Electronic Transactions on Numerical Analysis.
Volume 51, pp. 469–494, 2019.

ISSN 1068–9613.
DOI: 10.1553/etna_vol51s469

ETNA
Kent State University and
Johann Radon Institute (RICAM)

FLIP-FLOP SPECTRUM-REVEALING QR FACTORIZATION AND ITS APPLICATIONS TO SINGULAR VALUE DECOMPOSITION*

YUEHUA FENG†, JIANWEI XIAO‡, AND MING GU‡

Abstract. We present the Flip-Flop Spectrum-Revealing QR (Flip-Flop SRQR) factorization, a significantly faster and more reliable variant of the QLP factorization of Stewart for low-rank matrix approximations. Flip-Flop SRQR uses SRQR factorization to initialize a partial column-pivoted QR factorization and then computes a partial LQ factorization. As observed by Stewart in his original QLP work, Flip-Flop SRQR tracks the exact singular values with "considerable fidelity". We develop singular value lower bounds and residual error upper bounds for the Flip-Flop SRQR factorization. In situations where singular values of the input matrix decay relatively quickly, the low-rank approximation computed by Flip-Flop SRQR is guaranteed to be as accurate as the truncated SVD. We also perform a complexity analysis to show that Flip-Flop SRQR is faster than the randomized subspace iteration for approximating the SVD, the standard method used in the Matlab tensor toolbox. We additionally compare Flip-Flop SRQR with alternatives on two applications, a tensor approximation and a nuclear norm minimization, to demonstrate its efficiency and effectiveness.

Key words. QR factorization, randomized algorithm, low-rank approximation, approximate SVD, higher-order SVD, nuclear norm minimization

AMS subject classifications. 15A18, 15A23, 65F99

1. Introduction.

The singular value decomposition (SVD) of a matrix $A \in \mathbb{R}^{m\times n}$ is the factorization of A into the product of three matrices $A = U\Sigma V^T$ where the matrices $U = (u_1, \cdots, u_m) \in \mathbb{R}^{m\times m}$ and $V = (v_1, \cdots, v_n) \in \mathbb{R}^{n\times n}$ are orthogonal singular vector matrices and $\Sigma \in \mathbb{R}^{m\times n}$ is a rectangular diagonal matrix with non-increasing non-negative singular values σ_i $(1 \le i \le \min(m, n))$ on the diagonal. The SVD has become a critical analytic tool in large data analysis and machine learning [1, 19, 53].

Let Diag (x) denote the diagonal matrix with vector $x \in \mathbb{R}^n$ on its diagonal. For any $1 \le k \le \min(m, n)$, the rank-k truncated SVD of A is defined by

$$A_k \overset{def}{=} (u_1, \cdots, u_k)\,\mathrm{Diag}\,(\sigma_1, \cdots, \sigma_k)\,(v_1, \cdots, v_k)^T.$$

The rank-k truncated SVD turns out to be the best rank-k approximation to A with respect to spectral norm and Frobenius norm, as explained by the Eckart-Young-Mirsky Theorem [18, 26].

However, due to the prohibitive costs in computing the rank-k truncated SVD, in practical applications one typically computes a rank-k approximate SVD which satisfies some tolerance requirements [16, 27, 30, 42, 66]. The rank-k approximate SVD has been applied to many research areas including principal component analysis (PCA) [37, 58], web search models [38], information retrieval [4, 23], and face recognition [52, 22].

Among assorted SVD approximation algorithms, the pivoted QLP decomposition proposed by Stewart [66] is an effective and efficient one. The pivoted QLP decomposition is obtained by computing a QR factorization with column pivoting (QRCP) [6, 25] of A to get an upper triangular factor R and then computing an LQ factorization of R to get a lower triangular factor L. Stewart's key numerical observation is that the diagonal elements of L track the

*Received October 14, 2018. Accepted November 6, 2019. Published online on December 11, 2019. Recommended by Y. Saad. The research by M. Gu was supported in part by NSF award FRG-1760316.

†School of Mathematics, Physics and Statistics, Shanghai University of Engineering Science, China (fyh1001@hotmail.com).

‡Department of Mathematics, University of California, Berkeley (jwxiao@berkeley.edu, mgu@math.berkeley.edu).

ETNA
Kent State University and
Johann Radon Institute (RICAM)

singular values of A with "considerable fidelity" no matter what the matrix A is. The pivoted QLP decomposition is extensively analyzed in Huckaby and Chan [34, 35]. More recently, Duersch and Gu developed a much more efficient variant of the pivoted QLP decomposition, TUXV, and demonstrated its remarkable quality as a low-rank approximation empirically without a rigorous theoretical justification of TUXV's success [17].

In this paper, we present Flip-Flop SRQR, a slightly different variant of TUXV of Duersch and Gu [17]. Like TUXV, Flip-Flop SRQR requires the most effort for computing a partial QR factorization using a truncated randomized QRCP (TRQRCP) and a partial LQ factorization. Unlike TUXV, however, Flip-Flop SRQR also performs additional computations to ensure a spectrum-revealing QR factorization (SRQR) [75] before the partial LQ factorization.

We demonstrate the reliable theoretical quality of this variant as a low-rank approximation and its highly competitiveness with state-of-the-art low-rank approximation methods in real world applications in both low-rank tensor compression [13, 40, 60, 71] and nuclear norm minimization [7, 45, 49, 54, 68].

