quotient of $A^T A$ over the set of orthonormal bases (the *compact Stiefel manifold*). The gradient of the Rayleigh quotient on this manifold has been described in numerous places in the literature (see [12] and references there-in):

$$\text{grad } RayQuo(X) = (I - XX^T)A^T AX \ .$$

In constructing D, we can also insert an orthonormal basis G for the component of grad $RayQuo(X)$ orthogonal to the current iterate X. The resulting incremental SVD of AD will therefore be initialized with X_0 and will immediately search in the gradient direction. This effectively incorporates a steepest ascent search into the incremental SVD. This technique is detailed in Algorithm 3.

Algorithm 3 Steepest Ascent Multipass Incremental SVD.

```
Input: Rank-$k$ orthonormal basis $X_0$ approximating the right dominant singular subspace.
1: for $j = 1, 2, \ldots$ until $Q_{j-1}$, $B_{j-1}$, $X_{j-1}$ satisfy some convergence criterion do
2:     Compute orthonormal basis $G_{j-1}$ for colspan $\left(A^T A X_{j-1}\right)$, s.t. $G_{j-1}^T X_{j-1} = 0$
3:     Compute orthogonal matrix $D$
           $$D = \begin{bmatrix} X_{j-1} & G_{j-1} & D_3 & \ldots & D_f \end{bmatrix} \quad (5)$$
4:     Compute rank-$k$ factorization $Q_j B_j W_j^T$ of $AD$ via Algorithm 1.
5:     Set $X_j = DW_j$
6: end for
```

Because Algorithm 3 is related to steepest descent, the asymptotic rate of convergence is expected to be linear. Algorithm 2 does not employ gradient information, so the convergence should be slower. However, the asymptotic rate is still expected to be linear. Global convergence proofs for these methods, along with detailed rates of convergence, are upcoming. However, the numerical experiments in the next section confirm this intuition.

It should also be noted that both proposed multipass algorithms can be implemented in a memory efficient manner, incrementally passing through the columns of A. The Coordinate Ascent method requires two incremental passes through A to conduct an Incremental SVD of AD, while the Steepest Ascent method requires three passes through A to conduct an Incremental SVD of AD. This extra pass through A is necessitated by the production of the gradient used in D.

5 Numerical Performance

We conducted some experiments on the multipass algorithms described in Section 4. The purpose of this testing is to illustrate the convergence of the two proposed multipass methods. These tests were performed in MATLAB. The matrix A used was a small real 5000×100 matrix generated via the MATLAB command `randn`. The SVD of A was computed, the smaller 95 singular vales were scaled by .95 to introduce a small gap, and the multipass methods were allowed 49 passes through the matrix to compute the rank-5 dominant SVD. Figure 3 shows the results of these experiments.

Both multipass techniques demonstrate the anticipated linear convergence rate. Both methods converge to the solution, as measured by the error in the singular values. The steepest ascent method (Algorithm 3) makes more progress on each Incremental SVD of AD due the incorporation of gradient information. However, recall that forming the gradient requires an additional pass through A. As a result, the coordinate ascent method (Algorithm 2) makes more progress with respect to the number of passes through A. This suggests that different approaches may be preferable depending on such factors as the latency involved in retrieving the columns of the data matrix.

