Turkish Journal of Mathematics

http://journals.tubitak.gov.tr/math/

Research Article

Turk J Math
(2018) 42: 1779 – 1794

doi:10.3906/mat-1707-14

Block classical Gram–Schmidt-based block updating in low-rank matrix approximation

Hasan ERBAY[1,*], **Fatih VARÇIN**[1], **Fahrettin HORASAN**[1], **Cenker BİÇER**[2]

[1]Department of Computer Engineering, Faculty of Engineering, Kırıkkale University, Yahşihan, Kırıkkale, Turkey

[2]Department of Statistics, Faculty of Arts & Science, Kırıkkale University, Yahşihan, Kırıkkale, Turkey

Received: 06.07.2017 • **Accepted/Published Online:** 28.03.2018 • **Final Version:** 24.07.2018

Abstract: Low-rank matrix approximations have recently gained broad popularity in scientific computing areas. They are used to extract correlations and remove noise from matrix-structured data with limited loss of information. Truncated singular value decomposition (SVD) is the main tool for computing low-rank approximation. However, in applications such as latent semantic indexing where document collections are dynamic over time, i.e. the term document matrix is subject to repeated updates, SVD becomes prohibitive due to the high computational expense. Alternative decompositions have been proposed for these applications such as low-rank ULV/URV decompositions and truncated ULV decomposition. Herein, we propose a BLAS-3 compatible block updating truncated ULV decomposition algorithm based on the block classical Gram–Schmidt process. The simulation results presented show that the block update algorithm is promising.

Key words: Truncated ULVD, block classical Gram–Schmidt, block update

1. Introduction

Low-rank matrix approximations have recently gained broad popularity in scientific computing areas such as information retrieval [5, 6, 8, 21], signal processing [9, 20, 23], web search [17, 18], and machine learning [11, 14]. They are used to extract correlations and remove noise from matrix-structured data with limited loss of information.

The low-rank approximation of a given matrix $X \in \mathbb{R}^{m\times n}$ and positive constant $k \ll \max(m,n)$ is the matrix X_k that satisfies

$$\min \|X - X_k\| \quad \text{subject to} \quad \operatorname{rank} X_k = k, \tag{1}$$

where $\|\cdots\|$ represents either two-norm or *Frobenius-norm*. The existence of such a matrix follows from the singular value decomposition (SVD) of X. Moreover, with no doubt, the truncated SVD is the main tool for computing the low-rank approximation. However, in applications such as latent semantic indexing where document collections are dynamic over time, i.e. the term document matrix is subject to repeated updates, SVD becomes prohibitive due to the high computational expense. Alternative decompositions have been proposed for these applications such as low-rank ULV/URV decompositions [12] and truncated ULV decomposition [2], but these are not suitable for block updates. In addition to updating, the initial costs of computing the low-rank

*Correspondence: hasan_erbay@yahoo.com

2010 *AMS Mathematics Subject Classification:* 65F05, 65F30, 65F50

ULV/URV decompositions and the truncated ULV decomposition are better than those of SVD [12].

Truncated ULV decomposition has been used to extract matrix information such as numerical rank and numerical subspaces, and especially numerical null space. It can also be used to solve block update problems: given the truncated ULV decomposition of X, find the truncated ULV decomposition of the matrix

$$\bar{X} = \begin{pmatrix} X \\ A^T \end{pmatrix}, \tag{2}$$

where $A \in \mathbb{R}^{n \times p}$ is the new arrival block matrix.

In this manuscript, we develop a Level-3 Basic Linear Algebra Subprograms (BLAS-3) [10] compatible block update algorithm. The algorithm is based on the block classical Gram–Schmidt algorithm [1, 3, 22], which is detailed in Section 2. Since the update algorithm is built upon matrix–matrix operation rather than matrix–vector operation, it makes effective use of caching to avoid excessive movement of data to/from the memory.

The rest of the manuscript is organized as follows. In Section 2 we introduce some notations, cover critical background materials in numerical linear algebra, and develop some matrix computational tools. In Section 3 we give the steps of the block update algorithm and show how the refinement algorithm in [2] may be used as a "clean up" procedure. In Section 4 we present some simulation results from our numerical tests of the algorithm.

2. Notations, definitions, and computational tools

2.1. Notations

Throughout the paper, uppercase letters such as X denote matrices. The $n \times n$ identity matrix is denoted by I_n. Moreover, the norm $\|\cdots\|$ denotes the spectral norm and $\|\cdots\|_F$ denotes the Frobenius norm. The notation $\mathbb{R}^{m \times n}$ represents the set of $m \times n$ real matrices.

2.2. Definitions

Definition 1 (The singular value decomposition) *For a matrix $X \in \mathbb{R}^{m \times n}$ with $m \geq n$ the SVD is*

$$X = W \begin{pmatrix} \Sigma \\ 0 \end{pmatrix} Y^T, \tag{3}$$

where left and right singular matrices W and Y are orthogonal matrices and $\Sigma = \mathrm{diag}(\sigma_1, \cdots, \sigma_n)$ is a diagonal matrix with the ordering

$$\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n \geq 0. \tag{4}$$

The diagonal entries of Σ are called the singular values of X.

For a given positive integer $k \ll n \ll m$, we block-partition the SVD in (3) as

$$X = \begin{pmatrix} W_k & W_0 & W_\perp \end{pmatrix} \begin{pmatrix} \Sigma_k & 0 \\ 0 & \Sigma_0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} Y_k & Y_0 \end{pmatrix}^T, \tag{5}$$

with $\Sigma_k = \mathrm{diag}(\sigma_1, \cdots, \sigma_k)$ and $\Sigma_0 = \mathrm{diag}(\sigma_{k+1}, \cdots, \sigma_n)$ being diagonal matrices containing the k largest and $n-k$ smallest singular values of X, respectively. The matrix X_k defined by

$$X_k = W_k \Sigma_k Y_k^T \tag{6}$$

is called rank-k matrix approximation to X. For some tolerance ϵ_M proportional to the machine unit, if the singular values satisfy

$$\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_k \gg \epsilon_M \geq \sigma_{k+1} \geq \cdots \sigma_n \geq 0, \tag{7}$$

then the value k is called the numerical rank of the matrix X. However, we are aware that the determination of numerical rank is a sensitive computation, especially when there are no well-defined gaps between singular values [13, 24]. Moreover, in some situations, like the example in [13, §5.4.1], the tolerance ϵ_M is chosen slightly larger.