The rest of this paper is organized as follows: In Section 2 we introduce the TRQRCP algorithm, the SRQR factorization, the low-rank tensor compression, and the nuclear norm minimization. In Section 3, we introduce Flip-Flop SRQR and analyze its computational costs and low-rank approximation properties. In Section 4, we present numerical results comparing Flip-Flop SRQR with state-of-the-art low-rank approximation methods. In Section 5, we summarize our main results and draw some conclusions.

2. Preliminaries and background.

2.1. Partial QRCP. The QR factorization of a matrix $A \in \mathbb{R}^{m\times n}$ is $A = QR$ with an orthogonal matrix $Q \in \mathbb{R}^{m\times m}$ and an upper trapezoidal matrix $R \in \mathbb{R}^{m\times n}$, which can be computed by the LAPACK [2] routine xGEQRF, where x stands for the matrix data type. The standard QR factorization is not suitable for some practical situations where either the matrix A is rank deficient or only representative columns of A are of interest. Usually the QRCP is adequate for the aforementioned situations except a few rare examples such as the Kahan matrix [26]. Given a matrix $A \in \mathbb{R}^{m\times n}$, the QRCP of matrix A has the form

$$A\Pi = QR,$$

where $\Pi \in \mathbb{R}^{n\times n}$ is a permutation matrix, $Q \in \mathbb{R}^{m\times m}$ is an orthogonal matrix, and $R \in \mathbb{R}^{m\times n}$ is an upper trapezoidal matrix. QRCP can be computed by the LAPACK [2] routines xGEQPF and xGEQP3, where xGEQP3 is a more efficient blocked implementation of xGEQPF. For a given target rank k $(1 \le k \le \min(m,n))$, the partial QRCP factorization (Algorithm 4 in the appendix) has a 2×2 block form

$$A\Pi = Q\begin{bmatrix} R_{11} & R_{12} \\ & R_{22} \end{bmatrix} = \begin{bmatrix} Q_1 & Q_2 \end{bmatrix}\begin{bmatrix} R_{11} & R_{12} \\ & R_{22} \end{bmatrix}, \tag{2.1}$$

where $R_{11} \in \mathbb{R}^{k\times k}$ is upper triangular. The partial QRCP computes an approximate column subspace of A spanned by the leading k columns in $A\Pi$ up to the error term in R_{22}. Equivalently, (2.1) yields a low rank approximation

$$A \approx Q_1 \begin{bmatrix} R_{11} & R_{12} \end{bmatrix}\Pi^T, \tag{2.2}$$

with an approximation quality closely related to the error term in R_{22}.

The Randomized QRCP (RQRCP) algorithm [17, 75] is a more efficient variant of Algorithm 4. RQRCP generates a Gaussian random matrix $\Omega \in \mathcal{N}(0,1)^{(b+p)\times m}$ with $b + p \ll m$, where the entries of Ω are independently sampled from a normal distribution

ETNA
Kent State University and
Johann Radon Institute (RICAM)

Algorithm 1 Truncated Randomized QRCP (TRQRCP).

Inputs:
Matrix $A \in \mathbb{R}^{m\times n}$. Target rank k. Block size b. Oversampling size $p \geq 0$.
Outputs:
Orthogonal matrix $Q \in \mathbb{R}^{m\times m}$.
Upper trapezoidal matrix $R \in \mathbb{R}^{k\times n}$.
Permutation matrix $\Pi \in \mathbb{R}^{n\times n}$ such that $A\Pi \approx Q\,(:, 1:k)\,R$.
Algorithm:
Generate i.i.d. Gaussian random matrix $\Omega \in \mathcal{N}\,(0,1)^{(b+p)\times m}$.
Form the initial sample matrix $B = \Omega A$ and initialize $\Pi = I_n$.
for $j = 1 : b : k$ **do**
  $b = \min\,(k - j + 1, b)$.
  Do partial QRCP on $B\,(:, j : n)$ to obtain b pivots.
  Exchange corresponding columns in $A,\ B,\ \Pi$ and W^T.
  Do QR on $A\,(j : m, j : j + b - 1)$ using WY formula without updating the trailing matrix.
  Update $B\,(:, j + b : n)$ using the formula (2.4).
end for
$Q = Q_1 Q_2 \cdots Q_{\lceil k/b\rceil}$. $R =$ upper trapezoidal part of the submatrix $A\,(1 : k, 1 : n)$.

to compress A into $B = \Omega A$ with much smaller row dimension. In practice, b is the block size and p is the oversampling size. RQRCP repeatedly runs partial QRCP on B to obtain b column pivots, applies them to the matrix A, and then computes QR without pivoting (QRNP) on A and updates the remaining columns of B. RQRCP exits this process when it reaches the target rank k. QRCP and RQRCP choose pivots on A and B respectively. RQRCP is significantly faster than QRCP as B has much smaller row dimension than A. It is shown in [75] that RQRCP is as reliable as QRCP up to failure probabilities that decay exponentially with respect to the oversampling size p.

