EECS 598-005
ML Reproducibility Paper Selection

Andy Chen, Shion Matsumoto, Rohan Sinha Varma

October 4, 2021

Projection techniques to update the truncated SVD of evolving matrices

Vassilis Kalantzis, Georgios Kollias, Shashanka Ubaru, Athanasios N. Nikolakopoulos, Lior Horesh, Kenneth L. Clarkson

We chose this paper for the ubiquitous nature of SVD in machine learning, specifically in its applications for NLP with regards to latent semantic indexing and recommender systems. Additionally, this paper is related to NLA as it is focused on approximating the span of the given matrix (dimension reduction) through constructing a pair of subspaces. Overall, the paper is a nice intersection between our interest in NLP applications and NLA.

References

[Kal+20] Vassilis Kalantzis et al. "Projection techniques to update the truncated SVD of evolving matrices". In: (2020). arXiv: 2010.06392. URL: http://arxiv.org/abs/2010.06392.

Projection techniques to update the truncated SVD of evolving matrices

Vassilis Kalantzis* Georgios Kollias† Shashanka Ubaru‡
Athanasios N. Nikolakopoulos§ Lior Horesh¶ Kenneth L. Clarkson‖

Abstract

This paper considers the problem of updating the rank-k truncated Singular Value Decomposition (SVD) of matrices subject to the addition of new rows and/or columns over time. Such matrix problems represent an important computational kernel in applications such as Latent Semantic Indexing and Recommender Systems. Nonetheless, the proposed framework is purely algebraic and targets general updating problems. The algorithm presented in this paper undertakes a projection viewpoint and focuses on building a pair of subspaces which approximate the linear span of the sought singular vectors of the updated matrix. We discuss and analyze two different choices to form the projection subspaces. Results on matrices from real applications suggest that the proposed algorithm can lead to higher accuracy, especially for the singular triplets associated with the largest modulus singular values. Several practical details and key differences with other approaches are also discussed.

Keywords: Singular Value Decomposition, evolving matrices, Lanczos bidiagonalization, latent semantic analysis, recommender systems

1 Introduction

This paper considers the update of the truncated SVD of a sparse matrix subject to additions of new rows and/or columns. More specifically, let $B \in \mathbb{C}^{m\times n}$ be a matrix for which its rank-k (truncated) SVD B_k is available. Our goal is to obtain an approximate rank-k SVD A_k of matrix

$$A = \begin{pmatrix} B \\ E \end{pmatrix}, \quad \text{or} \quad A = \begin{pmatrix} B & E \end{pmatrix},$$

where E denotes the matrix of newly added rows or columns. This process can be repeated several times, where at each instance matrix A becomes matrix B at the next level. Note that a similar problem, not explored in this paper, is to approximate the rank-k SVD of B after modifying its (non-)zero entries, e.g., see [31].

Matrix problems such as the ones above hold an important role in several real-world applications. One such example is Latent Semantic Indexing (LSI) in which the truncated SVD of the current term-document matrix needs to be updated after a few new terms/documents have been added to the collection [3, 7, 31]. Another example is the update of latent-factor-based models of user-item rating matrices in top-N recommendation [6, 19, 23]. Additional applications in geostatistical screening can be found in [13, Chapter 6].

The standard approach to compute A_k is to disregard any previously available information and apply directly to A an off-the-shelf, high-performance, SVD solver [2, 12, 29, 10, 27]. This standard approach might be feasible when the original matrix is updated only once or twice, however becomes increasingly impractical as multiple row/column updates take place over time. Therefore, it becomes crucial to develop algorithms which return a reasonable approximation of A_k while taking advantage of B_k. Such schemes have already been considered extensively for the case of full SVD [4, 9, 17] and rank-k SVD [3, 23, 28, 31]. Nonetheless, for general-purpose matrices it is rather unclear how to enhance their accuracy.

1.1 Contributions.

1. We propose and analyze a projection scheme to update the rank-k SVD of evolving matrices. Our scheme uses a right singular projection subspace equal to $\mathbb{C}^n$, and only determines the left singular projection subspace.
2. We propose and analyze two different options to set

*IBM Research, Thomas J. Watson Research Center, Yorktown Heights, NY 10598, US. Email: vkal@ibm.com
†IBM Research, Thomas J. Watson Research Center, Yorktown Heights, NY 10598, US. Email: gkollias@us.ibm.com
‡IBM Research, Thomas J. Watson Research Center, Yorktown Heights, NY 10598, US. Email: shashanka.ubaru@us.ibm.com
§University of Minnesota, Minneapolis, MN 55455, US. Email: anikolak@umn.edu
¶IBM Research, Thomas J. Watson Research Center, Yorktown Heights, NY 10598, US. Email: lhoresh@us.ibm.com
‖IBM Research, Almaden Research Center, San Jose, CA 95120, US. Email: klclarks@us.ibm.com

the left singular projection subspace. A complexity analysis is also presented.

3. We present experiments performed on matrices stemming from applications in LSI and recommender systems. These experiments demonstrate the numerical behavior of the proposed scheme and showcase the various tradeoffs in accuracy versus complexity.

2 Background and notation

The (full) SVD of matrix B is denoted as $B = U\Sigma V^H$ where $U \in \mathbb{C}^{m\times m}$ and $V \in \mathbb{C}^{n\times n}$ are unitary matrices whose j'th column is equal to the left singular vector $u^{(j)}$ and right singular vector $v^{(j)}$, respectively. The matrix $\Sigma \in \mathbb{R}^{m\times n}$ has non-zero entries only along its main diagonal, and these entries are equal to the singular values $\sigma_1 \geq \cdots \geq \sigma_{\mathtt{min}(m,n)}$. Moreover, we define the matrices $U_j = \left[u^{(1)}, \ldots, u^{(j)}\right]$, $V_j = \left[v^{(1)}, \ldots, v^{(j)}\right]$, and $\Sigma_j = \mathtt{diag}\left(\sigma_1, \ldots, \sigma_j\right)$. The rank-$k$ truncated SVD of matrix B can then be written as $B_k = U_k\Sigma_k V_k^H = \sum_{j=1}^{k} \sigma_j u^{(j)} \left(v^{(j)}\right)^H$. We follow the same notation for matrix A with the exception that a circumflex is added on top of each variable, i.e., $A_k = \widehat{U}_k\widehat{\Sigma}_k\widehat{V}_k^H = \sum_{j=1}^{k} \widehat{\sigma}_j\widehat{u}^{(j)} \left(\widehat{v}^{(j)}\right)^H$, with $\widehat{U}_j = \left[\widehat{u}^{(1)}, \ldots, \widehat{u}^{(j)}\right]$, $\widehat{V}_j = \left[\widehat{v}^{(1)}, \ldots, \widehat{v}^{(j)}\right]$, and $\widehat{\Sigma}_j = \mathtt{diag}\left(\widehat{\sigma}_1, \ldots, \widehat{\sigma}_j\right)$.

The routines $\mathtt{nr}(K)$ and $\mathtt{nnz}(K)$ return the number of rows of matrix and non-zero entries of matrix K, respectively. Throughout this paper $\|\cdot\|$ will stand for the ℓ_2 norm when the input is a vector, and the spectral norm when the input is a matrix. Moreover, the term $\mathtt{range}(K)$ will denote the column space of matrix K, while $\mathtt{span}(\cdot)$ will denote the linear span of a set of vectors. The identity matrix of size n will be denoted by I_n.

2.1 Related work.

The problem of updating the SVD of an evolving matrix has been considered extensively in the context of LSI. Consider first the case $A = \begin{pmatrix} B \\ E \end{pmatrix}$, and let $(I - V_kV_k^H)E^H = QR$ such that Q is orthonormal and R is upper trapezoidal. The scheme in [31] writes

$$\begin{pmatrix} B \\ E \end{pmatrix} \approx \begin{pmatrix} U_k\Sigma_kV_k^H \\ E \end{pmatrix} = \begin{pmatrix} U_k & \\ & I_s \end{pmatrix} \begin{pmatrix} \Sigma_k \\ EV_k \;\; R^H \end{pmatrix} \begin{pmatrix} V_k \;\; Q \end{pmatrix}^H$$
$$= \left(\begin{pmatrix} U_k & \\ & I_s \end{pmatrix} F \right) \Theta \left(\begin{pmatrix} V_k \;\; Q \end{pmatrix} G \right)^H$$

where the matrix product $F\Theta G^H$ denotes the compact SVD of the matrix $\begin{pmatrix} \Sigma_k \\ EV_k \;\; R^H \end{pmatrix}$.