Definition 2 (The truncated ULV decomposition) *For a matrix $X \in \mathbb{R}^{m\times n}$ with numerical rank $k \ll n \ll m$, the truncated ULV decomposition (truncated ULV) is*

$$X = U_1 L V_1^T + E, \tag{8}$$

where $L \in \mathbb{R}^{k\times k}$ is a nonsingular lower triangular matrix, $U_1 \in \mathbb{R}^{m\times k}$ and $V_1 \in \mathbb{R}^{n\times k}$ are left orthogonal matrices *(i.e. $U_1^T U_1 = V_1^T V_1 = I_k$), and $E \in \mathbb{R}^{m\times n}$ is an error matrix.*

The theoretical constraints on L and E are

$$\min \|L^{-1}\|_2 \leq \epsilon_M^{-1}, \quad \|E\|_2 < \epsilon_M, \quad U_1^T E = 0. \tag{9}$$

However, these cannot be achieved, in general, without using the SVD. Thus, we weaken these conditions to an almost equivalent problem:

$$\min \|L^{-1}\|_F \text{ subject to } \|E\|_F < \epsilon_M, \quad U_1^T E = 0. \tag{10}$$

To meet these conditions, we enforce the constraint on $\|E\|_F$.

To be able to say that the truncated ULV decomposition in (8) is a rank-k matrix approximation to X, the singular values of L approximate the k largest singular values of X. Moreover, the matrices U_1 and V_1 in (8) are good approximations to singular subspaces W_k and Y_k, respectively.

Proposition 3 *Let $X = U_1 L V_1^T + E$ be a truncated ULV of the matrix $X \in \mathbb{R}^{m\times n}$ with rank k. Then*

$$E = PX, \quad P = I_m - U_1 U_1^+, \tag{11}$$

where U_1^+ is the left pseudoinverse of U_1.

Proof See [2]. □

2.3. Computational tools

Local QR The primary matrix computational tool is the orthogonal factorization routine `local_qr`, which inputs a rectangular matrix $Z \in \mathbb{R}^{m\times p}$, $p \leq n \ll m$, and outputs an upper triangular matrix $R \in \mathbb{R}^{p\times p}$ and a left orthogonal matrix $Q \in \mathbb{R}^{m\times p}$ such that Q and R satisfy

$$\left\| I_p - Q^T Q \right\| \leq \epsilon_M \Omega\,(m, p) \ll 1 \tag{12}$$

and, in the backward errors sense,

$$Z + \Delta Z = QR, \quad \|\Delta Z\| \leq \epsilon_M \Omega(m,p) \|Z\|, \tag{13}$$

where $\Omega(m,p)$ is a slightly growing function. The routine `local_qr` can be implemented using Householder reflections or Givens rotations. An error analysis on Householder QR factorization given in [15, §19.3] yields $\Omega(m,p) = cmp^{3/2}$, where c is a constant. Our `local_qr` is just the MATLAB function `qr(Z,0)`, but it can be coded appropriately to BLAS-3 operations using skinny QR as discussed in [16]. We note that, by symmetric matrix eigenvalue/singular value relationship, condition (12) implies

$$\|Q\| \leq 1 + \frac{1}{2}\epsilon_M \Omega(m,p) \leq \sqrt{2}. \tag{14}$$

The block classical Gram–Schmidt The block classical Gram–Schmidt process named `BCGS` and outlined in Algorithm 1 takes a left orthogonal matrix $U \in \mathbb{R}^{m \times q}$ satisfying the condition

Algorithm 1 The block classical Gram–Schmidt.

```
function [Q, R, S] = BCGS(U, B)
% Input:
    % U left orthogonal matrix
    % B rectangular matrix
% Output:
    % Q left orthogonal matrix
    % R upper triangular matrix
S = U^T B;
Z = B − US;
[Q, R] = local_QR(Z);
end BCGS
```

$$\left\|I_t - U^T U\right\| \leq \epsilon_M \Omega(m,p,q) \ll 1 \tag{15}$$

for a modest funtion $\Omega(m,p,q)$ and a rectangular matrix $B \in \mathbb{R}^{m \times p}$ with $p+q \leq n$ to output a left orthogonal matrix $Q \in \mathbb{R}^{m \times p}$, an upper triangular matrix $R \in \mathbb{R}^{p \times p}$, and a rectangular matrix $S \in \mathbb{R}^{q \times p}$ such that, in exact arithmetic,

$$Z = \left(I_m - UU^T\right) B, \tag{16}$$

$$Z = QR, \quad Q^T Q = I_p, \tag{17}$$

$$B = US + QR, \quad S = U^T B. \tag{18}$$

However, in floating point arithmetic, Algorithm 1 actually computes

$$S + \delta S = U^T B, \quad \|\delta S\| \leq \epsilon_M \Psi\left(m, p, q\right) \|B\|, \tag{19}$$

$$Z + \delta Z = B - US, \quad \|\delta Z\| \leq \epsilon_M \Upsilon\left(m, p\right) \|B\|. \tag{20}$$

The definitions of the functions $\Psi\left(m, p, q\right)$ and $\Upsilon\left(m, p\right)$ can be found in [3, §3]. Note that the output Q satisfies (12) and along with (20) yields

$$B = US + QR + \delta Z - \Delta Z. \tag{21}$$

Also note that the upper triangular matrix R together with Q and Z satisfies (13). The operation count of the `BCGS` procedure is $\mathcal{O}(mp(p+q))$.