Since the trailing matrix of A is usually not required for low-rank matrix approximations (see (2.2)), the TRQRCP (truncated RQRCP) algorithm of [17] re-organizes the computations in RQRCP to directly compute the approximation (2.2) without explicitly computing the trailing matrix R_{22}. For more details, both RQRCP and TRQRCP are based on the WY representation of the Householder transformations [5, 55, 61]:

$$Q = Q_1 Q_2 \cdots Q_k = I - YTY^T,$$

where $T \in \mathbb{R}^{k\times k}$ is an upper triangular matrix and $Y \in \mathbb{R}^{m\times k}$ is a trapezoidal matrix consisting of k consecutive Householder vectors. Let $W^T \overset{def}{=} T^T Y^T A$, then the trailing matrix update formula becomes $Q^T A = A - YW^T$. The main difference between RQRCP and TRQRCP is that RQRCP computes the whole trailing matrix update, while TRQRCP only computes the part of the update that affects the approximation (2.2). More discussions about RQRCP and TRQRCP can be found in [17]. The main steps of TRQRCP are briefly described in Algorithm 1. The deduction of the updating formula for B is as follows: consider partial QRs on $A\Pi$ and $B\Pi = \Omega A\Pi$,

$$A\Pi = Q \begin{bmatrix} R_{11} & R_{12} \\ & R_{22} \end{bmatrix}, \qquad B\Pi = \overline{Q} \begin{bmatrix} \overline{R}_{11} & \overline{R}_{12} \\ & \overline{R}_{22} \end{bmatrix}. \tag{2.3}$$

ETNA
Kent State University and
Johann Radon Institute (RICAM)

Partition $\overline{\Omega} \overset{def}{=} \overline{Q}^T \Omega Q \overset{def}{=} \begin{bmatrix} \overline{\Omega}_1 & \overline{\Omega}_2 \end{bmatrix}$, and therefore (2.3) implies

$$\overline{\Omega} \begin{bmatrix} R_{11} & R_{12} \\ & R_{22} \end{bmatrix} = \begin{bmatrix} \overline{R}_{11} & \overline{R}_{12} \\ & \overline{R}_{22} \end{bmatrix},$$

which leads to the updating formula for B in Algorithm 1,

$$\begin{bmatrix} \overline{R}_{12} \\ \overline{R}_{22} \end{bmatrix} \leftarrow \overline{\Omega}_2 R_{22} = \begin{bmatrix} \overline{R}_{12} - \overline{R}_{11} R_{11}^{-1} R_{12} \\ \overline{R}_{22} \end{bmatrix}. \tag{2.4}$$

With TRQRCP, the TUXV algorithm [17, Algorithm 7] (Algorithm 5 in the appendix), computes a low-rank approximation with the QLP factorization at a greatly accelerated speed, by computing a partial QR factorization with column pivoting, followed with a partial LQ factorization.

2.2. Spectrum-revealing QR factorization. Although both RQRCP and TRQRCP are very effective practical tools for low-rank matrix approximations, they are not known to provide reliable low-rank matrix approximations due to their underlying greediness in the column norm-based pivoting strategy. To solve this potential problem of column-based QR factorization, Gu and Eisenstat [28] proposed an efficient way to perform additional column interchanges to enhance the quality of the leading k columns in $A\Pi$ as a basis for the approximate column subspace. More recently, a more efficient and effective method, SRQR, was introduced and analyzed in [50, 75] to compute the low-rank approximation (2.2). The concept of spectrum-revealing, introduced in [50, 74], emphasizes the utilization of partial QR factorization (2.2) as a low-rank matrix approximation, as opposed to the more traditional rank-revealing factorization, which emphasizes the utility of the partial QR factorization (2.1) as a tool for numerical rank determination. The SRQR algorithm is described in Algorithm 2. The SRQR algorithm will always run to completion with a high-quality low-rank matrix approximation (2.2). For real data matrices that usually have fast decaying singular-value spectrum, this approximation is often as good as the truncated SVD. The SRQR algorithm of [75] explicitly updates the partial QR factorization (2.1) while swapping columns, but the SRQR algorithm can actually avoid any explicit computations on the trailing matrix R_{22} using TRQRCP instead of RQRCP to obtain exactly the same partial QR initialization. Below we outline the SRQR algorithm.

In (2.1), let

$$\widetilde{R} \overset{def}{=} \begin{bmatrix} R_{11} & a \\ & \alpha \end{bmatrix} \tag{2.5}$$

be the leading $(l+1) \times (l+1)$ $(l \geq k)$ submatrix of R. We define

$$g_1 \overset{def}{=} \frac{\|R_{22}\|_{1,2}}{|\alpha|} \qquad \text{and} \qquad g_2 \overset{def}{=} |\alpha| \left\| \widetilde{R}^{-T} \right\|_{1,2}, \tag{2.6}$$

where $\|X\|_{1,2}$ is the largest column 2-norm of X for any given X. In [75], the authors proved approximation quality bounds involving g_1, g_2 for the low-rank approximation computed by RQRCP or TRQRCP. RQRCP or TRQRCP will provide a good low-rank matrix approximation if g_1 and g_2 are $O(1)$. The authors also proved that $g_1 \leq \sqrt{\frac{1+\varepsilon}{1-\varepsilon}}$ and $g_2 \leq \frac{\sqrt{2(1+\varepsilon)}}{1-\varepsilon} \left(1 + \sqrt{\frac{1+\varepsilon}{1-\varepsilon}}\right)^{l-1}$ hold with high probability for both RQRCP and TRQRCP, where $0 < \varepsilon < 1$ is a user-defined parameter which guides the choice of the oversampling size

ETNA
Kent State University and
Johann Radon Institute (RICAM)

p. For reasonably choices of ε like $\varepsilon = \frac{1}{2}$, g_1 is a small constant while g_2 can potentially be an extremely large number, which can lead to a poor low-rank approximation quality. To avoid the potential large number of g_2, the SRQR algorithm (Algorithm 2) proposed in [75] uses a pair-wise swapping strategy to guarantee that g_2 is below some user-defined tolerance $g > 1$ which is usually chosen to be a small number larger than one, like 2.0. In Algorithm 2, we use an approximate formula instead of using the definition (2.6) directly to estimate g_2, i.e., $g_2 \approx \frac{|\alpha|}{\sqrt{d}} \left\| \Omega \widetilde{R}^{-T} \right\|_{1,2}$, based on the Johnson-Lindenstrauss Lemma [36].

