Low-Rank Incremental Methods for Computing Dominant Singular Subspaces[☆]

C.G. Baker[a], K.A. Gallivan[b], P. Van Dooren[c]

[a] *Computational Engineering and Energy Studies, Oak Ridge National Laboratory, PO Box 2008 MS6164, Oak Ridge TN 37831-6164*
[b] *Department of Mathematics, 208 Love Building, 1017 Academic Way, Florida State University, Tallahassee FL 32306-4510*
[c] *CESAME, Université catholique de Louvain, Av. Georges Lemaître 4, B-1348 Louvain-la-Neuve, Belgium*

Abstract

Computing the singular values and vectors of a matrix is a crucial kernel in numerous scientific and industrial applications. As such, numerous methods have been proposed to handle this problem in a computationally efficient way. This paper considers a family of methods for incrementally computing the dominant SVD of a large matrix A. Specifically, we describe a unification of a number of previously independent methods for approximating the dominant SVD via a single pass through A. We tie the behavior of these methods to that of a class of optimization-based iterative eigensolvers on A^TA. An iterative procedure is proposed which allows the computation of an accurate dominant SVD via multiple passes through A. We present an analysis of the convergence of this iteration and provide empirical demonstration of the proposed method on both synthetic and benchmark data.

Keywords: singular value decomposition, incremental SVD, iterative methods, pass-efficient linear algebra, convergence analysis

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 ordered non-decreasing. The σ_i are the *singular values* of A, and the columns of U and V are the *left and right singular vectors* of A, respectively. Often times, the SVD is abbreviated to ignore the right-most columns of U corresponding to the zero matrix below Σ. This is referred to in the literature as a *thin SVD* [1] or a *singular value factorization* [2]. The thin SVD is written as $A = U\Sigma V^T$, where U now denotes an $m \times n$ matrix with orthonormal columns, and Σ and V are the same as above.

The components of the SVD are optimal in many respects [1, 2], and these properties have resulted in the use of the SVD in many applications. One commonly used technique for dimensionality reduction of a large data set is *Principal Component Analysis* (PCA). Given a set of random variables, the goal of PCA is to determine a coordinate system such that the variances of any projection of the data set lie on the

[☆]The submitted manuscript has been authored by a contractor of the U.S. Government under Contract No. DE-AC05-00OR22725. Accordingly, the U.S. Government retains a non-exclusive, royalty-free license to publish or reproduce the published form of this contribution, or allow others to do so, for U.S. Government purposes.

Email addresses: `bakercg@ornl.gov` (C.G. Baker), `gallivan@math.fsu.edu` (K.A. Gallivan), `paul.vandooren@uclouvain.be` (P. Van Dooren)

coordinate axes. The method proceeds by neglecting coordinates which do not correspond to large variance. PCA identifies the principal components as the *dominant left singular vectors*, those left singular vectors associated with the largest singular values. This technique has been widely applied to problems in computer vision where expensive analysis benefits from a reduction in the size of the data, e.g. face and handwriting recognition [4, 5, 6].

Another related application of the SVD is that of the *Proper Orthogonal Decomposition* (POD). POD seeks to produce an orthonormal basis which captures the dominant behavior of a large-scale dynamical system based on observations of the system's state over time. Known also as the *Empirical Eigenfunction Decomposition* [7], this technique is motivated by interpreting the matrix A as a time series of discrete approximations to a function on a spatial domain. Sirovich [8] introduced the methods of snapshots to produce this basis. Given a dynamical system, the method of snapshots saves instantaneous solutions of the system (*the snapshots*) produced via a direct numerical simulation. The snapshots may be spaced across time and/or system parameters. The SVD of these snapshots then provides an orthonormal basis that approximates the eigenfunctions of the system. This basis can be employed for a number of purposes: to compress the snapshots, to project the snapshots to a lower dimension (where expensive interpolation may be more feasibly approached), or to use a Galerkin projection technique to produce a reduced-order model of the system [9, 10].

A common trait among these applications is the size of the data. For the computer vision cases, the matrix A contains a column for each image, with the images often being very large. In the case of the POD, each column of A represents a snapshot of the system degrees of freedom. These applications usually lead to a matrix that has many more rows than columns. It is matrices of this type 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 largest $k \ll n$ singular triplets or rank-k *dominant singular subspaces*, i.e., the subspaces associated with the dominant singular vectors.

One drawback of the SVD is the cost of its computation. Bidiagonalizing A and computing the SVD directly requires $14mn^2 + O(n^3)$ flops. This approach computes all n singular triplets. Computing the thin QR factorization $A = QR$ and the SVD of R produces all singular triplets of A in $6mn^2 + O(n^3)$ flops. This approach is more economical when $m \gg n$, and it allows any subset of k singular triplets to be formed at a total cost of $4mn^2 + 2mnk + O(n^3)$ flops. Alternatively, the SVD can be computed via the eigendecomposition of $A^TA = V\Sigma^2V^T$. This method requires mn^2 flops to form A^TA and $2mnk + O(n^3)$ to compute the first k columns of $U = AV^T\Sigma^{-1}$. By instead using an iterative eigensolver such as ARPACK [11] to compute only the largest eigenvectors, A^TA is repeatedly applied but never formed. Both approaches require the ability to apply A^T, which may add difficulty in applications where A is accessible only as a matrix-free linear operator. With the exception of iterative eigensolvers on A^TA, each of these methods requires $O(mn^2)$ floating point operations and $O(mn)$ storage.

These methods are referred to as *batch methods* because they require that all of A is available to perform the SVD. In some scenarios the columns of A will be produced incrementally, such as when producing snapshots for a POD-based method. It may be advantageous to perform the computation as the columns of A become available, instead of waiting until all columns of A are available before doing any computation. In other scenarios, the SVD of a matrix must be updated by appending some number of columns. This is typical when performing PCA on a growing database. Applications with this property are common, and include document retrieval, active recognition, and signal processing [12].

These characteristics on the availability of A have given rise to a class of *incremental methods*. Given the SVD of a matrix $A = U\Sigma V^T$, the goal of incremental methods is to compute the SVD of the related matrix $A_+ = [A\ P]$. Incremental (or *recursive*) methods are thus named because they update the current SVD using the new columns, instead of computing the updated SVD *ab initio*. These methods strive to update the SVD in a manner which is more efficient than the $O(mn^2)$ algorithmic complexity which would otherwise be incurred at each step by naïvely using a batch method to compute the SVD of $[A\ P]$. The cumulative cost over all columns of the matrix may be significantly higher than that of the batch methods—typically $O(mn^3)$ as compared to $O(mn^2)$. However, this additional cost can be justified by the availability of intermediate singular value decompositions, as well as the potential to amortize the SVD updates during the production of the columns of A.

Just as with the batch methods, the classical incremental methods produce a full SVD of A. However, for many of the motivating applications, only the dominant singular vectors and values of A are needed. Furthermore, for sufficiently large matrices, even the thin SVD of A–requiring $O(mn)$ memory–may be too large and its computational cost too high. An extreme memory hierarchy may favor only an incremental access to the columns of A, while penalizing or prohibiting writes to distant memory on a disk, a network, or in read-only storage.

These constraints, coupled with the need to compute only the dominant singular vectors and values of A, prompted the formulation of a class of low-rank, incremental algorithms for approximating the dominant SVD of the matrix A [13, 14, 15, 16, 17, 18, 19, 21]. These methods track a low-rank representation of A based on the SVD. As a new group of columns of A becomes available, this low-rank representation is updated, similar to traditional incremental SVD methods. However, the defining characteristic of these methods is that the resulting factorization is then reduced to the desired rank by truncating information corresponding to smaller singular values. In this manner, the dominant singular subspaces of the matrix A are tracked in an incremental fashion. A consequence of this truncation is that these methods generally produce an approximation of the dominant SVD. The benefits come from maintaining a low-rank factorization, reducing the computational requirement to $O(mk + nk)$ memory and $O(mnk)$ floating point operations.

This paper describes the family of low-rank incremental methods for computing dominant singular subspaces. We review previous approaches and propose a generic algorithm which unifies these methods. We relate this algorithm to a class of iterative methods for approximating the eigenvalues of $A^T A$. We describe an iterative approach exploiting multiple passes through A, in order to improve the accuracy of the computed factorization, and we analyze the convergence properties of this multipass approach. Lastly, we provide empirical results demonstrating the potential of the multipass approach and validating the convergence analysis.

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. The technique is flexible in that it can be tailored to the requirements of the application, e.g., those requiring only a left space basis, both left and right space bases, explicit singular vectors, etc. Such variants are discussed, and their efficiency is characterized according to their floating point operation counts.

This algorithm makes a single pass through the columns of A, updating a running estimate of the dominant SVD with each new block of columns. The estimate is updated by computing the dominant SVD of the current estimate and the incoming columns. The heart of the method is a straight-forward technique for efficiently isolating the dominant and subordinate subspaces, in order to truncate the latter. By only preserving the dominant SVD from step to step, the storage requirement and computational cost of the algorithm are minimized. The trade-off is that the ultimate estimate (in general) doesn't exactly match the dominant SVD of A, due to the information truncated at each step.