Two block classical Gram–Schmidt An important subproblem in the truncated ULV block update algorithm is for a given near left orthogonal matrix $U \in \mathbb{R}^{m\times q}$ satisfying condition (15) along with $B \in \mathbb{R}^{m\times p}$, $p + q \leq n$, to find a left orthogonal matrix $Q_B \in \mathbb{R}^{m\times p}$, an upper triangular matrix $R_B \in \mathbb{R}^{p\times p}$, and a rectangular matrix $S_B \in \mathbb{R}^{q\times p}$ such that, in exact aritmetic,

Algorithm 2 Two steps of block CGS.

```
function [Q_B, R_B, S_B] = T_BCGS(U, B)
% Input:
    % U left orthogonal matrix
    % B rectangular matrix
% Output:
    % Q_B left orthogonal matrix
    % R_B upper triangular matrix
[Q_1, R_1, S_1] = BCGS(U, B);
[Q_B, R_2, S_2] = BCGS(U, Q_1);
S_B = S_1 + S_2 R_1;
R_B = R_2 R_1
end T_BCGS
```

$$B = US_B + Q_B R_B, \tag{22}$$

$$U^T Q_B = 0, \quad S_B = U^T B. \tag{23}$$

The two block classical Gram–Schmidt algorithm (`T_BCGS`) that basically consists of two applications of the `BCGS` and is detailed in Algorithm 2 numerically solves the subproblem. The residual of (22) is bounded by

$$\|B - (US_B + Q_B R_B)\| \leq \epsilon_M \Gamma\left(m, p, q\right) \|B\|, \tag{24}$$

where $\Gamma(m,p,q)$ is a modest function given in [3, eq 3.5 in §3]. Moreover, the output Q_B obtained from `local_qr` satisfies

$$\left\|I_p - Q_B^T Q_B\right\| \le \epsilon_M \Omega(m,p) \ll 1 \tag{25}$$

and together with U satisfies

$$\left\|U^T Q_B^T\right\| \le 5\epsilon_M \Phi(m,p,q) \ll 1. \tag{26}$$

The detailed error analysis can be found in [3, §3].

Refinement To reconstruct the truncated ULV decomposition in (8), we also use a refinement algorithm that reduces $\|E\|_F$, detects rank degeneracy, corrects it, and sharpens the approximation. The algorithm inputs X, U_1, L, V_1, and $\|E\|_F$ with the condition

Algorithm 3 Vector-matrix product.

```
function  y = E_product(X, U, v)
% Input:
      % X  data matrix
      % U  left orthogonal matrix
      % v  vector
% Output:
      % y  vector
z = Xv;
f1 = U^T z; r1 = z - U f1;
f2U^T r1; y = r1 - U f2;
if(||y||_2 < sqrt(4/5)||r1||_2)
    j = min_{1<=i<=m} ||U^T e_i||_2; % e_i , ith column of the identity matrix
    t1 = U^T e_j; s1 = e_j - U t1;
    U^T s1; s2 = s1 - U t2;
    w = s2/||s2||_2;
    y = (w^T y)w;
end
end  E_product
```

$$\epsilon_M < \|E\|_F \le \sqrt{\epsilon_M^2 + \|E\|^2} \tag{27}$$

and outputs $\bar{U}_1 \in \mathbb{R}^{m\times s}$, $\bar{L} \in \mathbb{R}^{s\times k}$, $\bar{V}_1 \in \mathbb{R}^{n\times s}$ for $s \in \{k, k+1, k+2, \cdots, k+p\}$ and such that

$$X = \bar{U}_1 \bar{L} \bar{V}_1^T + \bar{E}, \quad \|\bar{E}\|_F < \epsilon_M. \tag{28}$$

The main steps of the algorithm are given in Algorithm 4. To keep computational complexity of the projections

$$Ev_1 = (I - U_1 U_1^+) X v_1, \tag{29}$$

$$E^T u_1 = X^T (I - U_1 U_1^+) u_1. \tag{30}$$

less than $\mathcal{O}(mn)$, instead of constructing the error matrix E, we use the procedure `E_product` outlined in Algorithm 3. The time complexity of the procedure is equal to

$$T_{\texttt{E\_product}} = \begin{cases} 4mk + \phi + \mathcal{O}(m) & \|y\|_2 \geq \texttt{sqrt}(\frac{4}{5})\|r_1\|_2 \\ 8mk + \phi + \mathcal{O}(m) & \text{otherwise,} \end{cases} \tag{31}$$

where ϕ is the number of operations to compute Xv. For more details including operations count and accuracy issues, see [4]. The procedure `modified_lanczos` in Algorithm 4 uses `E_product` and its operation count is $2 \times num\_iter \times T_{\texttt{E\_product}}$. The operations count of the procedure `inverse_iteration` is $num\_iter \times k^2$.

Moreover, the procedure `CGS_ORTH` in the refinement algorithm inputs $z \in \mathbb{R}^n$ and $V_1 \in \mathbb{R}^{n\times k}$ left orthogonal and outputs $d \in \mathbb{R}^{k+1}$ and $v_{k+1}\mathbb{R}^n$ such that

$$V_1^T v_{k+1} = 0, \quad (V_1 \; v_{k+1})d = z, \quad \|v_{k+1}\|_2 = 1. \tag{32}$$

The time complexity of `CGS_ORTH` is $\mathcal{O}(nk)$. The justification of the procedure is described in [4].

Thus, the overall time complexity of the refinement procedure is $\mathcal{O}(p(mk^2 + mn))$. Some theoretical results of the procedure can be found in [2].

3. Truncated ULV block update algorithm

For a matrix $X \in \mathbb{R}^{m\times n}$, $m \gg n$, assumed to have numerical rank $k \ll n$, with the truncated ULV given in (8), the matrix $\bar{X}$ given in (2) with $p < n$ can be rewritten as

$$\begin{aligned} \bar{X} &= \begin{pmatrix} U_1 L V_1^T + E \\ A^T \end{pmatrix} \\ &= \begin{pmatrix} U_1 L V_1^T \\ A^T \end{pmatrix} + \begin{pmatrix} E \\ 0 \end{pmatrix}. \end{aligned} \tag{33}$$

On the other hand, the algorithm `T_BCGS` with the input matrices A and V_1 produces the matrices V_{new}, L_{new}, and S_{new} such that

$$A^T = S_{new}^T V_1^T + L_{new} V_{new}^T. \tag{34}$$

Then, with the aid of equation (33), $\bar{X}$ can also be rewritten as

$$\bar{X} = \begin{pmatrix} U_1 & 0 \\ 0 & I_p \end{pmatrix} \begin{pmatrix} L & 0 \\ S_{new}^T & L_{new} \end{pmatrix} \begin{pmatrix} V_1^T \\ V_{new}^T \end{pmatrix} + \begin{pmatrix} E \\ 0 \end{pmatrix}.$$