Algorithm 2 Spectrum-revealing QR Factorization (SRQR).

Inputs:
Matrix $A \in \mathbb{R}^{m \times n}$. Target rank k. Block size b. Oversampling size $p \geq 0$.
Integer $l \geq k$. Tolerance $g > 1$ for g_2.
Outputs:
Orthogonal matrix $Q \in \mathbb{R}^{m \times m}$ formed by the first k reflectors.
Upper trapezoidal matrix $R \in \mathbb{R}^{k \times n}$.
Permutation matrix $\Pi \in \mathbb{R}^{n \times n}$ such that $A\Pi \approx Q\,(:, 1:k)\,R$.
Algorithm:
Compute Q, R, Π with RQRCP or TRQRCP to l steps.
Compute squared 2-norm of the columns of $B(:, l+1:n) : \widehat{r}_i \ (l+1 \leq i \leq n)$, where B is an updated random projection of A computed by RQRCP or TRQRCP.
Approximate squared 2-norm of the columns of $A(l+1:m, l+1:n) : r_i = \widehat{r}_i/(b+p) \ (l+1 \leq i \leq n)$.
$\imath = \mathbf{argmax}_{l+1 \leq i \leq n}\{r_i\}$.
Swap $\imath$-th and $(l+1)$-st columns of A, Π, r.
One-step QR factorization of $A(l+1:m, l+1:n)$.
$|\alpha| = R_{l+1,l+1}$.
$r_i = r_i - A(l+1, i)^2 \ (l+2 \leq i \leq n)$.
Generate a random matrix $\Omega \in \mathcal{N}(0,1)^{d \times (l+1)} \ (d \ll l)$.
Compute $g_2 = |\alpha| \left\| \widetilde{R}^{-T} \right\|_{1,2} \approx \frac{|\alpha|}{\sqrt{d}} \left\| \Omega \widetilde{R}^{-T} \right\|_{1,2}$.

while $g_2 > g$ **do**
  $\imath = \mathbf{argmax}_{1 \leq i \leq l+1}\{i\text{th column norm of } \Omega \widetilde{R}^{-T}\}$.
  Swap $\imath$-th and $(l+1)$-st columns of A and Π in a Round Robin rotation.
  Givens-rotate R back into upper-trapezoidal form.
  $r_{l+1} = R_{l+1,l+1}^2$, $r_i = r_i + A(l+1, i)^2 \ (l+2 \leq i \leq n)$.
  $\imath = \mathbf{argmax}_{l+1 \leq i \leq n}\{r_i\}$.
  Swap $\imath$-th and $(l+1)$-st columns of A, Π, r.
  One-step QR factorization of $A(l+1:m, l+1:n)$.
  $|\alpha| = R_{l+1,l+1}$.
  $r_i = r_i - A(l+1, i)^2 \ (l+2 \leq i \leq n)$.
  Generate a random matrix $\Omega \in \mathcal{N}(0,1)^{d \times (l+1)} (d \ll l)$.
  Compute $g_2 = |\alpha| \left\| \widetilde{R}^{-T} \right\|_{1,2} \approx \frac{|\alpha|}{\sqrt{d}} \left\| \Omega \widetilde{R}^{-T} \right\|_{1,2}$.
end while

2.3. Tensor approximation. In this section we review some basic notations and concepts involving tensors. A more detailed discussion of the properties and applications of tensors can

ETNA
Kent State University and
Johann Radon Institute (RICAM)

be found in the review [40]. A tensor is a d-dimensional array of numbers denoted by script notation $\mathcal{X} \in \mathbb{R}^{I_1 \times \cdots \times I_d}$ with entries given by

$$x_{j_1,\ldots,j_d}, \quad 1 \le j_1 \le I_1, \ldots, 1 \le j_d \le I_d.$$

We use the matrix $X_{(n)} \in \mathbb{R}^{I_n \times (\Pi_{j \ne n} I_j)}$ to denote the nth mode unfolding of the tensor $\mathcal{X}$. Since this tensor has d dimensions, there are altogether d-possibilities for unfolding. The n-mode product of a tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times \cdots \times I_d}$ with a matrix $U \in \mathbb{R}^{k \times I_n}$ results in a tensor $\mathcal{Y} \in \mathbb{R}^{I_1 \times \cdots \times I_{n-1} \times k \times I_{n+1} \times \cdots \times I_d}$, such that

$$y_{j_1,\ldots,j_{n-1},j,j_{n+1},\ldots,j_d} = (\mathcal{X} \times_n U)_{j_1,\ldots,j_{n-1},j,j_{n+1},\ldots,j_d} = \sum_{j_n=1}^{I_n} x_{j_1,\ldots,j_d} u_{j,j_n}.$$

Alternatively it can be expressed conveniently in terms of unfolded tensors:

$$\mathcal{Y} = \mathcal{X} \times_n U \Leftrightarrow Y_{(n)} = U X_{(n)}.$$

Decompositions of higher-order tensors have applications in signal processing [12, 15, 63], numerical linear algebra [13, 39, 76], computer vision [62, 72, 67], etc. Two particular tensor decompositions can be considered as higher-order extensions of the matrix SVD: CANDECOMP/PARAFAC (CP) [10, 31] decomposes a tensor as a sum of rank-one tensors, and the Tucker decomposition [70] is a higher-order form of the principal component analysis. Knowing the definitions of mode products and unfolding of tensors, we can introduce the sequentially truncated higher-order SVD (ST-HOSVD) algorithm for producing a rank $(k_1, \ldots, k_d)$ approximation to the tensor based on the Tucker decomposition format. The ST-HOSVD algorithm [3, 71] (Algorithm 6 in the appendix) returns a core tensor $\mathcal{G} \in \mathbb{R}^{k_1 \times \cdots \times k_d}$ and a set of unitary matrices $U_j \in \mathbb{R}^{I_j \times k_j}$, for $j = 1, \ldots, d$, such that

$$\mathcal{X} \approx \mathcal{G} \times_1 U_1 \cdots \times_d U_d,$$

where the right-hand side is called a Tucker decomposition. However, a straightforward generalization to higher-order ($d \ge 3$) tensors of the matrix Eckart-Young-Mirsky Theorem is not possible [14].