The above idea can be also applied to $A = \begin{pmatrix} B \;\; E \end{pmatrix}$. Indeed, if matrices Q and R are now determined as $(I - U_kU_k^H)E = QR$, we can approximate

$$\begin{aligned} \begin{pmatrix} B \;\; E \end{pmatrix} &\approx \begin{pmatrix} U_k\Sigma_kV_k^H \;\; E \end{pmatrix} \\ &= \begin{pmatrix} U_k \;\; Q \end{pmatrix} \begin{pmatrix} \Sigma_k \;\; U_k^HE \\ R \end{pmatrix} \begin{pmatrix} V_k^H & \\ & I_s \end{pmatrix} \\ &= \left(\begin{pmatrix} U_k \;\; Q \end{pmatrix} F \right) \Theta \left(\begin{pmatrix} V_k & \\ & I_s \end{pmatrix} G \right)^H \end{aligned}$$

where the matrix product $F\Theta G$ now denotes the compact SVD of the matrix $\begin{pmatrix} \Sigma_k \;\; U_k^HE \\ R \end{pmatrix}$.

When B_k coincides with the compact SVD of B, the above schemes compute the exact rank-k SVD of A, and no access to matrix B is required. Nonetheless, the application of the method in [31] can be challenging. For general updating problems, or problems where A does not satisfy a "low-rank plus shift" structure [32], replacing B by B_k might not lead to a satisfactory approximation of A_k. Moreover, the memory/computational cost associated with the computation of the QR and SVD decompositions in each one of the above two scenarios might be prohibitive. The latter was recognized in [28] where it was proposed to adjust the method in [31] by replacing matrices $(I - V_kV_k^H)E^H$ and $(I - U_kU_k^H)E$ with a low-rank approximation computed by applying the Golub-Kahan Lanczos bidiagonalization procedure [8]. Similar ideas have been suggested in [30] and [26] where the Golub-Kahan Lanczos bidiagonalization procedure was replaced by randomized SVD [10, 25] and graph coarsening [26], respectively.

3 The projection viewpoint

The methods discussed in the previous section can be recognized as instances of a Rayleigh-Ritz projection procedure and can be summarized as follows [28, 30]:

1. Compute matrices Z and W such that $\mathtt{range}(Z)$ and $\mathtt{range}(W^H)$ approximately capture $\mathtt{range}(\widehat{U}_k)$ and $\mathtt{range}(\widehat{V}_k^H)$, respectively.
2. Compute $[\Theta_k, F_k, G_k] = \mathtt{svd}(Z^HAW)$ where Θ_k, F_k, and G_k denote the k leading singular values and associated left and right singular vectors of Z^HAW, respectively.
3. Approximate A_k by the product $(ZF_k)\Theta_k(WG_k)^H$.

Ideally, the matrices Z and W should satisfy

$$\mathtt{span}\left(\widehat{u}^{(1)}, \ldots, \widehat{u}^{(k)}\right) \subseteq \mathtt{range}(Z), \quad \text{and}$$
$$\mathtt{span}\left(\widehat{v}^{(1)}, \ldots, \widehat{v}^{(k)}\right) \subseteq \mathtt{range}(W).$$

Table 1: *Different options to set the projection matrices Z and W for the row updating problem.*

Method	Z	W
[3]		V_k
[31]	$Z = \begin{pmatrix} U_k & \\ & I_s \end{pmatrix}$	$[V_k, Q]$
[28]		$[V_k, X_r]$
Alg. 1	$Z = \begin{pmatrix} U_k & \\ & I_s \end{pmatrix}$	I_n
Alg. 1	$Z = \begin{pmatrix} U_k, X_{\lambda,r} & \\ & I_s \end{pmatrix}$	I_n

Moreover, the size of matrix $Z^H AW$ should be as small as possible to avoid high computational costs during the computation of $[\Theta_k, F_k, G_k] = \texttt{svd}(Z^H AW)$.

Table 1 summarizes a few options to set matrices Z and W for the row updating problem. The method in [28] considers the same matrix Z as in [31] but sets $W = [V_k, X_r]$ where X_r denotes the $r \in \mathbb{Z}^*$ leading left singular vectors of $(I - V_k V_k^H)E^H$. The choice of matrices Z and W listed under the option "Algorithm 1" is explained in the next section. Note that the first variant of Algorithm 1 uses the same Z as in [31] and [28] but different W. This choice leads to similar or higher accuracy than the scheme in [31] and this is also achieved asymptotically faster. A detailed comparison is deferred to the Supplementary Material. The second variant of Algorithm 1 is a more expensive but also more accurate version of the first variant.

3.1 The proposed algorithm. Consider again the SVD update of matrix $A = \begin{pmatrix} B \\ E \end{pmatrix}$, with $E \in \mathbb{C}^{s \times n}$. The right singular vectors of A trivially satisfy $\widehat{v}^{(i)} \subseteq \texttt{range}(I_n),\ i = 1, \ldots, n$. Therefore, we can simply set $W = I_n$ and compute the k leading singular triplets $\left(\theta_i, f^{(i)}, g^{(i)}\right)$ of the matrix $Z^H AW = Z^H A$. Indeed, this choice of W is ideal in terms of accuracy while it also removes the need to compute an approximate factorization of matrix $(I - V_k V_k^H)E^H$. On the other hand, the number of columns in matrix $Z^H AW$ is now equal to n instead of $k + s$ in [31] and $k + l,\ l \ll s$, in [28, 30]. This difference can be important when the full SVD of $Z^H AW$ is computed as in [28, 30, 31].

Our approach is to compute the singular values of $Z^H A$ in a matrix-free fashion while also skipping the computation of the right singular vectors G_k. Indeed, the matrix G_k is only needed to approximate the k leading singular vectors $\widehat{V}_k$ of A. Assuming that an approximation $\overline{U}_k$ and $\overline{\Sigma}_k$ of the matrices $\widehat{U}_k$ and $\widehat{\Sigma}_k$ is available, $\widehat{V}_k$ can be approximated as $\overline{V}_k = A^H \overline{U}_k \overline{\Sigma}_k^{-1}$. The proposed method is sketched in Algorithm 1. In terms of computational cost, Steps 4 and 5 require approximately $2\texttt{nnz}(Z)k$ and $(2\texttt{nnz}(A) + n)k$ Floating Point Operations (FLOPs), respectively. The complexity of Step 3 will generally depend on the algorithm used to compute the matrices Θ_k and F_k. We assume that these are computed by applying the unrestarted Lanczos method to matrix $Z^H AA^H Z$ in a matrix-free fashion [21]. Under the mild assumption that Lanczos performs δ iterations for some $\delta \in \mathbb{Z}^*$ which is greater than or equal to k, a rough estimate of the total computational cost of Step 3 is $4\left(\texttt{nnz}(Z^H) + \texttt{nnz}(A)\right)\delta + 2\texttt{nr}(Z^H)\delta^2$ FLOPs. The exact complexity of Lanczos will depend on the choice of matrix Z. A detailed asymptotic analysis of the complexity of Algorithm 1 and comparisons with other schemes are deferred to the Supplemental.

Algorithm 1 RR-SVD ("AA^H" version).

1: **Input:** $B, U_k, \Sigma_k, V_k, E, Z$
2: **Output:** $\overline{U}_k \approx \widehat{U}_k, \overline{\Sigma}_k \approx \widehat{\Sigma}_k, \overline{V}_k \approx \widehat{V}_k$
3: Solve $[\Theta_k, F_k] = \texttt{svd}_k(Z^H A)$
4: Set $\overline{U}_k = ZF_k$ and $\overline{\Sigma}_k = \Theta_k$
5: Set $\overline{V}_k = A^H \overline{U}_k \overline{\Sigma}_k^{-1}$

Algorithm 2 RR-SVD ("$A^H A$" version).

1: **Input:** $B, U_k, \Sigma_k, V_k, E, Z$
2: **Output:** $\overline{U}_k \approx \widehat{U}_k, \overline{\Sigma}_k \approx \widehat{\Sigma}_k, \overline{V}_k \approx \widehat{V}_k$
3: Solve $[\Theta_k, G_k] = \texttt{svd}_k(Z^H A^H)$
4: Set $\overline{V}_k = ZG_k$ and $\overline{\Sigma}_k = \Theta_k$
5: Set $\overline{U}_k = A\overline{V}_k \overline{\Sigma}_k^{-1}$

Algorithm 1 can be adapted to approximate A_k for matrices of the form $A = \begin{pmatrix} B & E \end{pmatrix}$. The complete procedure is summarized in Algorithm 2. Note that by combining Algorithms 1 and 2 we can approximate the k leading singular triplets of matrices in which we add both new rows and columns.