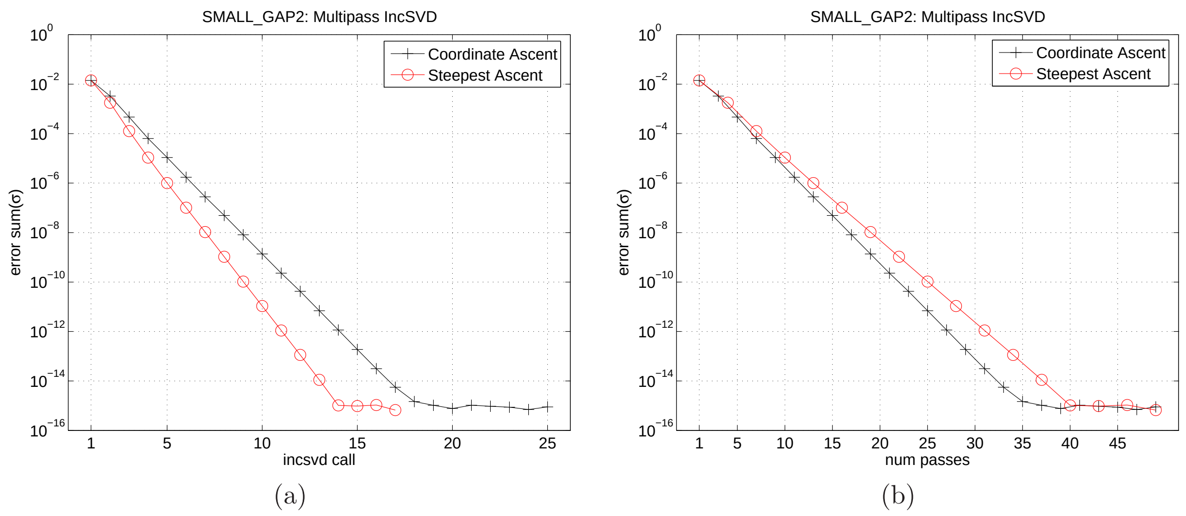

Figure 3: Convergence of singular values for Algorithms 2 and 3, plotted against (a) number of IncSVD calls on AD and (b) number of passes through A.

6 Concluding Remarks

We have presented a low-rank incremental method for computing dominant singular subspaces of a matrix. The presentation was general enough to unify similar methods previously described in the literature. We conducted a novel analysis of this class of methods, showing ties to an iterative eigensolver approach and providing some additional insight into the operation of the methods. This analysis also paved the way for the description of a framework for multipass incremental SVD approaches, which were demonstrated via numerical experiments.

The Incremental SVD methods were designed for the scenario where only a single pass through the matrix A is available, and such a scenario does not admit any other computational options to our knowledge. Though this setting clearly does not admit the application of the multipass methods developed here, the multipass methods still enjoy low memory usage and unique memory access patterns. Further research is necessary to identify cases where these methods may be preferred over other methods, e.g., iterative eigenvalue computations on $A^T A$.

Acknowledgments Useful discussions with Pierre-Antoine Absil, Petros Drineas, Danny Sorensen and Heidi Thornquist are gratefully acknowledged.

References

[1] G. Golub and C. Van Loan. *Matrix Computations.* The Johns Hopkins University Press, Baltimore, third edition, 1996.

[2] P. Businger. Updating a singular value decomposition. *BIT*, 10(3):376–385, 1970.

[3] James R. Bunch and Christopher P. Nielsen. Updating the singular value decomposition. *Numerische Mathematik*, 31(2):111–129, 1978.

[4] B. S. Manjunath, S. Chandrasekaran, and Y. F. Wang. An eigenspace update algorithm for image analysis. In *IEEE Symposium on Computer Vision*, page 10B Object Recognition III, 1995.

[5] A. Levy and M. Lindenbaum. Sequential Karhunen-Loeve basis extraction and its application to images. *IEEE Transactions on image processing*, 9(8):1371–1374, August 2000.

[6] Y. Chahlaoui, K. Gallivan, and P. Van Dooren. An incremental method for computing dominant singular spaces. In *Computational Information Retrieval*, pages 53–62. SIAM, 2001.

[7] M. Brand. Incremental singular value decomposition of uncertain data with missing values. In *Proceedings of the 2002 European Conference on Computer Vision*, 2002.

[8] C. G. Baker. A block incremental algorithm for computing dominant singular subspaces. Masters Thesis TR-041112, Department of Computer Science, Florida State University, 2004.

[9] Matthew Brand. Fast low-rank modifications of the thin singular value decomposition. *Linear Algebra and its Applications*, 415(1):20–30, May 2006.

[10] Y. Chahlaoui, K. Gallivan, and P. Van Dooren. Recursive calculation of dominant singular subspaces. *SIAM J. Matrix Anal. Appl.*, 25(2):445–463, 2003.

[11] M. Gu and S. C. Eisenstat. A stable and fast algorithm for updating the singular value decomposition. Technical Report YALEU/DCS/RR-966, Yale University, New Haven, CT, 1993.

[12] P.-A Absil, C. G. Baker, and K. A. Gallivan. Trust-region methods on Riemannian manifolds. *Foundations of Computational Mathematics*, 7(3):303–330, July 2007.