In ST-HOSVD, one key step is to compute an exact or approximate rank-k_r SVD of the tensor unfolding. Well known efficient ways to compute an exact low-rank SVD include Krylov subspace methods [44]. There are also efficient randomized algorithms to find an approximate low-rank SVD [30]. In the Matlab tensorlab toolbox [73], the most efficient method, MLSVD_RSI, is essentially ST-HOSVD with an randomized subspace iteration to find the approximate SVD of the tensor unfolding.

2.4. Nuclear norm minimization. Matrix rank minimization problem appears ubiquitously in many fields such as Euclidean embedding [20, 46], control [21, 51, 56], collaborative filtering [9, 57, 65], system identification [47, 48], etc. The regularized nuclear norm minimization problem is the following:

$$\min_{X \in \mathbb{R}^{m \times n}} f(X) + \tau \|X\|_* ,$$

where $\tau > 0$ is a regularization parameter and $\|X\|_* \overset{def}{=} \sum_{i=1}^{q} \sigma_i(X)$. The choice of the function $f(\cdot)$ is situational: $f(X) = \|M - X\|_1$ in robust principal component analysis (robust PCA) [8] and $f(X) = \|\pi_\Omega(M) - \pi_\Omega(X)\|_F^2$ in matrix completion [7].

ETNA
Kent State University and
Johann Radon Institute (RICAM)

Many researchers have devoted themselves to solve the above nuclear norm minimization problem and plenty of algorithms have been proposed including singular value thresholding (SVT) [7], fixed point continuous (FPC) [49], accelerated proximal gradient (APG) [68], augmented Lagrange multiplier (ALM) [45]. The most expensive part of these algorithms lies in the computation of the truncated SVD. Inexact augmented Lagrange multiplier (IALM) [45] has been proved to be one of the most accurate and efficient among them. We now describe IALM for the robust PCA and matrix completion problems.

The robust PCA problem can be formalized as a minimization problem of a sum of nuclear norm and a scaled matrix l_1-norm (sum of matrix entries in absolute value):

$$\min \|X\|_* + \lambda \|E\|_1, \qquad \text{subject to} \qquad M = X + E, \tag{2.7}$$

where M is a measured matrix, X has low-rank, E is an error matrix and sufficiently sparse, and λ is a positive weighting parameter. Algorithm 7 in the appendix describes details of the IALM method to solve the robust PCA problem [45].

The matrix completion problem [9, 45] can be written in the form:

$$\min_{X \in \mathbb{R}^{m \times n}} \|X\|_* \qquad \text{subject to} \qquad X + E = M, \quad \pi_\Omega(E) = 0, \tag{2.8}$$

where $\pi_\Omega : \mathbb{R}^{m \times n} \to \mathbb{R}^{m \times n}$ is an orthogonal projection that keeps the entries in Ω unchanged and sets those outside Ω zeros. In [45], the authors applied the IALM method to the matrix completion problem (Algorithm 8 in the appendix).

3. Flip-Flop SRQR factorization.

3.1. Flip-Flop SRQR factorization. In this section, we introduce our Flip-Flop SRQR factorization, a slightly different variant of TUXV (Algorithm 5 in the appendix), to compute an SVD approximation based on the QLP factorization. Given an integer $l \geq k$, we run SRQR (the version without computing the trailing matrix) with l steps on A,

$$A\Pi = QR = Q \begin{bmatrix} R_{11} & R_{12} \\ & R_{22} \end{bmatrix}, \tag{3.1}$$

where $R_{11} \in \mathbb{R}^{l \times l}$ is upper triangular, $R_{12} \in \mathbb{R}^{l \times (n-l)}$, and $R_{22} \in \mathbb{R}^{(m-l) \times (n-l)}$. Then we run partial QRNP with l steps on R^T,

$$R^T = \begin{bmatrix} R_{11}^T & \\ R_{12}^T & R_{22}^T \end{bmatrix} = \widehat{Q} \begin{bmatrix} \widehat{R}_{11} & \widehat{R}_{12} \\ & \widehat{R}_{22} \end{bmatrix} \approx \widehat{Q}_1 \begin{bmatrix} \widehat{R}_{11} & \widehat{R}_{12} \end{bmatrix}, \tag{3.2}$$

where $\widehat{Q} = \begin{bmatrix} \widehat{Q}_1 & \widehat{Q}_2 \end{bmatrix}$ with $\widehat{Q}_1 \in \mathbb{R}^{n \times l}$. Therefore, combining with the fact that

$$A\Pi\widehat{Q}_1 = Q \begin{bmatrix} \widehat{R}_{11} & \widehat{R}_{12} \end{bmatrix}^T,$$

we can approximate the matrix A by

$$A = QR\Pi^T = Q\left(R^T\right)^T \Pi^T \approx Q \begin{bmatrix} \widehat{R}_{11}^T \\ \widehat{R}_{12}^T \end{bmatrix} \widehat{Q}_1^T \Pi^T = A\left(\Pi\widehat{Q}_1\right)\left(\Pi\widehat{Q}_1\right)^T. \tag{3.3}$$