Throughout the remainder of this paper we focus in updating the rank-k SVD of matrix $A = \begin{pmatrix} B \\ E \end{pmatrix}$ by Algorithm 1. The discussion extends trivially to updates of matrix $A = \begin{pmatrix} B & E \end{pmatrix}$ by Algorithm 2.

4 Building the projection matrix Z

The accuracy of Step 5 in Algorithm 1 depends on the accuracy of the approximate leading singular values and associated left singular vectors from Step 3. In turn, these quantities depend on how well $\texttt{range}(Z)$ captures the singular vectors $\widehat{u}^{(1)}, \ldots, \widehat{u}^{(k)}$ [14, 18]. Therefore, our focus lies in forming Z such that the distance between the subspace $\texttt{range}(Z)$ and the left singular

vectors $\widehat{u}^{(1)}, \ldots, \widehat{u}^{(k)}$ is as small as possible.

4.1 Exploiting the left singular vectors of B. The following proposition presents a closed-form expression of the i'th left singular vector of matrix $A = \begin{pmatrix} B \\ E \end{pmatrix}$.

PROPOSITION 4.1. *The left singular vector $\widehat{u}^{(i)}$ associated with singular value $\widehat{\sigma}_i$ is equal to*

$$\widehat{u}^{(i)} = \begin{pmatrix} -(BB^H - \widehat{\sigma}_i^2 I_m)^{-1} B E^H \widehat{y}^{(i)} \\ \widehat{y}^{(i)} \end{pmatrix},$$

where $\widehat{y}^{(i)}$ satisfies the equation

$$\left[E \left(\sum_{j=1}^{n} v^{(j)} \left(v^{(j)}\right)^H \frac{\widehat{\sigma}_i^2}{\widehat{\sigma}_i^2 - \sigma_j^2} \right) E^H - \widehat{\sigma}_i^2 I_s \right] \widehat{y}^{(i)} = 0,$$

and $\sigma_j = 0$ for any $j = m+1, \ldots, n$ (when $n > m$).

Proof. Deferred to the Supplementary Material. □

The above representation of $\widehat{u}^{(i)}$ requires the solution of a nonlinear eigenvalue problem to compute $\widehat{y}^{(i)}$. Alternatively, we can express $\widehat{u}^{(i)}$ as follows.

PROPOSITION 4.2. *The left singular vector $\widehat{u}^{(i)}$ associated with singular value $\widehat{\sigma}_i$ is equal to*

$$\widehat{u}^{(i)} = \begin{pmatrix} u^{(1)}, \ldots, u^{(\mathtt{min}(m,n))} & \\ & I_s \end{pmatrix} \begin{pmatrix} \chi_{1,i} \\ \vdots \\ \chi_{\mathtt{min}(m,n),i} \\ \widehat{y}^{(i)} \end{pmatrix},$$

where the scalars $\chi_{j,i}$ are equal to

$$\chi_{j,i} = -\left(E v^{(j)}\right)^H \widehat{y}^{(i)} \frac{\sigma_j}{\sigma_j^2 - \widehat{\sigma}_i^2}.$$

Proof. Deferred to the Supplementary Material. □

Proposition 4.2 suggests that setting $Z = \begin{pmatrix} u^{(1)}, \ldots, u^{(\mathtt{min}(m,n))} & \\ & I_s \end{pmatrix}$ should lead to an exact (in the absence of round-off errors) computation of $\widehat{u}^{(i)}$. In practice, we only have access to the k leading left singular vectors of B, $u^{(1)}, \ldots, u^{(k)}$. The following proposition suggests that the distance between $\widehat{u}^{(i)}$ and the range space of $Z = \begin{pmatrix} u^{(1)}, \ldots, u^{(k)} & \\ & I_s \end{pmatrix}$ is at worst proportional to the ratio $\dfrac{\sigma_{k+1}}{\sigma_{k+1}^2 - \widehat{\sigma}_i^2}$.

PROPOSITION 4.3. *Let matrix Z in Algorithm 1 be defined as*

$$Z = \begin{pmatrix} u^{(1)}, \ldots, u^{(k)} & \\ & I_s \end{pmatrix},$$

and set $\gamma = O\left(\left\|E^H \widehat{y}^{(i)}\right\|\right)$.

Then, for any $i = 1, \ldots, k$:

$$\mathtt{min}_{z \in \mathtt{range}(Z)} \|\widehat{u}^{(i)} - z\| \leq \left| \frac{\gamma \sigma_{k+1}}{\sigma_{k+1}^2 - \widehat{\sigma}_i^2} \right|.$$

Proof. Deferred to the Supplementary Material. □

Proposition 4.3 implies that left singular vectors associated with larger singular values of A are likely to be approximated more accurately.

4.1.1 The structure of matrix $Z^H A$. Setting the projection matrix Z as in Proposition 4.3 gives

$$Z^H A = \left(V_k \Sigma_k \; E^H\right)^H.$$

Therefore, each Matrix-Vector (MV) product with matrix $Z^H A A^H Z$ requires two MV products with matrices Σ_k, V_k and E, for a total cost of about $4(nk + \mathtt{nnz}(E))$ FLOPs. Moreover, we have $\mathtt{nr}(Z^H) = s + k$, and thus a rough estimate of the cost of Step 3 in Algorithm 1 is $4(nk + \mathtt{nnz}(E))\delta + 2(s+k)\delta^2$ FLOPs.

4.2 Exploiting resolvent expansions. The choice of Z presented in Section 4.1 can compute the exact A_k provided that the rank of B is exactly k. Nonetheless, when the rank of B is larger than k and the singular values $\sigma_{k+1}, \ldots, \sigma_{\mathtt{min}(m,n)}$ are not small, the accuracy of the approximate A_k returned by Algorithm 1 might be poor. This section presents an approach to enhance the projection matrix Z.

Recall that the top part of $\widehat{u}^{(i)}$ is equal to $\widehat{f}^{(i)} = -(BB^H - \widehat{\sigma}_i^2 I_m)^{-1} B E^H \widehat{y}^{(i)}$. In practice, even if we knew the unknown quantities $\widehat{\sigma}_i^2$ and $\widehat{y}^{(i)}$, the application of matrix $(BB^H - \widehat{\sigma}_i^2 I_m)^{-1}$ for each $i = 1, \ldots, k$, is too costly. The idea presented in this section considers the approximation of $(BB^H - \widehat{\sigma}_i^2 I_m)^{-1}$, $i = 1, \ldots, k$, by $(BB^H - \lambda I_m)^{-1}$ for some fixed scalar $\lambda \in \mathbb{R}$.

LEMMA 4.1. *Let*

$$B(\lambda) = (I_m - U_k U_k^H)(BB^H - \lambda I_m)^{-1}$$

such that $\lambda > \widehat{\sigma}_k^2$. Then, we have that for any $i = 1, \ldots, k$:

$$B(\widehat{\sigma}_i^2) = B(\lambda) \sum_{\rho=0}^{\infty} \left[(\widehat{\sigma}_i^2 - \lambda) B(\lambda)\right]^{\rho}.$$

Proof. Deferred to the Supplementary Material. □

Clearly, the closer λ is to $\widehat{\sigma}_i^2$, the more accurate the approximation in Lemma 4.1 should be. We can now provide an expression for $\widehat{u}^{(i)}$ similar to that in Proposition 4.2.

PROPOSITION 4.4. *The left singular vector $\widehat{u}^{(i)}$ associated with singular value $\widehat{\sigma}_i$ is equal to*

$$\widehat{u}^{(i)} = \begin{pmatrix} u^{(1)}, \ldots, u^{(k)} & \\ & I_s \end{pmatrix} \begin{pmatrix} \chi_{1,i} \\ \vdots \\ \chi_{k,i} \\ \widehat{y}^{(i)} \end{pmatrix} - \begin{pmatrix} B(\lambda) \sum_{\rho=0}^{\infty} \left[(\widehat{\sigma}_i^2 - \lambda) B(\lambda)\right]^{\rho} BE^H \widehat{y}^{(i)} \\ \\ \end{pmatrix}.$$

Proof. Deferred to the Supplementary Material. □

Proposition 4.4 suggests a way to enhance the projection matrix Z shown in Proposition 4.3. For example, we can approximate $B(\lambda) \sum_{\rho=0}^{\infty} \left[(\widehat{\sigma}_i^2 - \lambda) B(\lambda)\right]^{\rho}$ by $B(\lambda)$, which gives the following bound for the distance of $\widehat{u}^{(i)}$ from $\texttt{range}(Z)$.