Algorithm 4 Refinement.

function $\left[\bar{U}_1, \bar{L}, \bar{V}_1\right]$ = refinement($X, U_1, L, V_1, norm\_E$)

% Input:

% X data matrix

% U_1 near left orthogonal matrix

% V_1 near left orthogonal matrix

% L lower triangular matrix

% Output:

% $\bar{U}_1$ near left orthogonal matrix

% $\bar{V}_1$ near left orthogonal matrix

% $\bar{L}$ lower triangular matrix

$[u_1, \sigma_1, v_1] = \texttt{modified\_lanczos}(X, U_1, num\_iter, initial\_guess);$

$z = X^T u_1 (= E^T u_1);$

$[d, v_{k+1}] = \texttt{CGS\_ORTH}(V_1, z);\ \ f = d(1:k);\ \ \alpha = d(k+1);$

$norm\_\bar{E} = \texttt{sqrt}((norm\_E)^2 - \sigma_1^2);$

$\bar{L} = \begin{bmatrix} L & 0 \\ f^T & \alpha \end{bmatrix};\quad \bar{U}_1 = [U_1\ u_1];\quad \bar{V}_1 = [V_1\ v_{k+1}];\ k = k+1;$

$[y_k, \sigma_k, z_k] = \texttt{inverse\_iteration}(\bar{L}, num\_iter, initial\_guess);$

$\texttt{while}(\texttt{sqrt}(\sigma_k^2 + (norm\_\bar{E})^2) \leq \epsilon_F)$

$Q^T z_k = e_k;$ % Q orthogonal matrix

$Q^T \bar{L} Z = \begin{bmatrix} \bar{L} & 0 \\ 0 & \sigma_k \end{bmatrix};$ % Z orthogonal matrix, maintains a lower triangular matrix

$\left[\bar{U}_1\ \bar{u}_1\right] = [U_1\ u_1]\,Q;\quad \left[\bar{V}_1\ \bar{v}_{k+1}\right] = [V_1\ v_{k+1}]\,Z;\ norm\_\bar{E} = \texttt{sqrt}(\sigma_{k+1}^2 + (norm\_\bar{E})^2);\ k = k-1;$

$[y_k, \sigma_k, z_k] = \texttt{inverse\_iteration}(\bar{L}, num\_iter, initial\_guess);$

end while

end refinement

Furthermore, by defining

$$\bar{U}_1 = \begin{pmatrix} U_1 & 0 \\ 0 & I_p \end{pmatrix}, \bar{L} = \begin{pmatrix} L & 0 \\ S_{new}^T & L_{new} \end{pmatrix}, \bar{V}_1 = \begin{pmatrix} V_1 & V_{new} \end{pmatrix}, \quad \text{and} \quad \bar{E} = \begin{pmatrix} E \\ 0 \end{pmatrix}$$

we obtain

$$\bar{X} = \bar{U}_1 \bar{L} \bar{V}_1^T + \bar{E}. \tag{35}$$

In order to say that the latter equation is the truncated ULV of matrix $\bar{X}$, we have to show that the conditions in Definition 2 are satisfied. First, it is obvious that matrix $\bar{L}$ is a lower triangular matrix; however, it is not so obvious that it protects its rank-revealing property, i.e. the rank is between k and $k+p$. We clarify this issue later.

Second, matrices $\bar{U}_1$ and $\bar{V}_1$ are left orthogonal. To prove the first, we consider

$$\begin{aligned}\bar{U}_1^T\bar{U}_1 &= \begin{pmatrix} U_1^T & 0 \\ 0 & I_p \end{pmatrix}\begin{pmatrix} U_1 & 0 \\ 0 & I_p \end{pmatrix} \\ &= \begin{pmatrix} U_1^T U_1 & 0 \\ 0 & I_p \end{pmatrix},\end{aligned}$$

and thus the left orthogonality of $\bar{U}_1$ follows from the left orthogonality of U_1. For the left orthogonality of $\bar{V}_1$, we consider

$$\begin{aligned}\bar{V}_1^T\bar{V}_1 &= \begin{pmatrix} V_1^T \\ V_{new}^T \end{pmatrix}\begin{pmatrix} V_1 & V_{new} \end{pmatrix} \\ &= \begin{pmatrix} V_1^T V_1 & V_1^T V_{new} \\ V_{new}^T V_1 & V_{new}^T V_{new} \end{pmatrix}.\end{aligned}$$

The off diagonal block entries are zero by equation (23), so the left orthogonality of V_1 implies that $\bar{V}_1$ is left orthogonal.

Third, matrices $\bar{L}$ and $\bar{E}$ satisfy the latter condition in (10), which is given in the following theorem.

Theorem 4 *Let X be an $m \times n$ matrix of numerical rank $k \ll n \ll m$ with the truncated ULV in (8). Let $\bar{X}$ be an $(m+p)\times n$ matrix as in (2) with the decomposition as in (35). Then $\bar{U}_1^T\bar{E} = 0$.*

Proof Before we work on $\bar{U}_1^T\bar{E}$ we first recall that, by equation (10), $U_1^T E = 0$. Then,

$$\begin{aligned}\bar{U}_1^T\bar{E} &= \bar{U}_1^T\left(\bar{X} - \bar{U}_1\bar{L}\bar{V}_1^T\right) \\ &= \bar{U}_1^T\bar{X} - \bar{U}_1^T\bar{U}_1\bar{L}\bar{V}_1^T \\ &= \bar{U}_1^T\bar{X} - \bar{L}\bar{V}_1^T \\ &= \begin{pmatrix} U_1^T & 0 \\ 0 & I \end{pmatrix}\begin{pmatrix} X \\ A^T \end{pmatrix} - \begin{pmatrix} L & 0 \\ S_{new}^T & L_{new} \end{pmatrix}\begin{pmatrix} V_1^T \\ V_{new}^T \end{pmatrix} \\ &= \begin{pmatrix} U_1^T X \\ A^T \end{pmatrix} - \begin{pmatrix} L V_1^T \\ S_{new}^T V_1^T + L_{new}V_{new}^T \end{pmatrix} \\ &= \begin{pmatrix} U_1^T X - L V_1^T \\ A^T - S_{new}^T V_1^T - L_{new}V_{new}^T \end{pmatrix} \\ &= 0.\end{aligned}$$