We denote the rank-k truncated SVD of $A\Pi\widehat{Q}_1$ by $\widetilde{U}_k \Sigma_k \widetilde{V}_k^T$. Let $U_k = \widetilde{U}_k$, $V_k = \Pi\widehat{Q}_1\widetilde{V}_k$, then using (3.3), a rank-k approximate SVD of A is obtained:

$$A \approx U_k \Sigma_k V_k^T, \tag{3.4}$$

where $U_k \in \mathbb{R}^{m \times k}$, $V_k \in \mathbb{R}^{n \times k}$ are column orthonormal and $\Sigma_k = \mathrm{Diag}(\sigma_1, \cdots, \sigma_k)$ with σ_i's being the leading k singular values of $A\Pi\widehat{Q}_1$. The Flip-Flop SRQR factorization is outlined in Algorithm 3.

ETNA
Kent State University and
Johann Radon Institute (RICAM)

Algorithm 3 Flip-Flop Spectrum-Revealing QR Factorization.

Inputs:

Matrix $A \in \mathbb{R}^{m\times n}$. Target rank k. Block size b. Oversampling size $p \geq 0$.
Integer $l \geq k$. Tolerance $g > 1$ for g_2.

Outputs:

$U \in \mathbb{R}^{m\times k}$ contains the approximate top k left singular vectors of A.
$\Sigma \in \mathbb{R}^{k\times k}$ contains the approximate top k singular values of A.
$V \in \mathbb{R}^{n\times k}$ contains the approximate top k right singular vectors of A.

Algorithm:

Run SRQR on A to l steps to obtain (R_{11}, R_{12}).
Run QRNP on $(R_{11}, R_{12})^T$ to obtain $\widehat{Q}_1$, represented by a sequence of Householder vectors.

$\widehat{A} = A\Pi\widehat{Q}_1$.
$[\widehat{U}, \widehat{\Sigma}, \widehat{V}] = svd\left(\widehat{A}\right)$.
$U = \widehat{U}\left(:, 1:, k\right), \Sigma = \widehat{\Sigma}\left(1:k, 1:k\right), V = \Pi\widehat{Q}_1\widehat{V}\left(:, 1:k\right)$.

3.2. Complexity analysis. In this section, we perform a complexity analysis of Flip-Flop SRQR. Since the approximate SVD only makes sense when the target rank k is small, we assume that $k \leq l \ll \min(m, n)$. The complexity analysis of Flip-Flop SRQR looks as follows:

1. The main cost of doing SRQR with TRQRCP on A is $2mnl+2(b+p)mn+(m+n)l^2$.
2. The main cost of the QR factorization for $[R_{11}, R_{12}]^T$ and forming $\widehat{Q}_1$ is $2nl^2 - \frac{2}{3}l^3$.
3. The main cost of computing $\widehat{A} = A\Pi\widehat{Q}_1$ is $2mnl$.
4. The main cost of computing $[U, \sim, \sim] = svd\left(\widehat{A}\right)$ is $O(ml^2)$.
5. The main cost of forming V_k is $2nlk$.

Since $k \leq l \ll \min(m, n)$, the complexity of Flip-Flop SRQR is $4mnl + 2(b + p)mn$ by omitting the lower-order terms.

On the other hand, the complexity of the approximate SVD with randomized subspace iteration (RSISVD) [27, 30] is $(4 + 4q)\, mn\, (k + p)$, where p is the oversampling size and q is the number of subspace iterations (see the detailed analysis in the appendix). In practice, p is chosen to be a small integer, for instance, 5 in RSISVD. In Flip-Flop SRQR, l is usually chosen to be a little bit larger than k, e.g., $l = k + 5$. We also found that $l = k$ is sufficient in terms of the approximation quality in our numerical experiments. Therefore, we observe that Flip-Flop SRQR is more efficient than RSISVD for any $q > 0$.

Since a practical dataset matrix can be too large to fit into the main memory, the cost of transferring the matrix into a memory hierarchy typically predominates the arithmetic cost. Therefore, we also compare the number of data passes between RSISVD and Flip-Flop SRQR. RSISVD needs $2q + 2$ passes, while Flip-Flop SRQR needs 4 passes: computing $B = \Omega A$, swapping the columns of A, realizing a partial QR factorization on A, and computing $A\Pi\widehat{Q}_1$. Therefore, Flip-Flop SRQR requires less data transfer than RSISVD for any $q > 1$.

3.3. Quality analysis of Flip-Flop SRQR. This section is devoted to the quality analysis of Flip-Flop SRQR. We start with Lemma 3.1.

LEMMA 3.1. *Given any matrix* $X = [X_1 X_2]$ *with* $X_i \in \mathbb{R}^{m\times n_i}$, $i = 1, 2$, *and* $n_1 + n_2 = n$,

$$\sigma_j(X)^2 \leq \sigma_j(X_1)^2 + \|X_2\|_2^2, \quad 1 \leq j \leq \min(m, n).$$

ETNA
Kent State University and
Johann Radon Institute (RICAM)

Proof. Since $XX^T = X_1X_1^T + X_2X_2^T$, we obtain the above result using [33, Theorem 3.3.16]. □

We are now ready to derive bounds for the singular values and the approximation error of Flip-Flop SRQR. We need to emphasize that even if the target rank is k, we run Flip-Flop SRQR with an actual target rank l which is a little bit larger than k. The difference between k and l can create a gap between the singular values of A so that we can obtain a reliable low-rank approximation.