PROPOSITION 4.5. *Let matrix Z in Algorithm 1 be defined as*

$$Z = \begin{pmatrix} u^{(1)}, \ldots, u^{(k)} & -B(\lambda) BE^H \\ & I_s \end{pmatrix}$$

and set $\gamma = O\left(\left\|E^H \widehat{y}^{(i)}\right\|\right)$.

Then, for any $\lambda \geq \widehat{\sigma}_1^2$ and $i = 1, \ldots, k$:

$$\texttt{min}_{z \in \texttt{range}(Z)} \|\widehat{u}^{(i)} - z\| \leq \left| \frac{\gamma \sigma_{k+1} (\widehat{\sigma}_i^2 - \lambda)}{(\sigma_{k+1}^2 - \widehat{\sigma}_i^2)(\sigma_{k+1}^2 - \lambda)} \right|.$$

Proof. Deferred to the Supplementary Material. □

Compared to the bound shown in Proposition 4.3, the bound in Proposition 4.5 is multiplied by $\frac{\widehat{\sigma}_i^2 - \lambda}{\sigma_{k+1}^2 - \lambda}$. In practice, due to cost considerations, we choose a single value of λ that is more likely to satisfy the above consideration, e.g., $\lambda \geq \widehat{\sigma}_1$.

4.2.1 Computing the matrix $B(\lambda)BE^H$. The construction of matrix Z shown in Lemma 4.5 requires the computation of the matrix $-B(\lambda)BE^H$. The latter is equal to the matrix X that satisfies the equation

$$-(BB^H - \lambda I_m)X = (I_m - U_k U_k^H)BE^H. \tag{4.1}$$

The eigenvalues of the matrix $-(BB^H - \lambda I_m)$ are equal to $\{\lambda - \widehat{\sigma}_i^2\}_{i=1,\ldots,m}$, and for any $\lambda > \widehat{\sigma}_1^2$, the matrix $-(BB^H - \lambda I_m)$ is positive definite. It is thus possible to compute X by repeated applications of the Conjugate Gradient method.

PROPOSITION 4.6. *Let $K = -(BB^H - \lambda I_m)$ and $\|e_j\|_K$ denote the K-norm of the error after j iterations of the Conjugate Gradient method applied to the linear system $-(BB^H - \lambda I_m)x = b$, where $b \in \texttt{range}\left((I_m - U_k U_k^H)BE^H\right)$. Then,*

$$\|e_j\|_K \leq 2 \left(\frac{\sqrt{\kappa} - 1}{\sqrt{\kappa} + 1}\right)^j \|e_0\|_K,$$

where $\kappa = \frac{\sigma_{\texttt{min}(m,n)}^2 - \lambda}{\sigma_{k+1}^2 - \lambda}$ and $\lambda > \widehat{\sigma}_1^2$.

Proof. Since $b \in \texttt{range}((I_m - U_k U_k^H)BE^H)$, the vector x satisfies the equation

$$-(I_m - U_k U_k^H)(BB^H - \lambda I_m)(I_m - U_k U_k^H)x = b. \tag{4.2}$$

The proof can then be found in [22]. □

COROLLARY 4.1. *The effective condition number satisfies the inequality $\kappa \leq \frac{\lambda}{\lambda - \sigma_{k+1}^2}$.*

Proposition 4.6 applies to each one of the s right-hand sides in (4.1). Assuming that the matrix $(I_m - U_k U_k^H)BE^H$ can be formed and stored, the effective condition number can be reduced even further. For example, solving (4.1) by the block Conjugate Gradient method leads to an effective condition number $\kappa = \frac{\sigma_{\texttt{min}(m,n)}^2 - \lambda}{\sigma_{k+s+1}^2 - \lambda}$ [20]. Additional techniques to solve linear systems with multiple right-hand sides can be found in [15, 16, 24].

Finally, notice that as λ increases, the effective condition number decreases. Thus from a convergence viewpoint, it is better to choose $\lambda \gg \widehat{\sigma}_i^2$. On the other hand, increasing λ leads to worse bounds in Proposition 4.5.

4.3 Truncating the matrix $B(\lambda)BE^H$. When the number of right-hand sides in (4.1), i.e., number of rows in matrix E, is too large, an alternative is to consider $-B(\lambda)BE^H \approx X_{\lambda,r} S_{\lambda,r} Y_{\lambda,r}^H$, where $X_{\lambda,r} S_{\lambda,r} Y_{\lambda,r}^H$ denotes the rank-r truncated SVD of matrix $-B(\lambda)BE^H$. We can then replace $-B(\lambda)BE^H$ by $X_{\lambda,r}$, since $\texttt{range}\left(X_{\lambda,r} S_{\lambda,r} Y_{\lambda,r}^H\right) \subseteq \texttt{range}\left(X_{\lambda,r}\right)$.

The matrix $X_{\lambda,r}$ can be approximated in a matrix-free fashion by applying a few iterations of Lanczos

bidiagonalization to matrix $B(\lambda)BE^H$. Each iteration requires two applications of Conjugate Gradient to solve linear systems of the same form as in (4.2). A second approach is to apply randomized SVD as described in [10, 5]. In practice, this amounts to computing the SVD of the matrix $B(\lambda)BE^HEB^HB(\lambda)R$ where R is a real matrix with at least r columns whose entries are i.i.d. Gaussian random variables of zero mean and unit variance.

4.3.1 The structure of matrix Z^HA. Setting the basis matrix Z as in Proposition 4.5 leads to

$$Z^HA = \left(V_k\Sigma_k \; B^HX_{\lambda,r} \; E^H\right)^H.$$

Each MV product with matrix Z^HAA^HZ then requires two MV products with matrices Σ_k, V_k, E and $B^HX_{\lambda,r}$, for a total cost of $4(n(k+r)+\texttt{nnz}(E))$. Moreover, we have $\texttt{nr}(Z^H) = k+r+s$, and thus a rough estimate of the cost of Step 3 in Algorithm 1 is $4(n(k+r)+\texttt{nnz}(E))\delta + 2(s+k+r)\delta^2$ FLOPs.

5 Evaluation

Our experiments were conducted in a Matlab environment (version R2020a), using 64-bit arithmetic, on a single core of a computing system equipped with an Intel Haswell E5-2680v3 processor and 32 GB of system memory.

Table 2: *Properties of the test matrices used throughout this section.*

Matrix	rows	columns	$nnz(A)$/rows	Source
MED	5,735	1,033	8.9	[1]
CRAN	4,563	1,398	17.8	[1]
CISI	5,544	1,460	12.2	[1]
ML1M	6,040	3,952	165.6	[11]

Table 2 lists the test matrices considered throughout our experiments along with their dimensions and source from which they were retrieved. The first three matrices come from LSI applications and represent term-document matrices, while the last matrix comes from recommender systems and represents a user-item rating matrix. The $k = 100$ leading singular values of each matrix listed in Table 2 are plotted in Figure 1.

Throughout this section we focus on accuracy and will be reporting: a) the relative error in the approximation of the k leading singular values of A, and b) the norm of the residual $A\widehat{v}^{(i)} - \widehat{\sigma}_i\widehat{u}^{(i)}$, scaled by $\widehat{\sigma}_i$. The scalar λ is set as $\lambda = 1.01\widehat{\sigma}_1^2$ where the latter singular value is approximated by a few iterations of Lanczos bidiagonalization.