□

The decomposition in (35) has a similar property as described in Proposition 3. Before stating the property we recall some auxiliary results of the Moore–Pensore pseudoinverse of $\bar{U}_1$ from [13]. Define $\bar{U}_1^+$, a unique matrix, to be the Moore-Pensore pseudoinverse of $\bar{U}_1$. Then it satisfies

$$\bar{U}_1\bar{U}_1^+\bar{U}_1 = \bar{U}_1, \tag{36}$$

$$\bar{U}_1^+\bar{U}_1\bar{U}_1^+ = \bar{U}_1^+, \quad (37)$$

$$(\bar{U}_1\bar{U}_1^+)^T = \bar{U}_1\bar{U}_1^+, \quad (38)$$

$$(\bar{U}_1^+U)^T = \bar{U}_1^+\bar{U}_1. \quad (39)$$

Proposition 5 *Let* $\bar{X} = \bar{U}_1\bar{L}\bar{V}_1^T + \bar{E}$ *be a decomposition of the matrix* $\bar{X} \in \mathbb{R}^{(m+p)\times n}$ *with rank* $k \ll n$ *as in (35). Then*

$$\bar{E} = \bar{P}\bar{X},$$

where $\bar{P} = I - \bar{U}_1\bar{U}_1^+$.

Proof Let us multiply $\bar{X}$ from left by $\bar{P}$ to obtain

$$\begin{aligned}\bar{P}\bar{X} &= \left(I - \bar{U}_1\bar{U}_1^+\right)\bar{X}\\ &= \bar{X} - \bar{U}_1\bar{U}_1^+\bar{X}\\ &= \bar{X} - \bar{U}_1\bar{U}_1^+\left(\bar{U}_1\bar{L}\bar{V}_1^T + \bar{E}\right)\\ &= \bar{X} - \bar{U}_1\bar{U}_1^+\bar{U}_1\bar{L}\bar{V}_1^T + \bar{U}_1\bar{U}_1^+\bar{E}.\end{aligned}$$

Equations (36) and (38) yield

$$\begin{aligned}\bar{P}\bar{X} &= \bar{X} - \bar{U}_1\bar{L}\bar{V}_1^T + \left(\bar{U}_1\bar{U}_1^+\right)^T\bar{E}\\ &= \bar{E} - \left(\bar{U}_1^+\right)^T\bar{U}_1^T\bar{E}.\end{aligned}$$

The proof follows from Theorem 4. □

The proposition allows that we do not have to store $\bar{E}$.

We now turn our attention to the conditions in (10). With the result stated in Theorem 4, to meet all conditions we have to consider the problem

$$\min \|\bar{L}^{-1}\|_F \text{ subject to } \|\bar{E}\|_F < \epsilon_M. \quad (40)$$

Here, we enforce the condition on the constraint on $\|\bar{E}\|_F$. To do so, we use the refinement algorithm discussed in Subsection 2.3. The refinement algorithm assures us that $\|\bar{E}\|_F < \epsilon_M$ is always maintained.

The block-truncated ULV update algorithm is summarized in Algorithm 5. The overall time complexity of the algorithm is $\mathcal{O}(mp(k^2+n) + np(k+p))$.

4. Numerical tests

In this section we present some simulation results from our numerical experiments. We use the so-called block exponential window process, at time step t, described as

$$X(t+1) = \begin{pmatrix}\alpha X(t)\\ A^T(t)\end{pmatrix},$$

Algorithm 5 Truncated ULV block update

```
function [Ū₁, L̄, V̄₁] = truncated_ULV_block_update(X, U₁, L, V₁, A)
% Input:
    % X data matrix
    % U₁ near left orthogonal matrix
    % V₁ near left orthogonal matrix
    % L lower triangular matrix
    % A new data matrix to be added
% Output:
    % Ū₁ near left orthogonal matrix
    % V̄₁ near left orthogonal matrix
    % L̄ lower triangular matrix
```

$\bar{X} = \begin{bmatrix} X \\ A^T \end{bmatrix}$;

$[V_{new}, L_{new}^T, S_{new}]$ = T_BCGS (V_1, A);

$\tilde{U}_1 = \begin{bmatrix} U_1 & 0 \\ 0 & I \end{bmatrix}$; $\tilde{L} = \begin{bmatrix} L & 0 \\ S_{new}^T & L_{new} \end{bmatrix}$; $\tilde{V}_1 = [V_1 \quad V_{new}]$; $norm\_\tilde{E} = norm\_E$; % $\tilde{E} = \begin{bmatrix} E \\ 0 \end{bmatrix}$

$[\bar{U}_1, \bar{L}, \bar{V}_1]$ = refinement$(\bar{X}, \tilde{U}_1, \tilde{L}, \tilde{V}_1, norm_\tilde{E})$;

end truncated_ULV_block_update

where $0 < \alpha \leq 1$ is called the forgetting factor.

The data matrix, say X_{data}, is an M-by-n random matrix, chosen from a uniform distribution on the interval $(0, 1)$, but scale r randomly selected rows by factor η so that rank changes occur often.

At the initial step $t = 0$, the decomposition of the data matrix $X(0) \in \mathbb{R}^{m \times n}$ containing the first m rows of X_{data} is obtained by using the MATLAB function `lulv` in the UTV tools software package [12]. If we let

$$M = 2^{\omega}, \quad m = 2^{\psi}, \quad n = 2^{\eta} \quad \text{with} \quad \eta < \psi < \omega,$$

then, at steps $t > 0$, the data block with the block size $p = 2^{\mu}$, $\mu < \eta$, defined as

$$A(t) = X_{data}^T(m + p * (t-1) : m + p * t, :),$$

is applied to the block exponential window process. The number of steps ν throughout the exponential window process is

$$\nu = 2^{\psi - \mu}(2^{\omega - \psi} - 1).$$

For each $t = 0, \cdots \nu$, we check the left orthogonality of the matrices $U_1(t)$ and $V_1(t)$ by computing

$$\left\| I - U_1^T(t) U_1(t) \right\|_F$$

and

$$\left\| I - V_1^T(t)V_1(t) \right\|_F,$$

respectively. We measure the decomposition error

$$\|E(t)\|_F = \left\| X(t) - U_1(t)L(t)V_1^T(t) \right\|_F$$

and, by using the equation (11), compute

$$\left\| U_1^T(t)E(t) \right\|_F$$

after each truncated ULV block update. We plot these quantities on $\log 10$ scale.