Section 2.1 introduces the subspace separation technique at the heart of the incremental methods discussed in this paper. Section 2.2 describes a generic low-rank incremental SVD based on this approach. Section 2.3 shows how the low-rank incremental methods in the literature fit into this framework and discusses the trade-offs in their implementations.

2.1. A Generic Separation Technique

Given an $m \times (k + l)$ matrix M, $m \gg k + l$, and its QR factorization,

$$M = \left[\overbrace{Q_1}^{k+l} \quad \overbrace{Q_2}^{m-k-l} \right] \begin{bmatrix} R \\ 0 \end{bmatrix} = Q_1 R \; ,$$

consider the SVD of R and partition it conformally as

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

where U_2, Σ_2, and V_2 contain the smallest l left singular vectors, singular values and right singular vectors of R, respectively. Define orthogonal transformations G_u and G_v such that they block diagonalize the singular vectors of R, like so:

$$G_u^T U = \begin{bmatrix} T_u & 0 \\ 0 & S_u \end{bmatrix} \quad \text{and} \quad G_v^T V = \begin{bmatrix} T_v & 0 \\ 0 & S_v \end{bmatrix} , \tag{1}$$

where T_u and T_v are $k \times k$. Apply these transformations to R to yield $R_{new} = G_u^T R G_v$. Then G_u and G_v rotate R to a coordinate system where its left and right singular bases are block diagonal. It follows that R_{new} has the form

$$R_{new} = G_u^T R G_v = \begin{bmatrix} T_u \Sigma_1 T_v^T & 0 \\ 0 & S_u \Sigma_2 S_v^T \end{bmatrix} . \tag{2}$$

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

$$\begin{aligned} M &= Q_1 R \\ &= (Q_1 G_u)(G_u^T R G_v) G_v^T \\ &\doteq Q_{new} R_{new} G_v^T \\ &= Q_{new} \begin{bmatrix} T_u \Sigma_1 T_v^T & 0 \\ 0 & S_u \Sigma_2 S_v^T \end{bmatrix} G_v^T , \end{aligned}$$

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, respectively.

It should be noted that G_u is not uniquely defined by Equation (1). This criterion admits any G_u whose first k columns are some orthonormal basis for the dominant left singular subspace of R, and whose last l columns therefore are some orthonormal basis for the subordinate (dominated) left singular subspace of R. This is also the case, *mutatis mutandis*, for G_v.

2.2. An Incremental Method

The technique of the previous section can be used to define a generic method that requires only one pass through the columns of an $m \times n$ matrix A to compute approximate bases for the left and right dominant singular subspaces. 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 subordinate subspaces in the new factorization, allowing the subordinate subspaces to be truncated to produce a new low-rank factorization $Q_j B_j W_j^T$. This procedure is detailed in Algorithm 1.

The previous literature proposed computing the orthogonal factorization in line 4 of Algorithm 1 via a Gram-Schmidt procedure:

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

This produces a new factorization

$$\begin{bmatrix} Q_{j-1} B_{j-1} W_{j-1}^T & A_j \end{bmatrix} = \hat{Q}_j \hat{B}_j \hat{W}_j^T , \tag{3}$$