THEOREM 3.2. *Given a matrix $A \in \mathbb{R}^{m\times n}$, a target rank k, an oversampling size p, and an actual target rank $l \geq k$, the matrices U_k, Σ_k, V_k in (3.4) computed by Flip-Flop SRQR satisfy*

$$\sigma_j(\Sigma_k) \geq \frac{\sigma_j(A)}{\sqrt[4]{1+\frac{2\|R_{22}\|_2^4}{\sigma_j^4(\Sigma_k)}}}, \quad 1 \leq j \leq k, \tag{3.5}$$

and

$$\left\|A - U_k\Sigma_k V_k^T\right\|_2 \leq \sigma_{k+1}(A)\sqrt[4]{1+2\left(\frac{\|R_{22}\|_2}{\sigma_{k+1}(A)}\right)^4}, \tag{3.6}$$

where $R_{22} \in \mathbb{R}^{(m-l)\times(n-l)}$ is the trailing matrix in (3.1). Using the properties of SRQR, we furthermore have

$$\sigma_j(\Sigma_k) \geq \frac{\sigma_j(A)}{\sqrt[4]{1+\min\left(2\widehat{\tau}^4, \tau^4(2+4\widehat{\tau}^4)\left(\frac{\sigma_{l+1}(A)}{\sigma_j(A)}\right)^4\right)}}, \quad 1 \leq j \leq k, \tag{3.7}$$

and

$$\left\|A - U_k\Sigma_k V_k^T\right\|_2 \leq \sigma_{k+1}(A)\sqrt[4]{1+2\tau^4\left(\frac{\sigma_{l+1}(A)}{\sigma_{k+1}(A)}\right)^4}, \tag{3.8}$$

where τ and $\widehat{\tau}$ defined in (3.12) have matrix dimension-dependent upper bounds:

$$\tau \leq g_1g_2\sqrt{(l+1)(n-l)} \qquad \text{and} \qquad \widehat{\tau} \leq g_1g_2\sqrt{l(n-l)},$$

where $g_1 \leq \sqrt{\frac{1+\varepsilon}{1-\varepsilon}}$ and $g_2 \leq g$ with high probability. The parameters $\varepsilon > 0$ and $g > 1$ are user-defined.

Proof. In terms of the singular value bounds, observing that

$$\begin{bmatrix}\widehat{R}_{11} & \widehat{R}_{12}\\ & \widehat{R}_{22}\end{bmatrix}\begin{bmatrix}\widehat{R}_{11} & \widehat{R}_{12}\\ & \widehat{R}_{22}\end{bmatrix}^T = \begin{bmatrix}\widehat{R}_{11}\widehat{R}_{11}^T + \widehat{R}_{12}\widehat{R}_{12}^T & \widehat{R}_{12}\widehat{R}_{22}^T\\ \widehat{R}_{22}\widehat{R}_{12}^T & \widehat{R}_{22}\widehat{R}_{22}^T\end{bmatrix},$$

ETNA
Kent State University and
Johann Radon Institute (RICAM)

we apply Lemma 3.1 twice for any $1 \le j \le k$,

$$\begin{aligned}
&\sigma_j^2\left(\begin{bmatrix}\widehat{R}_{11} & \widehat{R}_{12}\\ & \widehat{R}_{22}\end{bmatrix}\begin{bmatrix}\widehat{R}_{11} & \widehat{R}_{12}\\ & \widehat{R}_{22}\end{bmatrix}^T\right)\\
&\quad\le \sigma_j^2\left(\begin{bmatrix}\widehat{R}_{11}\widehat{R}_{11}^T+\widehat{R}_{12}\widehat{R}_{12}^T & \widehat{R}_{12}\widehat{R}_{22}^T\end{bmatrix}\right)+\left\|\begin{bmatrix}\widehat{R}_{22}\widehat{R}_{12}^T & \widehat{R}_{22}\widehat{R}_{22}^T\end{bmatrix}\right\|_2^2\\
&\quad\le \sigma_j^2\left(\widehat{R}_{11}\widehat{R}_{11}^T+\widehat{R}_{12}\widehat{R}_{12}^T\right)+\left\|\widehat{R}_{12}\widehat{R}_{22}^T\right\|_2^2+\left\|\begin{bmatrix}\widehat{R}_{22}\widehat{R}_{12}^T & \widehat{R}_{22}\widehat{R}_{22}^T\end{bmatrix}\right\|_2^2\\
&\quad\le \sigma_j^2\left(\widehat{R}_{11}\widehat{R}_{11}^T+\widehat{R}_{12}\widehat{R}_{12}^T\right)+2\left\|\begin{bmatrix}\widehat{R}_{12}\\ \widehat{R}_{22}\end{bmatrix}\right\|_2^4\\
&\quad= \sigma_j^2\left(\widehat{R}_{11}\widehat{R}_{11}^T+\widehat{R}_{12}\widehat{R}_{12}^T\right)+2\left\|R_{22}\right\|_2^4. \qquad (3.9)
\end{aligned}$$

The relation (3.9) can be further rewritten as

$$\sigma_j^4(A)\le\sigma_j^4\left(\begin{bmatrix}\widehat{R}_{11} & \widehat{R}_{12}\end{bmatrix}\right)+2\left\|R_{22}\right\|_2^4=\sigma_j^4(\Sigma_k)+2\left\|R_{22}\right\|_2^4,\quad 1\le j\le k,$$

which is equivalent to

$$\sigma_j(\Sigma_k)\ge\frac{\sigma_j(A)}{\sqrt[4]{1+\frac{2\|R_{22}\|_2^4}{\sigma_j^4(\Sigma_k)}}}.$$