We also track the numerical rank $k(t)$ of matrix $X(t)$ at each step t and plot it.

On the other hand, we compute the SVD of $X(t)$ using MATLAB's `svd` function to obtain

$$X(t) = W(t) \begin{pmatrix} \Sigma(t) \\ 0 \end{pmatrix} Y^T(t)$$

as a reference in checking the accuracy at each step t. We partition the right orthogonal factor as

$$\begin{pmatrix} Y_{k(t)}(t) & Y_0(t) \end{pmatrix}.$$

In the Davis–Kahan [7] framework the accuracy of the right subspace errors is characterized by

$$|(\sin\theta)(t)| = \left\| V_1^T(t)Y_0(t) \right\|_F.$$

For each t, we compute $(\sin\theta)(t)$ and plot this on $\log 10$ scale.

Moreover, at each block step t, we compare the block-truncated ULV update with the SVD block update algorithm given in [19] in terms of speed and plot this on $\log 10$ scale as well.

Example 6 *For the data matrix X_{data}, the initial matrix $X(0)$, and the data block, we let $\omega = 14$, $\psi = 13$, $\eta = 9$, and $\mu = 8$. Later, we multiply $r = \lfloor 95\% M \rfloor$ randomly selected rows of X_{data} by $\eta = 10^{-9}$. The rank tolerance $\epsilon = 10^{-8}$ and the forgetting factor $\alpha = 0.9$.*

Example 7 *For the data matrix X_{data}, the initial matrix $X(0)$, and the data block, we let $\omega = 14$, $\psi = 13$, $\eta = 9$, and $\mu = 8$. Later, we multiply $r = \lfloor 95\% M \rfloor$ randomly selected rows of X_{data} by $\eta = 10^{-9}$. The rank tolerance $\epsilon = 10^{-8}$ and the forgetting factor $\alpha = 0.7$.*

Example 8 *For the data matrix X_{data}, the initial matrix $X(0)$, and the data block, we let $\omega = 14$, $\psi = 13$, $\eta = 9$, and $\mu = 8$. Later, we multiply $r = \lfloor 95\% M \rfloor$ randomly selected rows of X_{data} by $\eta = 10^{-9}$. The rank tolerance $\epsilon = 10^{-8}$ and the forgetting factor $\alpha = 0.5$.*

Figures 1, 3, and 2 show the ability of the block update algorithm. The graphs demonstrate that the algorithm is robust and promising. Moreover, the block-truncated ULV update algorithm performs better than the block SVD update in [19].

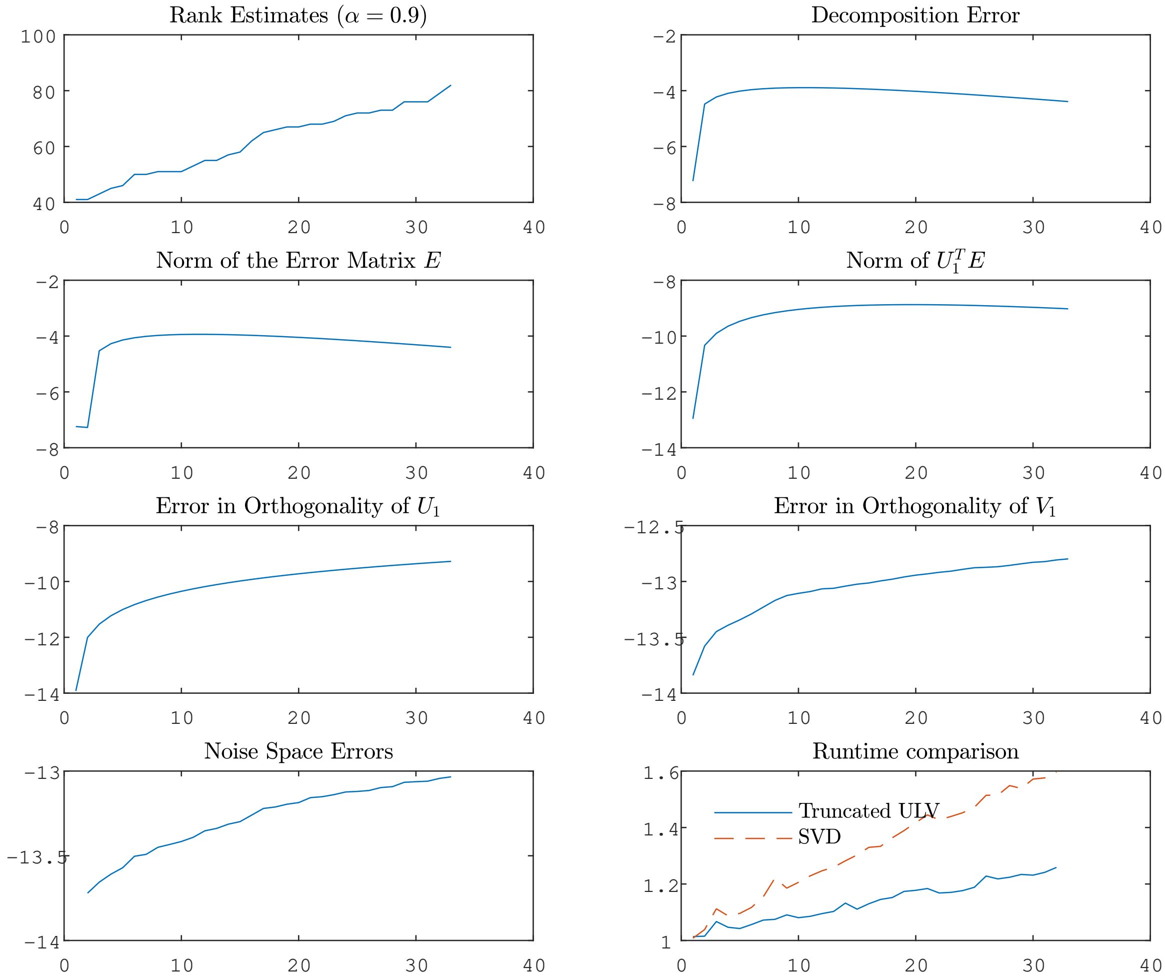

Figure 1. Numerical results by Example 6.