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 $W_1 = I_{l_1}$, rank $k_1 = l_1$, width $s_1 = l_1$
 3: for $j = 2, \dots, f$ do
          — Expand $Q_{j-1} B_{j-1} W_{j-1}^T$ with $A_j$ —
 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 $s_j = s_{j-1} + l_j$
          — Decouple subspaces and truncate —
 7:    Choose $k_j \in (0, k_{j-1} + l_j]$, 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
       dominant rank-$k_j$ singular subspaces in $\hat{B}_j$
 9:    $\bar{B}_j = G_u^T \hat{B}_j G_v$
10:    $\bar{Q}_j = \hat{Q}_j G_u$
11:    $\bar{W}_j = \hat{W}_j G_v$
12:    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 $B_j$, respectively
13: end for
Cost: $O(mnk)$ flops, one pass through $A$, $O(mk + nk)$ storage
Output: Rank-$k_f$ $Q_f B_f W_f^T$ approximating the dominant SVD of $A$
```

the structure of which is shown in Figure 1.

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 subordinate 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. Note, in steps 9-11 of Algorithm 1, it is not necessary to compute the columns and rows of $\bar{Q}_j$, $\bar{B}_j$, and $\bar{W}_j$ that are to be truncated.

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 size of the memory allocated for the factorization [13, 15, 18].

At each step j, this technique produces the rank-k_j factorization $Q_j B_j W_j^T$ that optimally approximates (in a 2-norm sense) the matrix $\begin{bmatrix} Q_{j-1} B_{j-1} W_{j-1}^T & A_j \end{bmatrix}$. Applying this heuristic inductively, the algorithm computes a final factorization $Q_f B_f W_f^T$ that seeks to similarly approximate 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

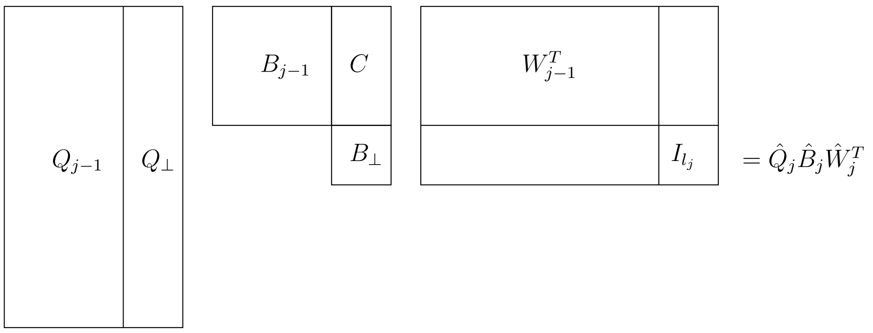

Figure 1: The structure of a Gram-Schmidt expansion step (lines 4-5 of Algorithm 1).

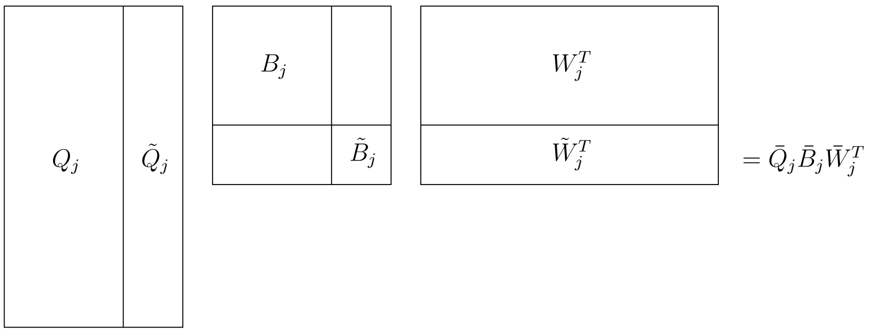

Figure 2: The result of the subspace separation step (lines 9-11 of Algorithm 1).

- 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 [18].

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:

$$A_{(1:s_j)} \left[\begin{array}{cc} \overbrace{W_j}^{k_j} & \overbrace{W_j^\perp}^{s_j-k_j} \end{array} \right] = \left[\begin{array}{cccc} \overbrace{Q_j B_j}^{k_j} & \overbrace{\tilde{Q}_1 \tilde{B}_1}^{d_1} & \cdots & \overbrace{\tilde{Q}_j \tilde{B}_j}^{d_j} \end{array} \right]. \tag{4}$$

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

$$A \left[\begin{array}{cc} W_f & W_f^\perp \end{array} \right] = \left[\begin{array}{cccc} Q_f B_f & \tilde{Q}_1 \tilde{B}_1 & \cdots & \tilde{Q}_f \tilde{B}_f \end{array} \right],$$

yielding the following additive decomposition:

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

This property is proven in [18, Appendix A] and is used to construct bounds on the error of the computed factorization [16].

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 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 [20], 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. Their algorithm is capable of computing the SVD of $\hat{B}_j$ in $O(j^2)$ computations instead of the $O(j^3)$ computations required for a dense SVD. They propose using this method as the foundation for an efficient batch incremental method.

Chandrasekaran *et al.* [13, 14] propose an algorithm for tracking the dominant singular subspace 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 $\hat{B}_j$ is a diagonal matrix whose nonzero elements are the current approximate singular values. Performing the Gram-Schmidt update (3) 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 [20] to compute the SVD of $\hat{B}_j$ in $O(k^2)$ and enable the computation of $\hat{Q}_j G_u$ and $\hat{W}_j G_v$ in $O(mk)$ and $O(nk)$ flops, respectively. However, the overhead of this approach is such that it is only worthwhile for extremely large values of k. Otherwise, it is more appropriate to use a classical dense SVD, requiring $O(mk^2)$ and $O(nk^2)$ flops to form $\hat{Q}_j G_u$ and $\hat{W}_j G_v$, respectively. The latter leads to an overall complexity of $O(mnk^2)$ to process all columns of A. 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 [15], Levy and Lindenbaum independently propose an approach for incrementally computing a basis for the dominant left singular subspace. Their algorithm, the *Sequential Karhunen-Loève* (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 dominant left singular basis (the Karhunen-Loève basis). However, for $m \gg n$, computing the dominant right singular basis does not add significant cost. Their block algorithm is rich in level 3 BLAS operations, although the naïve choice of G_u and G_v results in a higher operation count than some of the following methods.

In [17], 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 similar computational complexity. The main contributions of [17] are techniques for handling missing or uncertain values in the input data; Brand is not concerned with the complexity of the method, aside from the principle reduction in cost associated with tracking a low-rank subspace. A more recent work [19] is concerned with efficient methods for handling a variety of low-rank updates to a matrix, including the column append described in the incremental SVD. Algorithm 1 can similarly be easily generalized to include other low-rank updates; here we focus solely only the addition of new columns. In [19], Brand proposes a low-rank incremental SVD algorithm that achieves a linear $O(mnk)$ complexity by caching the rotations G_u into a small $k \times k$ matrix instead of accumulating them into the basis Q (and similarly for W). However, this approach assumes that either the incoming columns of A do not bring new subspace information (relative to Q), or that this information is truncated *before* being included into the current SVD. Otherwise, the truncation to low rank requires absorbing the cached rotations into Q and eliminates most of the efficiency gains.

In [21], Chahlaoui, Gallivan and Van Dooren independently propose yet another algorithm for incrementally tracking dominant singular subspaces. Their algorithm approximates the left singular subspace in a linear $8mnk + O(nk^3)$ flops. They describe a related algorithm which also computes the right singular subspace, requiring $10mnk$ flops. The efficiency gains over previous approaches are a result of a more efficient decoupling step, although there approach is described only for single vector updates. Their method proceeds using a URV form, where the structure of the middle matrix B_j is relaxed from diagonal to upper triangular. 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. Furthermore, 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. These bounds are essentially independent of the problem size, suggesting that the method is robust even for very large problems.

In [18], Baker presents the generic separation technique described in Section 2.1. This description allows for the unification of the previous methods. He presents an efficient block implementation which minimizes the computational complexity, to approximately $10mnk$ for arbitrary update size. This work illustrated that the limited freedom in choosing G_u and G_v must be balanced between lowering the complexity of the method (i.e., computing $\hat{Q}G_u$) and specifying the structure of the resulting factorization (diagonal versus triangular versus unstructured B). The work illustrates numerous approaches for achieving a linear complexity, a goal achievable by Levy *et al.* and Chahlaoui *et al.* only via constraints on the update size and by Brand only for low-rank A matrices.

3. Relationship to Iterative Eigensolvers

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

Given an orthogonal matrix D, $DD^T = D^TD = I_n$, consider the application of Algorithm 1 to the matrix AD. Partition the matrix D according to the block updates:

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

The algorithm is initialized with AD_1, and the factorization at step j is updated with the columns AD_j.

First note that recurrence (4) 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 $A \begin{bmatrix} D_1 & \cdots & D_j \end{bmatrix}$, so that the matrix $V_j \doteq \begin{bmatrix} D_1 & \cdots & D_j \end{bmatrix} W_j$ approximates the right singular subspace of A. It is easily verified that V_j has orthonormal columns of an appropriate dimension.

Next note the following:

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

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 V_j. We will show that Algorithm 1 performs a search at step j that maximizes the Ritz values along the "search direction" given by D_j. An outline of proof follows.

Recall from Algorithm 1 (line 12) 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 orthogonal factor after the expansion step (line 5). Equation (1) requires that the first k_j columns of G_v are a basis for the dominant right singular subspace of $\hat{B}_j$. Consequently, they are a global maximizer of Rayleigh quotient of $\hat{B}_j^T \hat{B}_j$:

$$\operatorname{RQ}(Y) \doteq \operatorname{trace}\left(Y^T \hat{B}_j^T \hat{B}_j Y\right) , \quad \text{for } Y^TY = I . \tag{5}$$

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 Equation (4) and the necessary definitions from Section 2.2:

$$\begin{aligned}
\mathrm{RQ}\left(Y\right) &= \operatorname{trace}\left(Y^T \hat{B}_j^T \hat{B}_j Y\right) \\
&= \operatorname{trace}\left(Y^T \hat{B}_j^T \hat{Q}_j^T \hat{Q}_j \hat{B}_j Y\right) \\
&= \operatorname{trace}\left(Y^T \begin{bmatrix} Q_{j-1}B_{j-1} & AD_j \end{bmatrix}^T \begin{bmatrix} Q_{j-1}B_{j-1} & AD_j \end{bmatrix} Y\right) \\
&= \operatorname{trace}\left(Y^T \begin{bmatrix} AV_{j-1} & AD_j \end{bmatrix}^T \begin{bmatrix} AV_{j-1} & AD_j \end{bmatrix} Y\right) \\
&= \operatorname{trace}\left(Y^T \begin{bmatrix} V_{j-1} & D_j \end{bmatrix}^T A^T A \begin{bmatrix} V_{j-1} & D_j \end{bmatrix} Y\right) \ .
\end{aligned}$$

Then the maximizer W_j of $\mathrm{RQ}\left(\cdot\right)$ also maximizes the Rayleigh quotient of $A^T A$ subject to the span of $\begin{bmatrix} V_{j-1} & D_j \end{bmatrix}$.

The incremental algorithm can be interpreted as follows. Each step of the algorithm updates the current right basis V_j along the directions prescribed by the orthogonal matrix D, so as to maximize the trace of $A^T A$. 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. The singular values are obviously non-decreasing from one step to the next, a fact that has been noted in previous literature (in particular, [16, 18]). This further implies that if the dominant singular subspace is discovered by the algorithm, then the subspace will not be discarded. There are two novel results that follow from this interpretation. First, this analysis suggests that the method can easily be modified to compute the singular subspaces associated with the smallest singular values; this notion is left for future investigation. Second, the search directions D_j offer an opportunity to influence the outcome of the algorithm.

4. A Family of Multipass IncSVD Methods

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

4.1. Multipass Approaches

Assume we have a rank-k orthonormal basis W_0. Consider an orthogonal matrix $D = \begin{bmatrix} W_0 & W_\perp \end{bmatrix}$. It is straightforward to show that a rank-k Incremental SVD of AD will initially produce a factorization whose right basis spans $\operatorname{colspan}\left(W_0\right)$. The algorithm will continue onward to process the rest of the directional information in D, as discussed in the previous section. In this way, we can describe an algorithm that makes multiple passes through A, initializing each new pass with the approximation computed by previous pass. 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 Multipass Incremental SVD.

```
Input: Rank-$k$ orthonormal basis $V_0$.
1: for $i = 1, 2, \ldots$ until $Q_{i-1}$, $B_{i-1}$, $V_{i-1}$ satisfy some convergence criterion do
2:     Compute orthogonal matrix $D$
           $D = \begin{bmatrix} V_{i-1} & D_2 & \ldots & D_f \end{bmatrix}$          (6)
3:     Compute rank-$k$ factorization $Q_i B_i W_i^T$ of $AD$ via Algorithm 1.
4:     Set $V_i = DW_i$
5: end for
Cost: $O(mnk)$ flops and 2 passes through $A$ per iteration, $O(mk + nk)$ storage
```

The analysis in Section 3 showed that if the initial iterate W_0 in Algorithm 1 is a basis for the dominant right singular subspace of AD, then each W_j is also a basis for the dominant right singular subspace, and Section 4.2 shows that Algorithm 2 is convergent to the dominant SVD of A. 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. The Incremental SVD was shown in Section 3 to implement a maximization of the Rayleigh quotient of A^TA 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 [22, 23] and references there-in):

$$\text{grad RQ}\,(V) = (I - VV^T)A^TAV \ .$$

In constructing D, we desire an orthonormal basis G for the component of the gradient orthogonal to the current iterate V. For such a D, the Incremental SVD of AD will be initialized with V_0 and immediately search in gradient-related directions. This effectively incorporates a steepest ascent search into the Incremental SVD. This technique is detailed in Algorithm 3.

Algorithm 3 Gradient-Accelerated Multipass Incremental SVD.