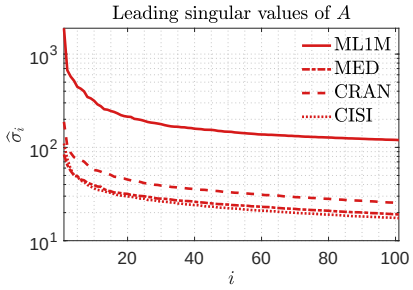

Figure 1: *Leading $k = 100$ singular values.*

5.1 Single update. In this section we consider the approximation of the $k = 50$ leading singular triplets of $A = \begin{pmatrix} B \\ E \end{pmatrix}$ where $B = A(1 : \lceil m/2 \rceil, :)$, i.e., the size of matrices B and E is about half the size of A. We run Algorithm 1 and set Z as in Propositions 4.3 and 4.5. For the enhanced matrix Z, the matrix $X_{\lambda,r}$ is computed by randomized SVD where $r = k$ and the number of columns in matrix R is equal to $2k$ (recall the discussion in Section 4.3). The associated linear system with $2k$ right-hand sides is solved by block Conjugate Gradient.

Figure 2 plots the relative error and residual norm in the approximation of the $k = 50$ leading singular triplets of A. As expected, enhancing the projection matrix Z by $X_{\lambda,r}$ leads to higher accuracy. This is especially true for the approximation of those singular triplets with corresponding singular values $\widehat{\sigma}_i \approx \lambda$.

In all of our experiments, the worst-case (maximum) relative error and residual norm was achieved in the approximation of the singular triplet $(\hat{\sigma}_{50}, \hat{u}^{(50)}, \hat{v}^{(50)})$. Table 3 lists the relative error and residual norm associated with the approximation of the singular triplet $(\hat{\sigma}_{50}, \hat{u}^{(50)}, \hat{v}^{(50)})$ as r varies from ten to fifty in increments of ten. As a reference, we list the same quantity for the case $Z = \begin{pmatrix} U_k \\ & I_s \end{pmatrix}$. As expected, enhancing the projection matrix Z by $X_{\lambda,r}$ leads to higher accuracy, especially for higher values of r.

5.2 Sequence of updates. In this experiment the rows of matrix E are now added in batches, i.e., we first approximate the k leading singular triplets of matrix $A^{(0)} = \begin{pmatrix} B \\ A(\lceil m/2 \rceil + 1 : \lceil m/2 \rceil + t, :) \end{pmatrix}$, then of matrix $A^{(1)} = \begin{pmatrix} A^{(0)} \\ A(\lceil m/2 \rceil + t + 1 : \lceil m/2 \rceil + 2t, :) \end{pmatrix}$, etc. Here, $t = \lceil m/2 \rceil / \phi$ denotes the step-size and $\phi \in \mathbb{Z}^*$ denotes

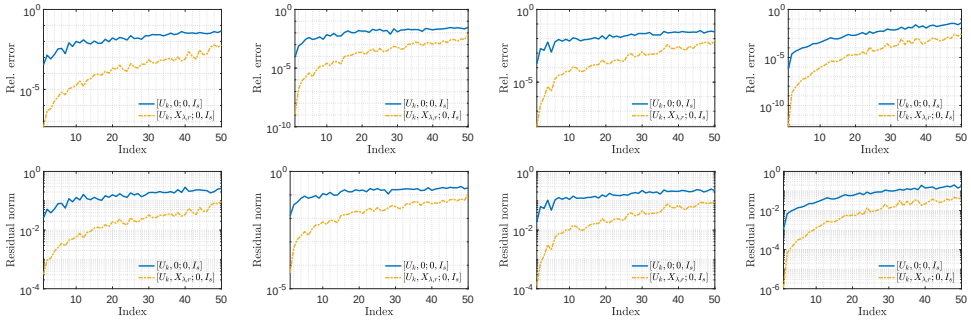

Figure 2: *Approximation of the leading $k = 50$ singular triplets for the single update case. From left to right: MED, CRAN, CISI, and 1M.*

Table 3: *Relative error and residual norm associated with the approximation of the singular triplet $(\widehat{\sigma}_{50}, \widehat{u}^{(50)}, \widehat{v}^{(50)})$.*

		MED		**CRAN**		**CISI**		**ML1M**	
	r	err.	res.	err.	res.	err.	res.	err.	res.
$Z = \begin{pmatrix} U_k & X_{\lambda,r} \\ & I_s \end{pmatrix}$	$r = 10$	0.036	0.234	0.026	0.176	0.025	0.214	0.031	0.156
	$r = 20$	0.031	0.184	0.021	0.155	0.023	0.189	0.012	0.143
	$r = 30$	0.021	0.114	0.017	0.134	0.017	0.161	0.008	0.121
	$r = 40$	0.009	0.091	0.013	0.111	0.012	0.134	0.005	0.112
	$r = 50$	0.004	0.053	0.007	0.098	0.007	0.081	0.003	0.076
$Z = \begin{pmatrix} U_k \\ & I_s \end{pmatrix}$	N/A	0.045	0.269	0.045	0.199	0.287	0.250	0.041	0.173

the total number of updates. Note that after the first update, the matrices U_k and V_k no longer denote the exact k leading left and right singular vectors of the $B \equiv A^{(j-1)}$ submatrix of matrix $A^{(j)}$. We set $\phi = 12$ and plot the accuracy achieved after one, six, and twelve updates, in Figures 3 and 4. Notice that enhancing Z by $X_{\lambda,r}$ leads to similar accuracy for all updates, while in the opposite case accuracy deteriorates as the updates accumulate. On a separate note, the accuracy of the k leading singular triplets of A is higher when matrix E is added to B in batches rather than in a single update as in the previous section.

Table 4 lists relative error and residual norm associated with the approximation of the singular triplet $(\widehat{\sigma}_{50}, \widehat{u}^{(50)}, \widehat{v}^{(50)})$ by Algorithm 1 and the method in [31]. The number of sought singular triplets k was varied from ten to thirty. Comparisons against the method in [28] were also performed but not reported since the latter was always less accurate than [31]. Overall, Algorithm 1 provided higher accuracy, especially for those singular triplets whose corresponding singular value was closer to λ.

6 Conclusion

This paper presented an algorithm to update the rank-k truncated SVD of evolving matrices. The proposed algorithm undertakes a projection viewpoint and aims on building a pair of subspaces which approximate the linear span of the k leading singular vectors of the updated matrix. Two different options to set these subspaces were considered. Experiments performed on matrices stemming from applications in LSI and recommender systems verified the effectiveness of the proposed scheme in terms of accuracy.

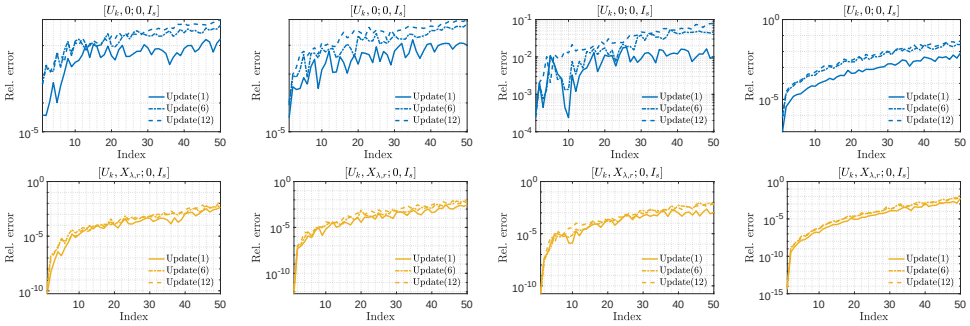

Figure 3: *Relative error in the approximation of the $k = 50$ leading singular values of A for the multiple updates case. From left to right: MED, CRAN, CISI, and 1M.*

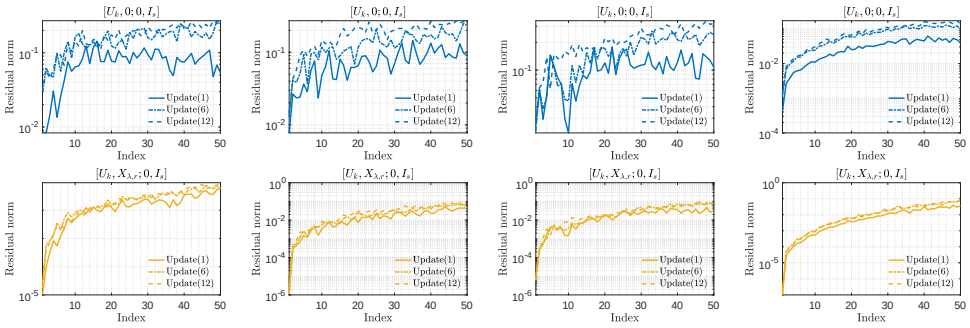

Figure 4: *Residual norm of the approximation of the $k = 50$ leading singular triplets of A for the multiple updates case. From left to right: MED, CRAN, CISI, and 1M.*

Table 4: *Maximum relative error and residual norm of the approximation of the k leading singular triplets of A for the multiple updates case as k varies.*

		MED		**CRAN**		**CISI**		**ML1M**	
	Method	err.	res.	err.	res.	err.	res.	err.	res.
$k = 10$	[31]	0.046	0.172	0.043	0.192	0.054	0.274	0.002	0.058
	Alg. 1	0.001	0.045	0.008	0.090	0.002	0.054	3.0e-5	0.007
$k = 20$	[31]	0.067	0.212	0.064	0.255	0.075	0.224	0.022	0.131
	Alg. 1	0.004	0.073	0.005	0.076	0.003	0.053	0.002	0.040
$k = 30$	[31]	0.076	0.384	0.060	0.290	0.084	0.330	0.023	0.123
	Alg. 1	0.006	0.067	0.008	0.088	0.004	0.070	0.001	0.041

References

[1] `http://web.eecs.utk.edu/research/lsi/`.