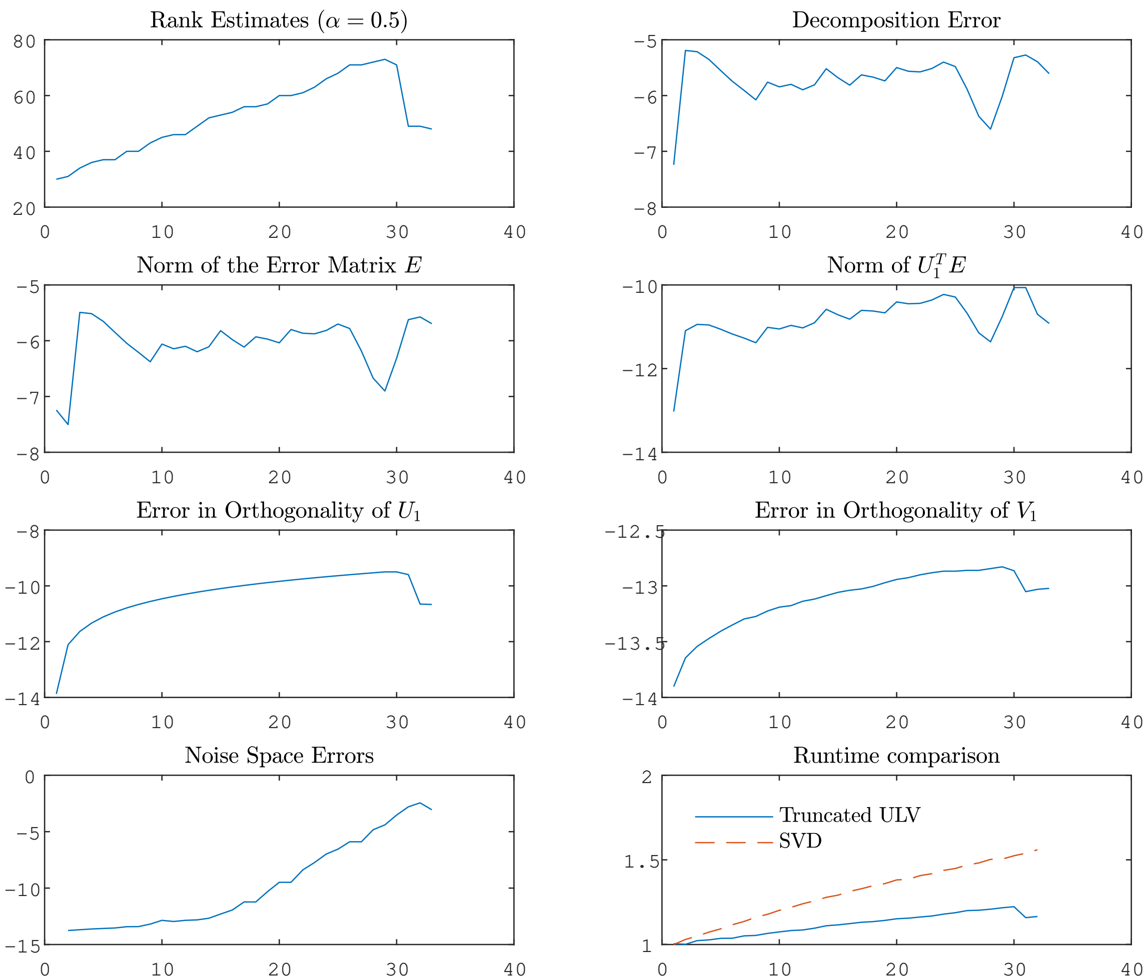

Figure 2. Numerical results by Example 7.