```
Input: Rank-$k$ orthonormal basis $V_0$.
1: for $i = 1, 2, \ldots$ until $Q_{i-1}$, $B_{i-1}$, $V_{i-1}$ satisfy some convergence criterion do
2:     Compute orthonormal basis $G$ for colspan$\left(A^TAV_{i-1}\right)$, s.t. $G^TV_{i-1} = 0$
3:     Compute orthogonal matrix $D$
           $D = \begin{bmatrix} V_{i-1} & G & D_3 & \ldots & D_f \end{bmatrix}$        (7)
4:     Compute rank-$k$ factorization $Q_iB_iW_i^T$ of $AD$ via Algorithm 1.
5:     Set $V_i = DW_i$
6: end for
Cost: $O(mnk)$ flops and 3 passes through $A$ per iteration, $O(mk + nk)$ storage
```

Remark 1. The explicit inclusion of gradient information into the search directions and the known convergence properties of steepest ascent imply that an iterative approach based on Algorithm 3, but using a truncated matrix D, would still converge. Such an approach would modify the ratio of floating point operations per A-access per iteration, potentially favoring a scenario where the latency involved in accessing A was lower. This approach is currently under investigation.

Astute readers may be concerned about the computational costs associated with forming AD in Algorithm 2 and Algorithm 3. An orthogonal matrix D, explicitly formed, requires $O(n^2)$ storage and $O(n^3)$ flops; computing AD via direct multiplication requires $O(mn^2)$ flops. Fortunately, an efficient method exists for specifying D and incrementally computing the columns of AD.

Note that Algorithms 2 and 3 specify only the first k and $2k$ columns of D, respectively. Consider the case of Algorithm 2. Let the matrix $\tilde{D}$ consist of those first k columns of D specified by the algorithm: $\tilde{D} = V$. We can compute a Householder QR factorization of $\tilde{D}$ taking the following form:

$$\tilde{D} = H_1 \cdots H_k \begin{bmatrix} I_k \\ 0 \end{bmatrix} \ .$$

Note that the matrix $D \doteq H_1 \cdots H_k$ is an orthogonal matrix satisfying the requirements of Algorithm 2. Furthermore, because D is the product of k Householder reflectors, it can be represented using a rank-k factorization [24, 25] as follows:

$$D = H_1 \cdots H_k \doteq I - YZ^T, \quad Y, Z \in \mathbb{R}^{n \times k} \ .$$

Then computing a block of columns of AD_j as needed by Algorithm 1 is eased noting the following:

$$AD_j = AD \begin{bmatrix} 0 & I & 0 \end{bmatrix}^T = (A - AYZ^T) \begin{bmatrix} 0 & I & 0 \end{bmatrix}^T = A_j - (AY)Z_j^T \ ,$$

where $Z^T = \begin{bmatrix} Z_1^T & \cdots & Z_f^T \end{bmatrix}$. In the case of Algorithm 2, AY can be computed in advance, requiring only $O(mnk)$ flops, $O(mk)$ storage and one pass through A. This development also holds true for Algorithm 3, except that D must be represented via a rank-$2k$ factorization due to the extra constraints on its content. Additionally, the gradient must be computed. Because AV_{j-1} is equal to $Q_{j-1}B_{j-1}$, the term A^TAV_{j-1} can be computed via a single multiplication A^TQ_{j-1}, requiring an additional $O(mnk)$ flops and one pass through A.

4.2. Convergence Properties of the Multipass IncSVD

One pass of the Multipass IncSVD (Algorithm 2) consists of $f-1$ SVD computations of matrices of dimension $(k+l) \times (k+l)$, corresponding to the $f-1$ iterations of Algorithm 1. In order to analyze the convergence of the algorithm, we consider one of these SVD computations. At each step of the pass, one can partition the matrix $(AD)\begin{bmatrix} W & W^\perp \end{bmatrix}$ as follows, where the first block row has k rows and the second block row has l rows (we assume for illustration that $f = 5$):

$$(AD)\begin{bmatrix} W & W^\perp \end{bmatrix} = \begin{bmatrix} Q & Q_\perp \end{bmatrix} M, \quad \text{where} \quad M \doteq \begin{bmatrix} B & A_{12} & A_{13} & A_{14} & A_{15} \\ & A_{22} & A_{23} & A_{24} & A_{25} \\ & A_{32} & A_{33} & A_{34} & A_{35} \end{bmatrix}.$$

If we assume that we just performed the reduction corresponding to block column 3 then M has the block pattern

$$M = \begin{bmatrix} B & A_{12} & 0 & A_{14} & A_{15} \\ & A_{22} & A_{23} & A_{24} & A_{25} \\ & A_{32} & 0 & A_{34} & A_{35} \end{bmatrix}.$$

The SVD corresponding to block column 4 computes updating rotations U_{up} and V_{up} such that

$$(AD)\begin{bmatrix} \hat{W} & \hat{W}^\perp \end{bmatrix} = \begin{bmatrix} \hat{Q} & \hat{Q}_\perp \end{bmatrix} \hat{M}$$

where

$$\begin{aligned} \begin{bmatrix} \hat{W} & \hat{W}^\perp \end{bmatrix} &\doteq \begin{bmatrix} W & W^\perp \end{bmatrix} V_{up} \\ \begin{bmatrix} \hat{Q} & \hat{Q}_\perp \end{bmatrix} &\doteq \begin{bmatrix} Q & Q_\perp \end{bmatrix} U_{up} \\ \hat{M} &\doteq U_{up}^T M V_{up} \end{aligned}$$

and where $\hat{M}$ now has the pattern

$$\hat{M} = \begin{bmatrix} \hat{B} & \hat{A}_{12} & \hat{A}_{13} & 0 & \hat{A}_{15} \\ & \hat{A}_{22} & \hat{A}_{23} & \hat{A}_{24} & \hat{A}_{25} \\ & \hat{A}_{32} & \hat{A}_{33} & 0 & \hat{A}_{35} \end{bmatrix}. \tag{8}$$

The transformations U_{up} and V_{up} are in fact implemented in two steps. First, we construct a transformation G_0 applied to the bottom 2 block rows of M in order to eliminate the A_{34} block, corresponding to the Gram-Schmidt expansion of the current factorization:

$$\tilde{M} \doteq \begin{bmatrix} I_k & \\ & G_0 \end{bmatrix} M = \begin{bmatrix} B & A_{12} & 0 & A_{14} & A_{15} \\ & \tilde{A}_{22} & \tilde{A}_{23} & \tilde{A}_{24} & \tilde{A}_{25} \\ & \tilde{A}_{32} & \tilde{A}_{24} & 0 & \tilde{A}_{35} \end{bmatrix}. \tag{9}$$

In a second stage the $(k+l)\times(k+l)$ transformations G_u and G_v are computed to perform the SVD

$$G_u^T \begin{bmatrix} B & A_{14} \\ & \tilde{A}_{24} \end{bmatrix} G_v = \begin{bmatrix} \hat{B} & 0 \\ 0 & \hat{A}_{24} \end{bmatrix}.$$

Then U_{up} is the product of embeddings of G_0 and G_u, and V_{up} is an embedding of G_v. We now prove the following results regarding this j-th block row step (here $j=4$).

Lemma 1. *In the above j-th updating step we have the following inequalities*

$$\operatorname{trace}\left(\hat{B}\hat{B}^T\right) \geq \operatorname{trace}\left(BB^T\right) + \operatorname{trace}\left(A_{1j}A_{1j}^T\right)$$

$$\sigma_i(\hat{B}) \geq \sigma_i(B), \quad i=1,\ldots,k$$
$$\|A_{:,i}\|_2 \leq \sigma_{min}(\hat{B}), \quad i=1,\ldots,j$$

Proof. Let us point out that the SVD performed in that step,

$$U_{up}^T M_{1j} G_v = \hat{M}_{1j}, \quad \text{where} \quad M_{1j} \doteq \begin{bmatrix} B & A_{1j} \\ & A_{2j} \\ & A_{3j} \end{bmatrix}, \quad \hat{M}_{1j} \doteq \begin{bmatrix} \hat{B} & 0 \\ & \hat{A}_{2j} \\ & 0 \end{bmatrix}$$

satisfies the following extremal property

$$\operatorname{trace}\left(\hat{B}\hat{B}^T\right) = \max_{U^TU=I_k} \operatorname{trace}\left(U^T M_{1j} M_{1j}^T U\right).$$

The first inequality then follows from the suboptimal matrix $U^T \doteq \begin{bmatrix} I_k & 0 \end{bmatrix}$. The second inequality follows from the fact that when bordering a matrix B as in M_{1j}, all singular values can only increase. The third inequality just says that the singular values of the $(2,2)$ block in $\hat{M}_{1j}$ are smaller than those of its $(1,1)$ block. Notice also that the additional transformations of a single pass do not further affect the norm of the j-th block column. □