[2] J. Baglama and L. Reichel, *Augmented implicitly restarted Lanczos bidiagonalization methods*, SIAM Journal on Scientific Computing, 27 (2005), pp. 19–42.

[3] M. W. Berry, S. T. Dumais, and G. W. O'Brien,

Using linear algebra for intelligent information retrieval, SIAM Review, 37 (1995), pp. 573–595.

[4] M. Brand, *Fast online SVD revisions for lightweight recommender systems*, in Proceedings of the 2003 SIAM International Conference on Data Mining, SIAM, 2003, pp. 37–46.

[5] K. L. Clarkson and D. P. Woodruff, *Numerical linear algebra in the streaming model*, in Proceedings of the forty-first annual ACM symposium on Theory of computing, 2009, pp. 205–214.

[6] P. Cremonesi, Y. Koren, and R. Turrin, *Performance of recommender algorithms on top-n recommendation tasks*, in Proceedings of the fourth ACM conference on Recommender systems, 2010, pp. 39–46.

[7] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman, *Indexing by latent semantic analysis*, Journal of the American society for information science, 41 (1990), pp. 391–407.

[8] G. Golub and W. Kahan, *Calculating the singular values and pseudo-inverse of a matrix*, Journal of the Society for Industrial and Applied Mathematics, Series B: Numerical Analysis, 2 (1965), pp. 205–224.

[9] M. Gu, Stanley, and S. C. Eisenstat, *A stable and fast algorithm for updating the singular value decomposition*, tech. rep., 1994.

[10] N. Halko, P.-G. Martinsson, and J. A. Tropp, *Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions*, SIAM Review, 53 (2011), pp. 217–288.

[11] F. M. Harper and J. A. Konstan, *The movielens datasets: History and context*, ACM transactions on interactive intelligent systems, 5 (2015), pp. 1–19.

[12] V. Hernandez, J. E. Roman, and V. Vidal, *SLEPc: A scalable and flexible toolkit for the solution of eigenvalue problems*, ACM Transactions on Mathematical Software (TOMS), 31 (2005), pp. 351–362.

[13] L. Horesh, A. R. Conn, E. A. Jimenez, and G. M. van Essen, *Reduced space dynamics-based geostatistical prior sampling for uncertainty quantification of end goal decisions*, in Numerical Analysis and Optimization, Springer, 2015, pp. 191–221.

[14] Z. Jia and G. Stewart, *An analysis of the Rayleigh-Ritz method for approximating eigenspaces*, Mathematics of computation, 70 (2001), pp. 637–647.

[15] V. Kalantzis, C. Bekas, A. Curioni, and E. Gallopoulos, *Accelerating data uncertainty quantification by solving linear systems with multiple right-hand sides*, Numerical Algorithms, 62 (2013), pp. 637–653.

[16] V. Kalantzis, A. C. I. Malossi, C. Bekas, A. Curioni, E. Gallopoulos, and Y. Saad, *A scalable iterative dense linear system solver for multiple right-hand sides in data analytics*, Parallel Computing, 74 (2018), pp. 136–153.

[17] M. Moonen, P. Van Dooren, and J. Vandewalle, *A singular value decomposition updating algorithm for subspace tracking*, SIAM Journal on Matrix Analysis and Applications, 13 (1992), pp. 1015–1038.

[18] Y. Nakatsukasa, *Accuracy of singular vectors obtained by projection-based SVD methods*, BIT Numerical Mathematics, 57 (2017), pp. 1137–1152.

[19] A. N. Nikolakopoulos, V. Kalantzis, E. Gallopoulos, and J. D. Garofalakis, *Eigenrec: generalizing PureSVD for effective and efficient top-N recommendations*, Knowledge and Information Systems, 58 (2019), pp. 59–81.

[20] D. P. O'Leary, *The block Conjugate Gradient algorithm and related methods*, Linear Algebra and its Applications, 29 (1980), pp. 293–322.

[21] Y. Saad, *Numerical methods for large eigenvalue problems: revised edition*, SIAM, 2011.

[22] Y. Saad, M. Yeung, J. Erhel, and F. Guyomarc'h, *A deflated version of the Conjugate Gradient algorithm*, SIAM Journal on Scientific Computing, 21 (2000), pp. 1909–1926.

[23] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, *Incremental singular value decomposition algorithms for highly scalable recommender systems*, in Fifth international conference on computer and information science, vol. 1, Citeseer, 2002, pp. 27–8.

[24] A. Stathopoulos and K. Orginos, *Computing and deflating eigenvalues while solving multiple right-hand side linear systems with an application to quantum chromodynamics*, SIAM Journal on Scientific Computing, 32 (2010), pp. 439–462.

[25] S. Ubaru, A. Mazumdar, and Y. Saad, *Low rank approximation using error correcting coding matrices*, in International Conference on Machine Learning, 2015, pp. 702–710.

[26] S. Ubaru and Y. Saad, *Sampling and multilevel coarsening algorithms for fast matrix approximations*, Numerical Linear Algebra with Applications, 26 (2019), p. e2234.

[27] S. Ubaru, A.-K. Seghouane, and Y. Saad, *Find the dimension that counts: Fast dimension estimation and Krylov PCA*, in Proceedings of the 2019 SIAM International Conference on Data Mining, SIAM, 2019, pp. 720–728.

[28] E. Vecharynski and Y. Saad, *Fast updating algorithms for latent semantic indexing*, SIAM Journal on Matrix Analysis and Applications, 35 (2014), pp. 1105–1131.

[29] L. Wu and A. Stathopoulos, *A preconditioned hybrid SVD method for accurately computing singular triplets of large matrices*, SIAM Journal on Scientific Computing, 37 (2015), pp. S365–S388.

[30] I. Yamazaki, S. Tomov, and J. Dongarra, *Sampling algorithms to update truncated SVD*, in 2017 IEEE International Conference on Big Data, IEEE, 2017, pp. 817–826.

[31] H. Zha and H. D. Simon, *On updating problems in latent semantic indexing*, SIAM Journal on Scientific Computing, 21 (1999), pp. 782–791.

[32] H. Zha and Z. Zhang, *Matrices with low-rank-plus-shift structure: partial SVD and latent semantic indexing*, SIAM Journal on Matrix Analysis and Applications, 21 (2000), pp. 522–536.

Supplementary Material

Asymptotic complexity

The asymptotic complexity analysis of the method in [31] is as follows. We need $O\left(ns^2 + nsk\right)$ FLOPs to form $(I_s - V_k V_k^H)E^H$ and compute its QR decomposition. The SVD of the matrix $Z^H AW$ requires $O\left((k+s)^3\right)$ FLOPs. Finally, the cost to form the approximation of matrices $\widehat{U}_k$ and $\widehat{V}_k$ is equal to $O\left(k^2(m+n) + nsk\right)$ FLOPs.

The asymptotic complexity analysis for the "SV" variant of the method in [28] is as follows. We need $O\left((\texttt{nnz}(E) + nk)\delta_1 + (n+s)\delta_1^2\right)$ FLOPs to approximate the r leading singular triplets of $(I_s - V_k V_k^H)E^H$, where $\delta_1 \in \mathbb{Z}^*$ is greater than or equal to r (i.e., δ_1 is the number of Lanczos bidiagonalization steps). The cost to form and compute the SVD of the matrix $Z^H AW$ is equal to $(k+s)(k+r)^2 + \texttt{nnz}(E)k + rs$ where the first term stands for the actual SVD and the rest of the terms stand for the formation of the matrix $Z^H AW$. Finally, the cost to form the approximation of matrices $\widehat{U}_k$ and $\widehat{V}_k$ is equal to $O\left(k^2(m+n) + nrk\right)$ FLOPs.