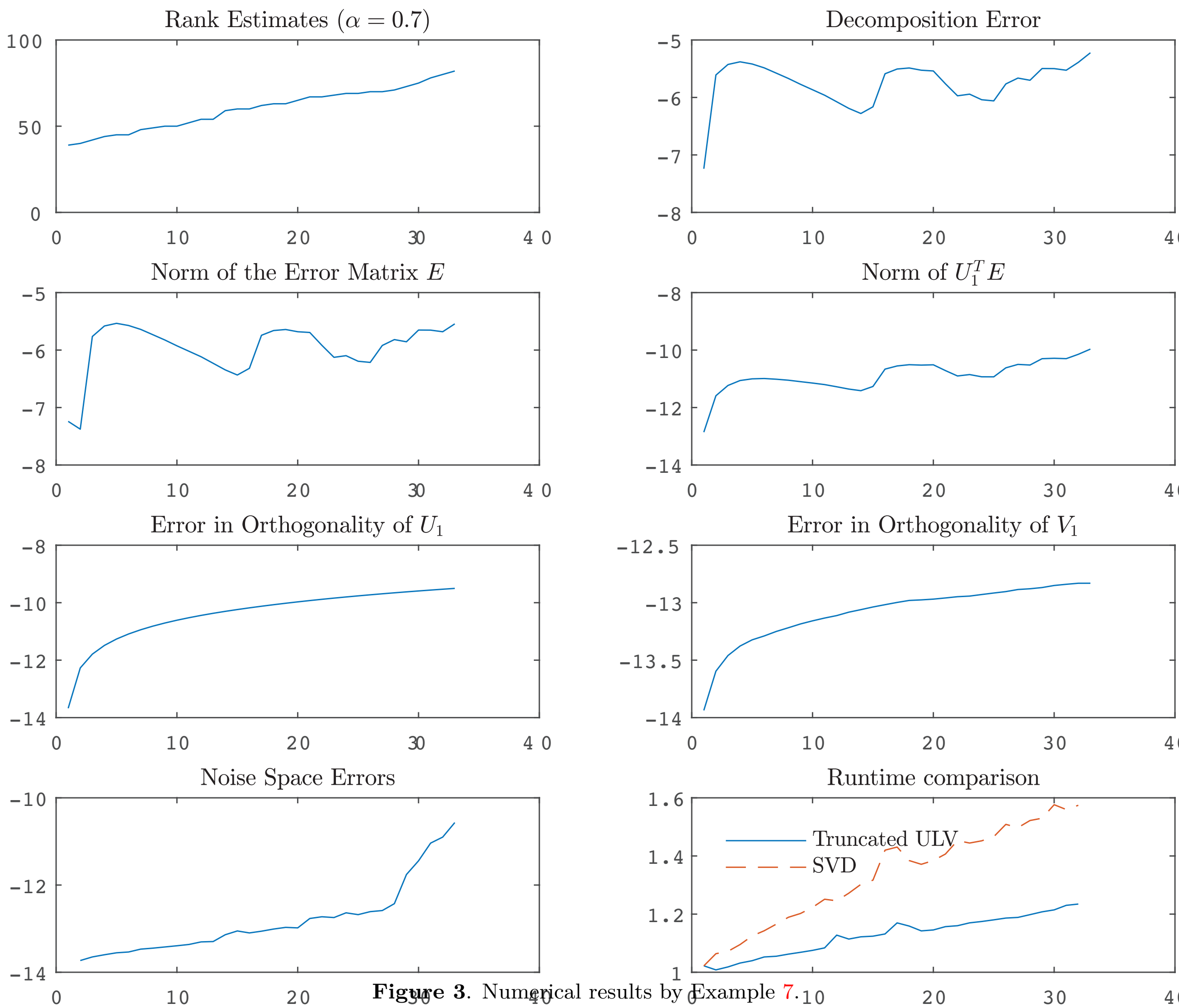

Figure 3. Numerical results by Example 7.

5. Conclusion

We have proposed a BLAS-3 compatible block update algorithm based on the block classical Gram–Schmidt process. Since the update algorithm is built upon matrix–matrix operation rather than matrix–vector operation, it makes effective use of caching to avoid excessive movement of data to/from memory. We have seen that the analysis and the numerical results are consistent.

Acknowledgment

The authors would like to thank the anonymous reviewers for their constructive comments to improve the quality of the manuscript. This study was supported by the Kırıkkale University Scientific Research Projects Coordination Unit, project number 2016/150.

References

[1] Barlow JL, Aydoğan E, Erbay H. Block updates on truncated ULV decomposition. Advances in Computational Science, Engineering and Information Technology 2013; 225: 73-79.

[2] Barlow JL, Erbay H. Modifible low-rank approximation to a matrix. Numer Linear Algebr 2009; 16: 833-860.

[3] Barlow JL, Smoktunowicz A. Reorthogonalized block classical Gram-Schmidt. Numer Math 2013; 123: 395–423.

[4] Barlow JL, Smoktunowicz A, Erbay H. Improved Gram-Schmidt type downdating methods. BIT 2005; 45: 259-285.

[5] Berry MW, Browne M. Understanding Search Engines: Mathematical Modeling and Text Retrieval. Philadelphia, PA, USA: SIAM, 2005.

[6] Berry MW, Dumais S, O'Brien GW. Using linear algebra for intelligent information retrieval. SIAM Rev 1995; 37: 573-595.

[7] Davis C, Kahan WM. The rotation of eigenvectors by a perturbation III. SIAM J Numer Anal 1970; 7: 1-46.

[8] Deerwester S, Dumais S, Furnas S, Landauer TG, Harshman R. Indexing by latent semantic analysis. J Am Soc Inform Sci 1990; 41: 391-407.

[9] Deprettere F. SVD and Signal Processing: Algorithms, Applications and Architectures. Amsterdam, the Netherlands: North-Holland Publishing Co., 1989.

[10] Dongarra JJ, du Croz J, Hammarling S, Duff I. A set of level 3 basic linear algebra subprograms. ACM T Math Software 1990; 16: 1-17.

[11] Drineas P, Kannan R, Mahoney MW. Fast Monte Carlo algorithms for matrices II: Computing a low-rank approximation to a matrix. SIAM J Comput 2006; 36: 158-183.

[12] Fierro RD, Hansen PC. Low-rank revealing UTV decompositions. Numer Algorithms 1997; 15: 37-55.

[13] Golub GH, Van Loan CF. Matrix Computations. Baltimore, MD, USA: Johns Hopkins Press, 2013.

[14] Halko N, Martinsson P, Tropp J. Finding structure with randomness: probabilistic algorithms for constructing approximate matrix decompositions. SIAM Rev 2011; 53: 217–288.

[15] Higham NJ. Accuracy and Stability of Numerical Analysis. Cambridge, UK: Cambridge University Press, 2002.

[16] Hoemmen M. Communication-avoiding Krylov Subspace Methods. Berkeley, CA, USA: University of California, 2010.

[17] Kleinberg JM. Two algorithms for nearest-neighbor search in high dimensions. In: Proceedings of the Twenty-Ninth Annual ACM Symposium on Theory of Computing, 1997, pp. 599-608.

[18] Kleinberg JM. Authoritative sources in a hyperlinked environment. J ACM 1999; 46: 604-632.

[19] Matthew B. Fast low-rank modifications of the thin singular value decomposition. Linear Algebra Appl 2006; 415: 20-30.

[20] Moonen M, De Moor B. SVD and Signal Processing III: Algorithms, Architectures and Applications. Amsterdam, the Netherlands: Elsevier, 1995.

[21] Papadimitriou CH, Raghavan P, Tamaki H, Vempala S. Latent semantic indexing: a probabilistic analysis. In: Proceedings of the Seventeenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, 1998, pp. 159-168.

[22] Stewart GW. Block Gram-Schmidt orthogonalization. SIAM J Sci Comput 2008; 31: 761-775.

[23] Vaccaror RJ. SVD and Signal Processing II: Algorithms, Analysis and Applications. Amsterdam, the Netherlands: Elsevier, 1991.

[24] Watkins DS. Fundamentals of Matrix Computations. New York, NY, USA: John Wiley and Sons, 2002.