The proof of the next theorem then immediately follows from that lemma.

Theorem 1. *After one pass of the Incremental SVD algorithm, the squared Frobenius norm of $\hat{B}$ increased at least with the sum of the squared Frobenius norms of the blocks $A_{1j}^{(j-1)}$ as they were just before iteration j:*

$$\|\hat{B}^{(f)}\|_F^2 \geq \|B^{(0)}\|_F^2 + \sum_{j=2}^{f} \|A_{1j}^{(j-1)}\|_F^2$$

and the 2-norms of the ultimate block columns $A_{:,j}$ are all bounded above by $\sigma_{min}(\hat{B}^{(f)})$. Moreover, the singular values of $\hat{B}$ are strictly increasing as long as the blocks $A_{1j}^{(j-1)}$ are nonzero.

This theorem includes a subtlety, in that the entry $A_{1j}^{(j-1)}$ as seen "just before iteration j" depends on the value of Q, which potentially (and presumably) changes during the pass. Furthermore, all entries of $[A_{12}\ldots A_{1f}]^{(j)}$ potentially change if Q changes at step j. However, it is also that case that the change in Q at step j is dictated by the value of $A_{1j}^{(j-1)}$. After the first pass, the norm of $A_{:,j}$ is less than $\sigma_{min}(\hat{B})$, and therefore the norm of A_{2j} is so bounded. Then, so long as $\sigma_k(A)$ is strictly larger than $\sigma_{k+1}(A)$, the only change to Q comes from the elimination of $A_{1j}^{(j-1)}$ at step j, where the assumption stands to prevent simple swapping of rows between Q and $Q_\perp$. Therefore, when $A_{1j}^{(j-1)}$ is zero, there is no change to Q and no change to the other A_{1i}. This assumption on the $\sigma(A)$ is reasonable, as it is otherwise necessary so that the rank-k dominant singular subspaces are uniquely defined; therefore, we shall assume henceforth that $\sigma_k(A)$ is strictly greater than $\sigma_{k+1}(A)$. With this corollary and the previous theorem, we can prove the following result regarding the convergence of the multipass iteration.

Theorem 2. *The Multipass IncSVD algorithm converges to a matrix M where the blocks $[A_{12}\dots A_{1f}]$ are zero, and where Q and V contain singular subspaces of A.*

Proof. Assume for the purpose of contradiction that $[A_{12}\dots A_{1f}]$ does not converge to zero, i.e., there exists some ϵ such that there are an infinite number of passes after which $[A_{12}\dots A_{1f}]$ is greater than ϵ. Because the singular values of B are upper bounded by those of A and because the previous theorem states that they are strictly increasing as long as the block $A_{1j}^{(j-1)}$ at each iteration j of the pass is non-zero, it follows that the norm of the these $A_{1j}^{(j-1)}$ must converge to zero. As a result, Q converges to some limit point, so that all of the blocks A_{1j} stabilize. As they will each be considered in turn, their limit points must each be zero. However, this contradicts our assumption that the block row has a norm greater than ϵ in this sequence, and thus we have $[A_{12}\dots A_{1f}]$ converging to zero.

Then, in the limit, M becomes block diagonal and we have a factorization of the form

$$A\begin{bmatrix}V & V_\perp\end{bmatrix} = \begin{bmatrix}Q & Q_\perp\end{bmatrix}\begin{bmatrix}B & 0\\ 0 & A_2\end{bmatrix} .$$

From this, we identify that $AV = QB$ and $A^TQ = VB$, and it is apparent that Q and V describe left and right singular subspaces of A, respectively. □

Note that this result claims convergence to some pair of singular subspaces, instead of the stronger and more desirable result of convergence to the dominant singular subspaces. Unfortunately, as with other ascent methods such as gradient ascent, there is the possibility of convergence to saddle points (critical points which are not local extrema) [26, 27]. However, because the incremental SVD is an ascent method, convergence to saddle points is unstable, i.e., any neighborhood of a saddle contains a point from which the algorithm will proceed away from the saddle point. In practice, demonstrating convergence to saddle points is possible only for carefully constructed matrices A. These cases can typically be handled by choosing a random initial V_0. In general, and like other ascent methods, the incremental SVD enjoys global convergence to a local maximizer, i.e., the dominant singular subspaces.

We will assume from now on that we are sufficiently close to the dominant SVD of A, such that M is nearly block diagonal and that the norms of the A_{1i} blocks are bounded by $\delta \ll \sigma_k(A) - \sigma_{k+1}(A)$. Since we know we eventually converge to a block diagonal matrix and we have stable convergence only to local maximizers, this is a valid assumption when studying the ultimate rate of convergence to the dominant SVD. The following Lemma is inspired from the perturbation theory of [3].

Lemma 2. *At some step j, let $\sigma_+ \doteq \sigma_{min}(B)$, $\sigma_- \doteq \|\tilde{A}_{2j}\|_2$, $\delta \doteq \|A_{1j}\|_2$ and suppose that $\delta \ll \sigma_+ - \sigma_-$. Then the block diagonalization*

$$G_u^T\begin{bmatrix}B & A_{1j}\\ 0 & \tilde{A}_{2j}\end{bmatrix}G_v = \begin{bmatrix}\hat{B} & 0\\ 0 & \hat{A}_{2j}\end{bmatrix}$$

is obtained by transformation matrices of the form

$$G_u = \begin{bmatrix}I_k & -X^T\\ X & I_l\end{bmatrix} + \mathcal{O}(\delta^2), \quad G_v = \begin{bmatrix}I_k & -Y^T\\ Y & I_l\end{bmatrix} + \mathcal{O}(\delta^2), \tag{10}$$

where $\|X\|_2 \le \delta\sigma_-/(\sigma_+^2 - \sigma_-^2) + \mathcal{O}(\delta^2)$.

Proof. Theorem 4.4 of [3] says that the orthogonal transformation matrices performing the block diagonalization can be chosen of the form (10) with matrices X and Y that are $\delta/(\sigma_+ - \sigma_-)$-close to the identity. Moreover, we can exploit the fact that $\tilde{M}_{1j}$ is upper block triangular to improve the bound on X. The off-diagonal blocks of $\hat{M}_{1j}$ yield the equations

$$XB = \tilde{A}_{2j}Y + \mathcal{O}(\delta^2), \quad A_{1j} + X^T\tilde{A}_{2j} = BY^T + \mathcal{O}(\delta^2),$$

from which we obtain the Sylvester equation in X^T

$$A_{1j}\tilde{A}_{2j}^T + X^T\tilde{A}_{2j}\tilde{A}_{2j}^T = BB^TX^T + \mathcal{O}(\delta^2) .$$

The direct application of Lemma 3.5 in [3] then yields the bound $\|X\|_2 \leq \|A_{1j}\tilde{A}_{2j}^T\|_2/(\sigma_+^2 - \sigma_-^2) + \mathcal{O}(\delta^2)$ which is the desired result. □

Let us now look at the effect of one such transformation on the full matrix M. The right transformation G_v only affects the block columns 1 and j, but the left transformation G_u^T affects all block columns and yields the following updates for the blocks in the first row:

$$\hat{A}_{1i} \doteq A_{1i} + X^T\tilde{A}_{2i} + \mathcal{O}(\delta^2) \ .$$

In analyzing this update, we seek a bound on the norm of the matrix $\tilde{A}_{2i}$. We could simply use the norm of the entire block column containing $\tilde{A}_{2i}$, an amount known *a posteriori* as the largest discarded singular value corresponding to that step, an amount that must itself be bounded above by σ_{k+1}. However, this overestimate does not consider the fact that some of the energy of the i-th block column may be stored in the third block row, in $\tilde{A}_{3i}$, where it will not influence $\hat{A}_{1i}$ under G_u^T. As will be detailed below, certain properties of the data matrix A and the search directions D can effect the distribution of energy in the i-th block column to the benefit of the algorithm's performance. We wish therefore to include this information in our bound.