The asymptotic complexity analysis of Algorithm 1 is as follows. First, notice that Algorithm 1 requires no effort to build W. For the case where Z is set as in Proposition 4.3, termed as "Alg. 1 (a)", we also need no FLOPs to build Z. The cost to solve the projected problem by unrestarted Lanczos is then equal to $O\left((\texttt{nnz}(E) + nk)\delta_2 + (k+s)\delta_2^2\right)$ FLOPs, where $\delta_2 \in \mathbb{Z}^*$ is greater than or equal to k (i.e., δ_2 is the number of unrestarted Lanczos steps). Finally, the cost to form the approximation of matrices $\widehat{U}_k$ and $\widehat{V}_k$ is equal to $O\left(k^2 m + (\texttt{nnz}(A) + n)k\right)$ FLOPs. For the case where Z is set as in Proposition 4.5, termed as "Alg. 1 (b)", we need

$$\chi = O\left(\texttt{nnz}(A)\delta_3 + m\delta_3^2\right)$$

FLOPs to build $X_{\lambda,r}$, where $\delta_3 \in \mathbb{Z}^*$ is greater than or equal to k (i.e., δ_3 is either the number of Lanczos bidiagonalization steps or the number of columns of matrix R in randomized SVD).

Table 5: *Detailed asymptotic complexity of Algorithm 1 and the schemes in [31] and [28]. All δ variables are replaced by k.*

Scheme	Building Z	Building W	Solving the projected problem	Other
[31]	-	$ns^2 + nsk$	$(k+s)^3$	$k^2(m+n) + nsk$
[28]	-	$(\texttt{nnz}(E) + nk)k + (n+s)k^2$	$(k+s)(k+r)^2 + \texttt{nnz}(E)k + rs$	$k^2(m+n) + nrk$
Alg. 1 (a)	-	-	$(\texttt{nnz}(E) + nk)k + (k+s)k^2$	$k^2 m + (\texttt{nnz}(A) + n)k$
Alg. 1 (b)	χ	-	$(\texttt{nnz}(E) + (n+r)k)k + (k+r+s)k^2$	$k^2 m + (\texttt{nnz}(A) + n)k$

The above discussion is summarized in Table 5 where we list the asymptotic complexity of Algorithm 1 and the schemes in [31] and [28]. The complexities of the latter two schemes were also verified by adjusting the complexity analysis from [28]. To allow for a practical comparison, we replaced all δ variables with k since in practice these variables are equal to at most a small integer multiple of k.

Consider now a comparison between Algorithm 1 (a) and the method in [31]. For all practical purposes, these two schemes return identical approximations to A_k. Nonetheless, Algorithm 1 (a) requires no effort to build W. Moreover, the cost to solve the projected problem is linear with respect to s and cubic with respect to k, instead of cubic with respect to the sum $s+k$ in [31]. The only scenario where Algorithm 1 can be potentially more expensive than [31] is when matrix A is exceptionally dense, and both k and s are very small. Similar observations can be made for the relation between Algorithm 1 (b) and the methods in [28], although the comparison is more involved.

Proofs

Proof of Proposition 4.1 The scalar-vector pair $(\widehat{\sigma}_i^2, \widehat{u}^{(i)})$ satisfies the equation $(AA^H - \widehat{\sigma}_i^2 I_{m+s})\widehat{u}^{(i)} = 0$. If we partition the i'th left singular vector as

$$\widehat{u}^{(i)} = \begin{pmatrix} \widehat{f}^{(i)} \\ \widehat{y}^{(i)} \end{pmatrix},$$

we can write

$$\begin{pmatrix} BB^H - \widehat{\sigma}_i^2 I_m & BE^H \\ EB^H & EE^H - \widehat{\sigma}_i^2 I_s \end{pmatrix} \begin{pmatrix} \widehat{f}^{(i)} \\ \widehat{y}^{(i)} \end{pmatrix} = 0.$$

The leading m rows satisfy $(BB^H - \widehat{\sigma}_i^2 I_m)\widehat{f}^{(i)} = -BE^H \widehat{y}^{(i)}$. Plugging the expression of $\widehat{f}^{(i)}$ in the second block of rows and considering the full SVD $B = U\Sigma V^H$ leads to

$$\begin{aligned} 0 &= \left[EE^H - EB^H(BB^H - \widehat{\sigma}_i^2 I_m)^{-1} BE^H - \widehat{\sigma}_i^2 I_s\right] \widehat{y}^{(i)} \\ &= \left[E(I_s - B^H(BB^H - \widehat{\sigma}_i^2 I_m)^{-1} B)E^H - \widehat{\sigma}_i^2 I_s\right] \widehat{y}^{(i)} \\ &= \left[E(VV^H + V\Sigma^T(\widehat{\sigma}_i^2 I_m - \Sigma\Sigma^T)^{-1}\Sigma V^H)E^H - \widehat{\sigma}_i^2 I_s\right] \widehat{y}^{(i)} \\ &= \left[EV(I_n + \Sigma^T \left(\widehat{\sigma}_i^2 I_m - \Sigma\Sigma^T\right)^{-1} \Sigma)V^H E^H - \widehat{\sigma}_i^2 I_s\right] \widehat{y}^{(i)}. \end{aligned}$$

The proof concludes by noticing that

$$I_n + \Sigma^T \left(\widehat{\sigma}_i^2 I_m - \Sigma\Sigma^T\right)^{-1} \Sigma = \begin{pmatrix} 1 + \dfrac{\sigma_1^2}{\widehat{\sigma}_i^2 - \sigma_1^2} & & \\ & \ddots & \\ & & 1 + \dfrac{\sigma_n^2}{\widehat{\sigma}_i^2 - \sigma_n^2} \end{pmatrix} = \begin{pmatrix} \dfrac{\widehat{\sigma}_i^2}{\widehat{\sigma}_i^2 - \sigma_1^2} & & \\ & \ddots & \\ & & \dfrac{\widehat{\sigma}_i^2}{\widehat{\sigma}_i^2 - \sigma_n^2} \end{pmatrix},$$

where for the case $m < n$, we have $\sigma_j = 0$ for any $j = m+1, \ldots, n$. In case $\widehat{\sigma}_i = \sigma_j$, the Moore-Penrose pseudoinverse $(BB^H - \widehat{\sigma}_i^2 I_m)^\dagger$ is considered instead.

Proof of Proposition 4.2 Since the left singular vectors of B span $\mathbb{R}^m$, we can write

$$BE^H \widehat{y}^{(i)} = \sum_{j=1}^{m} \sigma_j u^{(j)} \left(Ev^{(j)}\right)^H \widehat{y}^{(i)}.$$

The proof concludes by noticing that the top $m \times 1$ part of $\widehat{u}^{(i)}$ can be written as

$$\begin{aligned} \widehat{f}^{(i)} &= -(BB^H - \widehat{\sigma}_i^2 I_m)^{-1} BE^H \widehat{y}^{(i)} \\ &= -U(\Sigma\Sigma^T - \widehat{\sigma}_i^2 I_m)^{-1} \Sigma \left(EV\right)^H \widehat{y}^{(i)} \\ &= -\sum_{j=1}^{\min(m,n)} u^{(j)} \frac{\sigma_j}{\sigma_j^2 - \widehat{\sigma}_i^2} \left(Ev^{(j)}\right)^H \widehat{y}^{(i)} \\ &= -\sum_{j=1}^{\min(m,n)} u^{(j)} \frac{\sigma_j}{\sigma_j^2 - \widehat{\sigma}_i^2} \left(Ev^{(j)}\right)^H \widehat{y}^{(i)} \\ &= \sum_{j=1}^{\min(m,n)} u^{(j)} \chi_{j,i}. \end{aligned}$$