In order to bound the norm of $\tilde{A}_{2i}$, we need to quantify the proportion of energy contained there relative to the entire energy for the i-th block column. Specifically, we are interested in the ratio

$$\gamma_{ij} = \frac{\|\tilde{A}_{2i}^{(j)}\|_2}{\left\|\begin{bmatrix} A_{1i}^{(j)} & \tilde{A}_{2i}^{(j)} & \tilde{A}_{3i}^{(j)} \end{bmatrix}\right\|_2},$$

where i denotes the index of the block column under consideration and j denotes the step number. We are concerned only with $i < j$, as these are the block columns, already processed by the algorithm, whose fill-in of the first block row is the subject of our bounding effort. Note that all $\gamma_{ij} \leq 1$. For matrix partitionings where block columns have one vector, γ_{ij} approximates the angle between the data truncated at step i and the incoming data at step j (this neglects the negligible components in $A_{1i}^{(j)}$). In the more general case consisting of non-trivial block widths, this is not exactly the same anymore. Nevertheless, it remains true that $\gamma_{ij} = 0$ iff the respective block columns are orthogonal.

We now use this to bound the norm of the new $\hat{A}_{1i}$ blocks at the end of one pass of the algorithm. In the next theorem, the quantities δ, σ_-, σ_+ and γ could be defined as a function of the step j as well as the pass; this would produce a tighter bound. For simplicity, we will allow them to take their worst-case values or their nearly-converged values, whichever is appropriate. We will define

$$\begin{aligned} \sigma_+ &= \min_j \sigma_{min}(B^{(j)}) \\ \sigma_- &= \max_j \|\hat{A}_{2j}^{(j)}\|_2 \\ \delta &= \max_j \|\hat{A}_{1j}^{(j)}\|_2 \\ \gamma &= \max_{i<j} \gamma_{ij} \ . \end{aligned}$$

Under these definitions, σ_+ is the smallest singular value of B; upon convergence, this term tends to $\sigma_k(A)$, and it does not change dramatically from one step j to the next. Term σ_- is the largest truncated singular value, which is bounded above by $\sigma_{k+1}(A)$; δ is the largest first block row norm; and γ is the worst-case energy distribution involved in fill-in of the first block row.

Theorem 3. *Let γ, σ_+, σ_- and δ be defined as in the preceding paragraph, and define $c \doteq \gamma\sigma_-^2/(\sigma_+^2 - \sigma_-^2)$. If $cf \leq 1$, then the ultimate rate of convergence of the Multipass IncSVD algorithm is linear and the norm of the off diagonal blocks A_{1i} decreases by a factor cf at each pass of the algorithm.*

Proof. At each step j of the pass, we apply a rotation G_u^T to annihilate the block A_{1j}. That block gets annihilated, but all other blocks A_{1i} can slightly increase by an amount bounded by $\|X^T \tilde{A}_{2i}\| \leq [\delta\sigma_- / (\sigma_+^2 - \sigma_-^2)] \cdot [\sigma_- \gamma] = c\delta$. Since all $f-1$ blocks in the first row get annihilated at least once per pass, the maximum norm of any block is $c(f-2)\delta$ at the end of the pass, provided we neglect second order terms in δ. □

Remark 2. The condition $cf < 1$ is achieved as soon as we have $\frac{\sigma_k}{\sigma_{k+1}} \geq \sqrt{1 + f\gamma}$ which looks quite demanding. But this is a severe overestimate due to the way our bound was obtained. The growth of the blocks is very likely not to accumulate and we expect instead to have the more practical condition $c < 1$ which is instead satisfied as

$$\frac{\sigma_k}{\sigma_{k+1}} \geq \sqrt{1+\gamma} \ .$$

Note that, given a matrix A with a fixed n number of columns, the number of iterations f required to complete one pass of the MultiPass IncSVD depends on the average size l of the block updates at each step, according to $f \approx n/l$. Theorem 3 suggests that the rate of the convergence of the multipass algorithm improves as update size is increased (so that f is decreased). This follows intuition; larger block updates allow more of A to be considered at each step. In the extreme (but presumably intractable) case where $l = n - k$ and we perform $f = 1$ iterations per pass, it is straightforward to show that the algorithm requires only one pass to compute exactly the dominant SVD. This is because the local approximation of Algorithm 1 achieves a global perspective in this circumstance.

Notice that the convergence is fast under two circumstances: for small γ, and for large gaps between σ_k and σ_{k+1}. The latter is well-known, being mentioned in previous literature. A larger gap between σ_k and σ_{k+1} allows the algorithm to more easily distinguish between dominant and subordinate singular subspaces. As σ_k/σ_{k+1} goes to infinity, c goes to zero, and the convergence is expected to accelerate as well. In particular, if A has rank k, i.e., $\sigma_{k+1} = 0$, then $c = 0$ and convergence should occur in one pass. All previous literature has remarked on the ability of a low-rank incremental SVD to capture the dominant SVD of a low-rank matrix.

As regards γ, it is clear that a smaller value yields faster convergence. The question remains as to how this term γ, computable only *a posteriori* and at great expense, is influenced by *a priori* properties of A. The ratios γ_{ij} are small when there is little correspondence between truncated data and incoming data. Therefore, if the truncated data is orthogonal to the incoming data, the convergence is predicted to be immediate. This will be the case, in particular, if the columns of AD are themselves orthogonal. However, this is only likely to happen in two scenarios, neither of which are easily duplicated. The first is the unlikely scenario where A has orthogonal columns. The second is the scenario where AD has orthogonal block columns, i.e., the block columns of D contain the right singular vectors. However, even if all right singular vectors were known to us (in which case, the Incremental SVD is not needed), to form AD for the purposes of computing its Incremental SVD would be too expensive. However, there is a situation where we can exploit this effect. In the scenario where $\sigma_{k+1} = \ldots = \sigma_n$, the matrix AD becomes orthogonal near convergence. As the first block column of D converges to a dominant right singular basis, the latter columns converge to a subordinate right singular basis, so that $AV_\perp = Q_\perp \sigma_{k+1}$ has orthogonal columns. In this case, it can even be proven that convergence occurs in one pass. For the nearby problems where $\sigma_{k+1} \approx \sigma_n$, we still see this effect drive γ to zero and improve the speed of convergence, as we will demonstrate in the following section.

5. Numerical Performance

We present some empirical evidence regarding the numerical performance of the single- and multi-pass Incremental SVD methods presented in this paper. First, we present a qualitative examination of a single pass of Algorithm 1 over a benchmark database of images. Next, we present a study of the convergence of the proposed multipass algorithms on a set of synthetic matrices. These matrices are parameterized according to singular values, in an attempt to demonstrate the convergence speed under the scenarios discussed in the previous section. Lastly, we compare the performance of the simple multipass algorithm (Algorithm 2) against the gradient-accelerated multipass algorithm (Algorithm 3). The following experiments are conducted in

MATLAB (R2008a) on a Intel-based Linux computer. The algorithmic implementations are available in the Incremental SVD package[1].

A low-rank incremental SVD will in most cases produce approximations to the dominant singular subspaces of a matrix. While Section 4 described techniques for improving the quality of this approximation, a single pass approach may be useful in circumstances where great accuracy is unnecessary or where the availability of A admits only a single pass.

Figure 3 shows the dominant left singular vectors computed via one pass through the ORL Database of Faces [28]. This database consists of 10 images each of 40 subjects, each containing 92×112 grayscale pixels. The resulting matrix is 10304×400. Algorithm 1 computed $k = 10$ dominant singular triplets, with an update of $l = 10$ columns of A at each iteration. For qualitative comparison, Figure 4 displays the left singular vectors produced by computing all singular triplets via MATLAB's `svd` function. The maximum angle between the two subspaces was 16.3°; the maximum relative error in the computed singular values was 4.8%. This experiment reproduces the results achieved in [15].

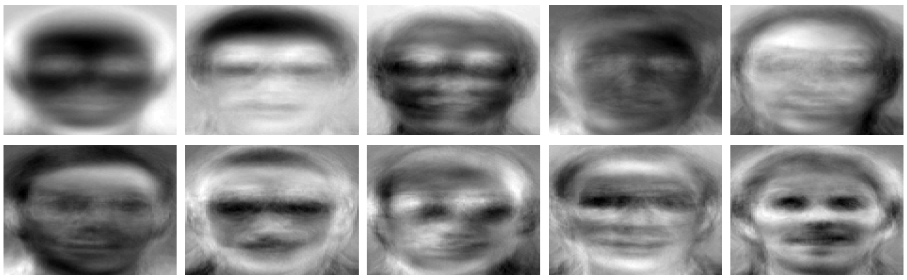
Figure 3: Left singular vectors ("eigenfaces") for ORL Database of Faces computed via Algorithm 1 with rank $k = 10$ and updates of size $l = 10$.

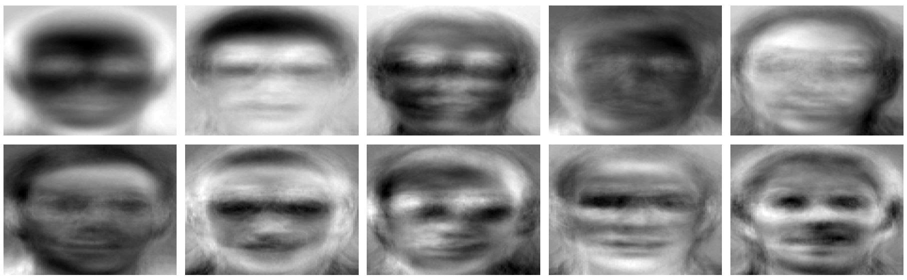
Figure 4: Left singular vectors for ORL Database of Faces computed directly via MATLAB's `svd`.

In the case that a more accurate decomposition is needed, the multipass algorithms of Section 4 can be used, assuming that the matrix A is available for multiple passes. The analysis in Section 4.2 shows that the multipass algorithms converge to the dominant SVD. The convergence analysis proved a linear rate of convergence with coefficient $c \leq \gamma/(\kappa^2 - 1)$, where γ is as previously defined and $\kappa = \sigma_k/\sigma_{k+1}$. In these experiments, randomly generated singular bases U and V were combined with singular values synthesized in order to illustrate the effect that the singular values of A have on the convergence speed of the multipass algorithm. In particular, we demonstrate the effect of larger κ on c; and synthetically reducing the gap $\sigma_{k+1} - \sigma_n$, in order to reduce γ. Figures 5 and 6 illustrate the results of this experiment.

The test matrix for the experiments in Figures 5 and 6 had dimension $m \times n$, where $m = 10,000$ and $n = 500$. Algorithm 2 was used to compute the dominant $k = 10$ singular triplets. Each invocation was

[1]IncPACK: http://www.math.fsu.edu/∼cbaker/IncPACK/

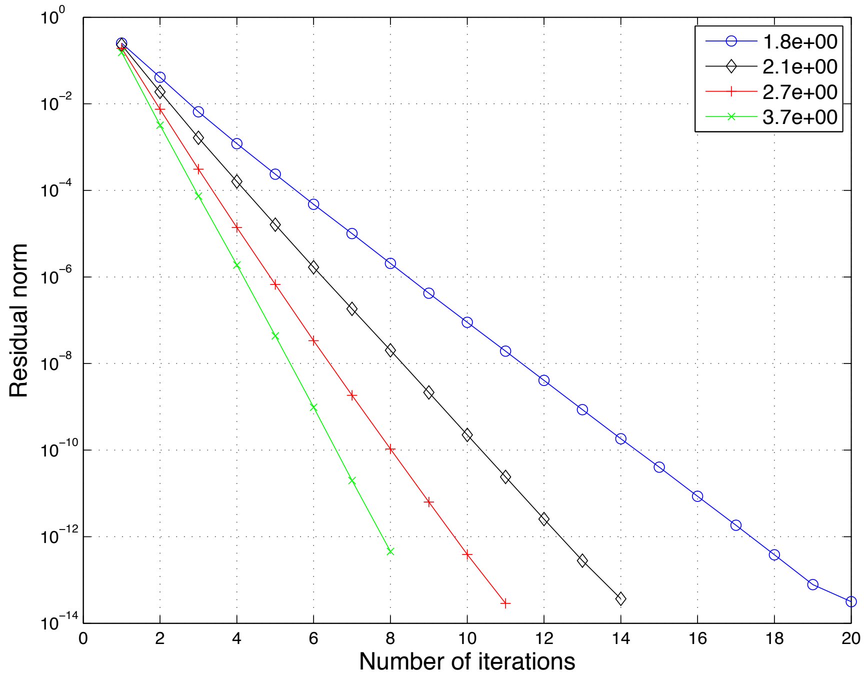

Figure 5: Singular values σ_1 and σ_n remain constant for all tests, as do the singular bases. $\sigma_{2:k}$ are modified in order to increase κ, while $\sigma_{k+1:n}$ are left constant. The plot labels denote the synthesized κ.

allowed 49 passes through A, corresponding to 25 iterations of Algorithm 2 (the first iteration of Algorithm 2 requires only a single pass through A if $V_0 = \begin{bmatrix} I & 0 \end{bmatrix}^T$); iterations after convergence to machine precision are neglected from the plot. The figures plot the norm of the residual $A^T U - V\Sigma$, for iteration estimates U, Σ and V. This norm is equivalent to that of the first block row in the matrix M, analyzed in Section 4.2, and the norm of the gradient of the Rayleigh quotient. In Figure 5, the singular values were modified to illustrate the effect of the ratio σ_k/σ_{k+1}. These results suggest that a larger gap between the targeted and discarded singular subspaces results in better performance of the algorithm. This is not a new result; the previous literature discusses the importance of this gap. However, this is the first demonstration of the impact of this gap on the convergence of a multipass algorithm. In Figure 6, the singular values were modified to illustrate the effect of the gap $\sigma_{k+1} - \sigma_n$. As predicted, a small gap leads to small γ and faster convergence. In particular, for the case of $\sigma_{k+1} = \ldots = \sigma_n$, the algorithm converges in one iteration of Algorithm 2, i.e., one pass of Algorithm 1. In each case, the algorithm converges linearly to the dominant SVD of A. Table 1 lists the observed convergence rates for these plots as well as the bounds predicted by the convergence analysis.

Table 1: Observed and predicted convergence rates for the experiments in Figures 5 and 6.

Figure 5				Figure 6			
κ	γ	obs. c	pred. c	κ	γ	obs. c	pred. c
1.8	0.46	0.20	0.22	1.7	0.47	0.19	0.22
2.1	0.46	0.10	0.13	1.7	0.41	0.19	0.21
2.7	0.46	0.05	0.07	1.4	0.11	0.076	0.10
3.7	0.46	0.02	0.03	1.3	0.0069	0.0069	0.0086
				1.3	6.5e-15	perfect	8.4e-15

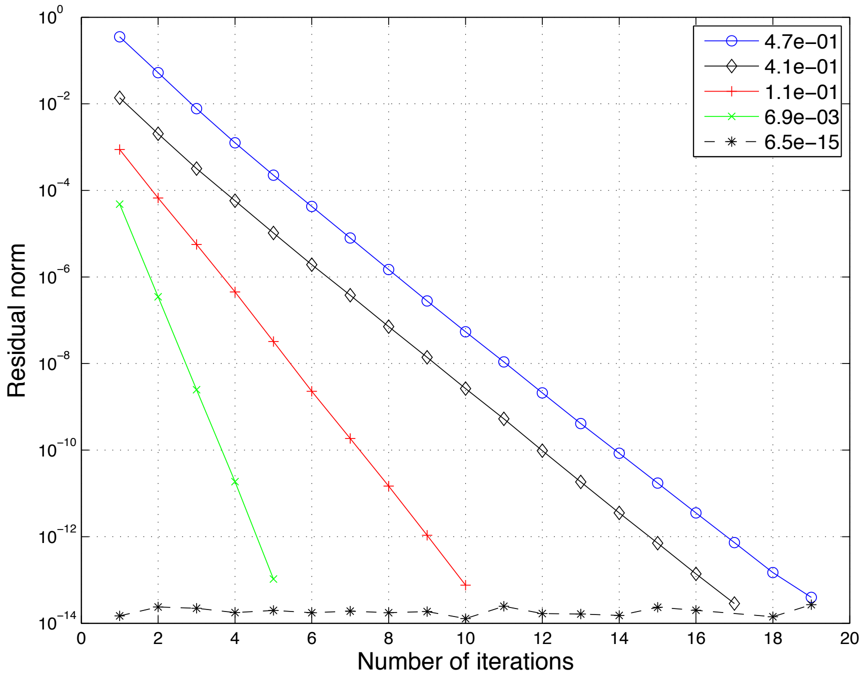

Figure 6: Singular values σ_1 and σ_n remain constant for all tests, as do the singular bases. $\sigma_{k:n-1}$ are modified in order to decrease the gap $\sigma_{k+1} - \sigma_n$. The plot labels denote the observed γ.

The convergence analysis of Section 4.2 assumes that $\sigma_+/\sigma_- > 1$, which will be satisfied in the case that σ_k is strictly greater than σ_{k+1}. Furthermore, this strict inequality is necessary in order to make a rigorous distinction between the dominated and subordinate singular values and subspaces. However, Figure 7 illustrates that even when $\sigma_k = \sigma_{k+1}$, the multipass algorithm may enjoy convergence, at a linear rate, albeit a very slow one.

Lastly, Figure 8 compares the convergence of Algorithm 2 against Algorithm 3. Note that the gradient information injected into Algorithm 3 improves the rate of convergence, as intended. This figure plots the error in the left singular subspace; this "subspace error" is computed as the sum of the squares of the canonical angles between the basis produced by the Incremental SVD and the basis produced by MATLAB's `svd`. This error metric shows a similar plot as does the residual error metric used in the previous figures. In particular, it should be noted here that this is plotted against the number of algorithmic iterations. Because of the need to construct and include the gradient information, Algorithm 3 incurs a higher cost per iteration, in terms of memory storage, floating point operations, and data movement of A. This should be considered in an ultimate comparison of the two algorithms.

6. Concluding Remarks