Proof of Proposition 4.3 We have

$$\begin{aligned}
\min_{z\in\mathrm{range}(Z)}\|\widehat{u}^{(i)}-z\| &\le \left\| \begin{pmatrix} u^{(k+1)},\dots,u^{(\min(m,n))} \end{pmatrix} \begin{pmatrix} \chi_{k+1,i} \\ \vdots \\ \chi_{\min(m,n),i} \end{pmatrix} \right\| \\
&= \left\| \begin{pmatrix} 0_{k,k} & & & \\ & \frac{\sigma_{k+1}}{\sigma_{k+1}^2-\widehat{\sigma}_i^2} & & \\ & & \ddots & \\ & & & \frac{\sigma_{\min(m,n)}}{\sigma_{\min(m,n)}^2-\widehat{\sigma}_i^2} \end{pmatrix} V^H E^H \hat{y}^{(i)} \right\| \\
&\le \max\left\{ \left| \frac{\sigma_j}{\sigma_j^2-\widehat{\sigma}_i^2} \right| \right\}_{j=k+1,\dots,\min(m,n)} \left\| E^H \widehat{y}^{(i)} \right\|.
\end{aligned}$$

The proof follows by noticing that due to Cauchy's interlacing theorem we have $\sigma_{k+1}^2 \le \widehat{\sigma}_i^2,\ i=1,\dots,k$, and thus $\left|\frac{\sigma_{k+1}}{\sigma_{k+1}^2-\widehat{\sigma}_i^2}\right| \ge \cdots \ge \left|\frac{\sigma_{\min(m,n)}}{\sigma_{\min(m,n)}^2-\widehat{\sigma}_i^2}\right|$.

Proof of Lemma 4.1 We can write

$$\begin{aligned}
B(\lambda) &= \left(I - U_k U_k^H\right) U \begin{pmatrix} \sigma_1^2-\lambda & & \\ & \ddots & \\ & & \sigma_m^2-\lambda \end{pmatrix}^{-1} U^H \\
&= U \begin{pmatrix} 0_{k,k} & & & \\ & \frac{1}{\sigma_{k+1}^2-\lambda} & & \\ & & \ddots & \\ & & & \frac{1}{\sigma_m^2-\lambda} \end{pmatrix} U^H,
\end{aligned}$$

where $\sigma_j = 0$ for any $j > \min(m,n)$. Let us now define the scalar $\gamma_{j,i} = \dfrac{\widehat{\sigma}_i^2-\lambda}{\sigma_j^2-\lambda}$. Then,

$$B(\lambda)\left[(\widehat{\sigma}_i^2-\lambda)B(\lambda)\right]^\rho = U \begin{pmatrix} 0_{k,k} & & & \\ & \frac{\gamma_{k+1,i}^\rho}{\sigma_{k+1}^2-\lambda} & & \\ & & \ddots & \\ & & & \frac{\gamma_{m,i}^\rho}{\sigma_m^2-\lambda} \end{pmatrix} U^H.$$

Accounting for all powers $p = 0, 1, 2, \dots$, gives

$$B(\lambda)\sum_{\rho=0}^{\infty}\left[(\widehat{\sigma}_i^2-\lambda)B(\lambda)\right]^\rho = U \begin{pmatrix} 0_{k,k} & & & \\ & \frac{\sum_{\rho=0}^\infty \gamma_{k+1,i}^\rho}{\sigma_{k+1}^2-\lambda} & & \\ & & \ddots & \\ & & & \frac{\sum_{\rho=0}^\infty \gamma_{m,i}^\rho}{\sigma_m^2-\lambda} \end{pmatrix} U^H.$$

Since $\lambda > \widehat{\sigma}_k^2 \geq \sigma_k^2$, it follows that for any $j > k$ we have $|\gamma_{j,i}| < 1$. Therefore, the geometric series converges and $\sum_{\rho=0}^{\infty} \gamma_{j,i}^{\rho} = \dfrac{1}{1-\gamma_{j,i}} = \dfrac{\sigma_j^2 - \lambda}{\sigma_j^2 - \widehat{\sigma}_i^2}$. It follows that $\dfrac{1}{\sigma_j^2 - \lambda} \sum_{\rho=0}^{\infty} \gamma_j^{\rho} = \dfrac{1}{\sigma_j^2 - \widehat{\sigma}_i^2}$.

We finally have

$$\begin{aligned} B(\lambda) \sum_{\rho=0}^{\infty} \left[(\widehat{\sigma}_i^2 - \lambda) B(\lambda) \right]^{\rho} &= U \begin{pmatrix} 0_{k,k} & & & \\ & \dfrac{1}{\sigma_{k+1}^2 - \widehat{\sigma}_i^2} & & \\ & & \ddots & \\ & & & \dfrac{1}{\sigma_m^2 - \widehat{\sigma}_i^2} \end{pmatrix} U^H \\ &= \left(I - U_k U_k^H \right) B(\widehat{\sigma}_i^2). \end{aligned}$$

This concludes the proof.

Proof of Proposition 4.4 First, notice that

$$(BB^H - \widehat{\sigma}_i^2 I_m)^{-1} = U_k U_k^H (BB^H - \widehat{\sigma}_i^2 I_m)^{-1} + (I_m - U_k U_k^H)(BB^H - \widehat{\sigma}_i^2 I_m)^{-1}.$$

Therefore, we can write

$$(BB^H - \widehat{\sigma}_i^2 I_m)^{-1} BE^H \widehat{y}^{(i)} = U_k (\Sigma_k^2 - \widehat{\sigma}_i^2 I_k)^{-1} \Sigma_k (EV_k)^H \widehat{y}^{(i)} + (I_m - U_k U_k^H)(BB^H - \widehat{\sigma}_i^2 I_m)^{-1} BE^H \widehat{y}^{(i)}.$$

The left singular vector $\widehat{u}^{(i)}$ can be then expressed as

$$\begin{aligned} \widehat{u}^{(i)} &= \begin{pmatrix} -(BB^H - \widehat{\sigma}_i^2 I_m)^{-1} BE^H \\ I_s \end{pmatrix} \widehat{y}^{(i)} \\ &= \begin{pmatrix} u^{(1)}, \ldots, u^{(k)} & \\ & I_s \end{pmatrix} \begin{pmatrix} \chi_{1,i} \\ \vdots \\ \chi_{k,i} \\ \widehat{y}^{(i)} \end{pmatrix} - \begin{pmatrix} B(\widehat{\sigma}_i^2) BE^H \widehat{y}^{(i)} \\ \end{pmatrix}. \end{aligned}$$

The proof concludes by noticing that by Lemma 4.1 we have $B(\widehat{\sigma}_i^2) = B(\lambda) \sum\limits_{\rho=0}^{\infty} \left[(\widehat{\sigma}_i^2 - \lambda) B(\lambda) \right]^{\rho}$.

Proof of Proposition 4.5 The proof exploits the formula

$$(B(\widehat{\sigma}_i^2) - B(\lambda)) BE^H = (I - U_k U_k^H) U \left[(\Sigma\Sigma^T - \widehat{\sigma}_i^2 I_m)^{-1} - (\Sigma\Sigma^T - \lambda I_m)^{-1} \right] U^H U \Sigma V^H E^H.$$

It follows

$$\begin{aligned} \mathtt{min}_{z \in \mathtt{range}(Z)} \|\widehat{u}^{(i)} - z\| &\leq \left\| \begin{pmatrix} \left[B(\widehat{\sigma}_i^2) - B(\lambda) \right] BE^H \widehat{y}^{(i)} \\ \end{pmatrix} \right\| \\ &\leq \left\| \begin{pmatrix} 0_{k,k} & & & \\ & \dfrac{\sigma_{k+1}(\widehat{\sigma}_i^2 - \lambda)}{(\sigma_{k+1}^2 - \widehat{\sigma}_i^2)(\sigma_{k+1}^2 - \lambda)} & & \\ & & \ddots & \\ & & & \dfrac{\sigma_{\mathtt{min}(m,n)}(\widehat{\sigma}_i^2 - \lambda)}{(\sigma_{\mathtt{min}(m,n)}^2 - \widehat{\sigma}_i^2)\left(\sigma_{\mathtt{min}(m,n)}^2 - \lambda\right)} \end{pmatrix} \right\| \left\| E^H \widehat{y}^{(i)} \right\| \\ &\leq \mathtt{max} \left\{ \left| \frac{\sigma_j(\widehat{\sigma}_i^2 - \lambda)}{(\sigma_j^2 - \widehat{\sigma}_i^2)(\sigma_j^2 - \lambda)} \right| \right\}_{j=k+1,\ldots,\mathtt{min}(m,n)} \left\| E^H \widehat{y}^{(i)} \right\|. \end{aligned}$$