Low-rank incremental SVD methods have been repeatedly and independently described in the literature. This paper presented a generic approach, unifying the previous methods. We also presented an exploration of the underlying mechanics of the iteration, resulting in a link between the Incremental SVD of A and an iterative solution of the related eigenvalue problem on A^TA. This decoupled the method from its heuristic origins and enabled the description of techniques for restarting the incremental SVD, in order to exploit multiple passes through A in applications where this is a possibility. We presented a convergence analysis for the multipass iteration, with *a priori* bounds on the rate of convergence, and we illustrated these bounds on synthetic problems with key characteristics. In particular, we demonstrated a special convergence scenario

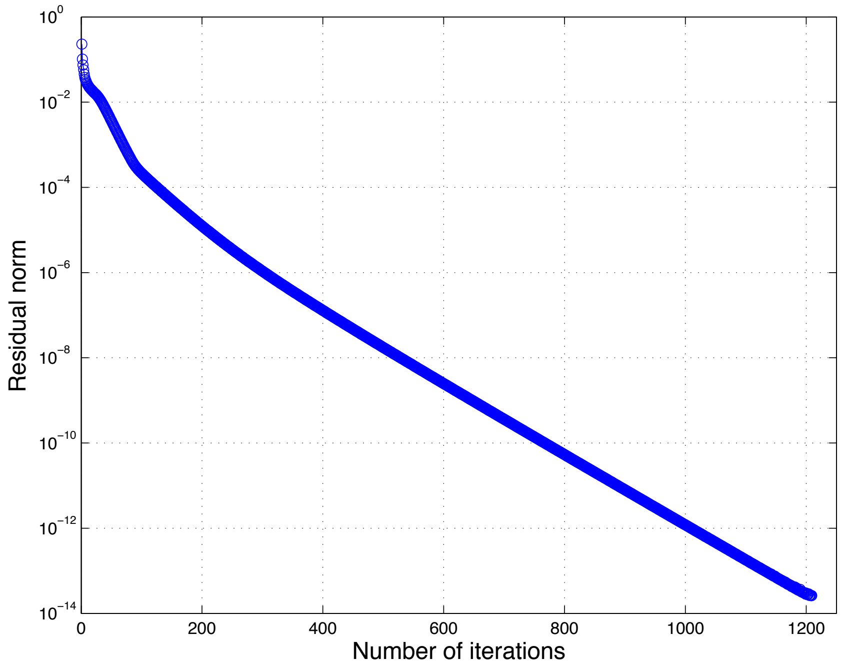

Figure 7: No gap between targeted and neglected singular values: $\sigma_k = \sigma_{k+1}$ sees slow, but eventual, convergence.

where a matrix A with $\sigma_{k+1}(A) = \sigma_n(A)$ enjoys convergence in a single pass of the incremental SVD. The latter may warrant further study, as it implies that (under this scenario) the full set of singular values can be identified in $O(mnk)$ and via only a single pass through the matrix.

The idea behind the Incremental SVD is simple: a linear-time incremental pass through A, tracking a low-rank factorization updated in a locally optimal manner. This idea can potentially be applied to factorizations with different structure and different measures of optimality. Further research will investigate such extensions; for example, for symmetry-preserving SVD [29] or CUR-like decompositions [30]. Also, as briefly mentioned in Section 3, the link with the optimizing eigensolver suggests that the Incremental SVD can be directed to track the smallest singular values; this is confirmed by preliminary experiments. Such an application may be especially useful, since finding the smallest singular values via $A^T A$-based approaches can be difficult due to numerical problems associated with the squaring of the condition number.

The Incremental SVD was designed to address the scenario where access to A is limited. However, to our knowledge, the method has found little use in more general cases. Currently, iterative eigensolvers such as ARPACK [11] provide a successful and popular approach for computing the dominant SVD. These approaches require multiple applications of $A^T A$ and are obviously limited to scenarios where such access to A is available. In the future, we intend to compare the performance of the multipass Incremental SVD against such methods. The comparatively slow (linear) rate of convergence of the multipass Incremental SVD may limit its usefulness in computing high-accuracy singular subspaces. Still, the method may find use in computing a good initial iterate for a Krylov approach or some other locally superlinear method, such as a Newton or trust-region SVD solver [22, 23, 31].

The algorithms described in this paper are freely available in the IncPACK MATLAB package, which may be downloaded from http://www.math.fsu.edu/∼cbaker/IncPACK/.

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

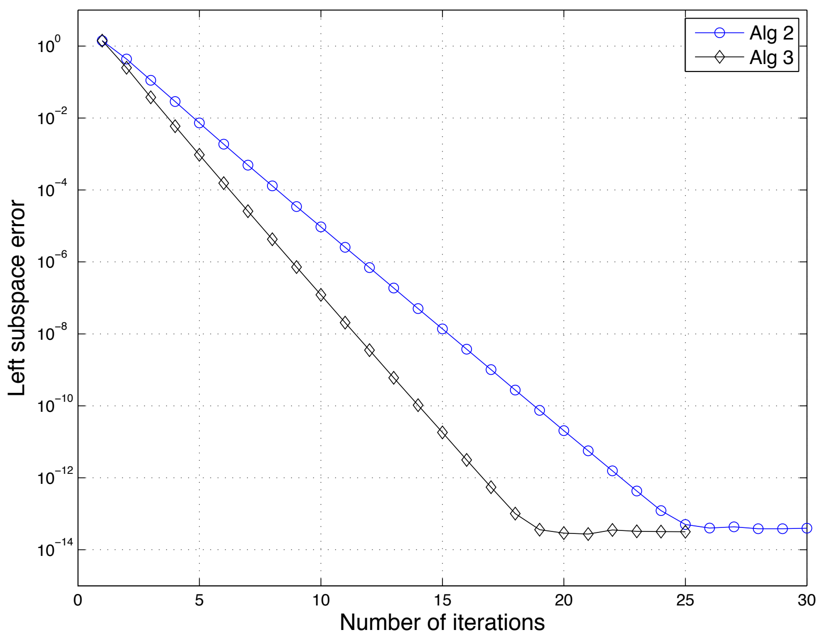

Figure 8: A comparison of Algorithm 2 and Algorithm 3.

References

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

[2] G. W. Stewart. *Matrix Algorithms, Volume I: Basic Decompositions.* Society for Industrial and Applied Mathematics, Philadelphia, 1998.

[3] G. W. Stewart, J. Sun. *Matrix Perturbation Theory.* Academic Press, Boston, 1990.

[4] L. Sirovich and M. Kirby. Low-dimensional procedure for the characterization of human faces. *J. Opt. Soc. Am. A*, 4(3):519–524, 1987.

[5] Matthew Turk and Alex Pentland. Eigenfaces for recognition. *J. Cognitive Neuroscience*, 3(1):71–86, 1991.

[6] Lars Eldén. *Matrix Methods in Data Mining and Pattern Recognition.* Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2007.

[7] L. Sirovich. Empirical eigenfunctions and low dimensional systems. In L. Sirovich, editor, *New Perspectives in Turbulence*, pages 139–163. Springer, New York, 1991.

[8] L. Sirovich. Turbulence and the dynamics of coherent structures. Part I: Coherent structures. *Quarterly of Applied Mathematics*, 45(3):561–571, October 1987.

[9] H. T. Banks, R. C. H. del Rosario and H. T. Tran. Proper orthogonal decomposition-based control of transverse beam vibrations: experimental implementation. *IEEE Transactions on Control Systems Technology*, 10(5):717–726, September 2002.

[10] G. Berkooz, P. Holmes and J. L. Lumley. The proper orthogonal decomposition in the analysis of turbulent flows. *Annual Review of Fluid Mechanics*, 25:539–575, 1993.

[11] R. B. Lehoucq, D. C. Sorensen, C. Yang. *ARPACK Users' Guide: Solution of Large-Scale Eigenvalue Problems with Implicitly Restarted Arnoldi Methods.* Society for Industrial and Applied Mathematics, Philadelphia, 1998.

[12] Matthew Brand. Fast online svd revisions for lightweight recommender systems. In *In SIAM International Conference on Data Mining*, 2003.

[13] 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.

[14] S. Chandrasekaran, B. S. Manjunath, Y. F. Wang, J. Winkeler and H. Zhang. An eigenspace update algorithm for image analysis. *Graphical models and image processing: GMIP*, 59(5):321–332, 1997.

[15] 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.

[16] 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.

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

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

[19] Matthew Brand. Fast low-rank modifications of the thin singular value decomposition. *Linear Algebra and its Applications*, 415(1):20 – 30, 2006. Special Issue on Large Scale Linear and Nonlinear Eigenvalue Problems.

[20] 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.

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

[22] P.-A. Absil, R. Mahony and R. Sepulchre. *Optimization Algorithms on Matrix Manifolds.* Princeton University Press, Princeton, first edition, 2007.

[23] 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.

[24] C. Bischof and C. Van Loan. The WY representation for products of Householder matrices. *SIAM J. Sci. Stat. Comput.*, 8:s2–s13, 1987.

[25] R. Schreiber and C. Van Loan. A storage-efficient WY representation for products of Householder transformations. *SIAM J. Sci. Stat. Comput.*, 10(1):53–57, January 1989.

[26] A. Goldstein. *Constructive real analysis.* Harper & Row, New York, 1967.

[27] J. Dennis and R. Schnabel. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations.* Society for Industrial and Applied Mathematics, Philadelphia, 1996.

[28] F.S. Samaria and A.C. Harter. Parameterisation of a stochastic model for human face identication. In *Applications of Computer Vision*, 1994, Proceedings of the Second IEEE Workshop on, pages 138–142, Dec 1994.

[29] M.I. Shah and D.C. Sorensen. A symmetry preserving singular value decomposition. *SIAM J. Matrix Anal. Appl.*, 28(3):746–769, 2006.

[30] P. Drineas, R. Kannan and M.W. Mahoney. Fast Monte Carlo algorithms for matrices III: Computing a compressed approximate matrix decomposition. *SIAM J. Comput.*, 36(1):184–206, 2006.

[31] M.E. Hochstenbach. A Jacobi-Davidson type SVD method. *SIAM J. Sci. Comput.*, 23(2):606-628, 2001.