For the residual matrix bound, we let

$$\begin{bmatrix}\widehat{R}_{11} & \widehat{R}_{12}\end{bmatrix}\overset{def}{=}\begin{bmatrix}\overline{R}_{11} & \overline{R}_{12}\end{bmatrix}+\begin{bmatrix}\delta\overline{R}_{11} & \delta\overline{R}_{12}\end{bmatrix},$$

where $\begin{bmatrix}\overline{R}_{11} & \overline{R}_{12}\end{bmatrix}$ is the rank-k truncated SVD of $\begin{bmatrix}\widehat{R}_{11} & \widehat{R}_{12}\end{bmatrix}$. Since

$$\left\|A-U_k\Sigma_kV_k^T\right\|_2=\left\|A\Pi-Q\begin{bmatrix}\overline{R}_{11} & \overline{R}_{12}\\ & 0\end{bmatrix}^T\widehat{Q}^T\right\|_2, \qquad (3.10)$$

it follows from the orthogonality of singular vectors that

$$\begin{bmatrix}\overline{R}_{11} & \overline{R}_{12}\end{bmatrix}^T\begin{bmatrix}\delta\overline{R}_{11} & \delta\overline{R}_{12}\end{bmatrix}=0,$$

and therefore

$$\begin{aligned}
\begin{bmatrix}\widehat{R}_{11} & \widehat{R}_{12}\end{bmatrix}^T\begin{bmatrix}\widehat{R}_{11} & \widehat{R}_{12}\end{bmatrix}&=\begin{bmatrix}\overline{R}_{11} & \overline{R}_{12}\end{bmatrix}^T\begin{bmatrix}\overline{R}_{11} & \overline{R}_{12}\end{bmatrix}\\
&\quad+\begin{bmatrix}\delta\overline{R}_{11} & \delta\overline{R}_{12}\end{bmatrix}^T\begin{bmatrix}\delta\overline{R}_{11} & \delta\overline{R}_{12}\end{bmatrix},
\end{aligned}$$

which implies

$$\widehat{R}_{12}^T\widehat{R}_{12}=\overline{R}_{12}^T\overline{R}_{12}+\left(\delta\overline{R}_{12}\right)^T\left(\delta\overline{R}_{12}\right). \qquad (3.11)$$

Similarly to the deduction of (3.9), from (3.11) we can derive

$$\begin{aligned}
\left\|\begin{bmatrix}\delta\overline{R}_{11} & \delta\overline{R}_{12}\\ & \widehat{R}_{22}\end{bmatrix}\right\|_2^4&\le\left\|\begin{bmatrix}\delta\overline{R}_{11} & \delta\overline{R}_{12}\end{bmatrix}\right\|_2^4+2\left\|\begin{bmatrix}\delta\overline{R}_{12}\\ \widehat{R}_{22}\end{bmatrix}\right\|_2^4\\
&\le\sigma_{k+1}^4(A)+2\left\|\begin{bmatrix}\widehat{R}_{12}\\ \widehat{R}_{22}\end{bmatrix}\right\|_2^4=\sigma_{k+1}^4(A)+2\left\|R_{22}\right\|_2^4.
\end{aligned}$$

ETNA
Kent State University and
Johann Radon Institute (RICAM)

Combining with (3.10), it now follows that

$$\left\|A-U_k\Sigma_k V_k^T\right\|_2=\left\|A\Pi-Q\begin{bmatrix}\overline{R}_{11} & \overline{R}_{12}\\ & 0\end{bmatrix}^T\widehat{Q}^T\right\|_2=\left\|\begin{bmatrix}\delta\overline{R}_{11} & \delta\overline{R}_{12}\\ & \widehat{R}_{22}\end{bmatrix}\right\|_2$$
$$\leq\sigma_{k+1}(A)\sqrt[4]{1+2\left(\frac{\|R_{22}\|_2}{\sigma_{k+1}(A)}\right)^4}.$$

To obtain an upper bound of $\|R_{22}\|_2$ in (3.5) and (3.6), we follow the analysis of SRQR in [75].

From the analysis of [75, Section IV], Algorithm 2 ensures that $g_1\leq\sqrt{\frac{1+\varepsilon}{1-\varepsilon}}$ and $g_2\leq g$ with high probability (the actual probability-guarantee formula can be found in [75, Section IV]), where g_1 and g_2 are defined by (2.6). Here $0<\varepsilon<1$ is a user defined parameter to adjust the choice of the oversampling size p used in the TRQRCP initialization part in SRQR. $g>1$ is a user-defined parameter in the extra swapping part in SRQR. Let

$$\tau\overset{def}{=}g_1g_2\frac{\|R_{22}\|_2}{\|R_{22}\|_{1,2}}\frac{\left\|\widetilde{R}^{-T}\right\|_{1,2}^{-1}}{\sigma_{l+1}(A)}\quad\text{and}\quad\widehat{\tau}\overset{def}{=}g_1g_2\frac{\|R_{22}\|_2}{\|R_{22}\|_{1,2}}\frac{\left\|R_{11}^{-T}\right\|_{1,2}^{-1}}{\sigma_k(\Sigma_k)},\tag{3.12}$$

where $\widetilde{R}$ is defined in equation (2.5). Since $\frac{1}{\sqrt{n}}\|X\|_{1,2}\leq\|X\|_2\leq\sqrt{n}\|X\|_{1,2}$ and $\sigma_i(X_1)\leq\sigma_i(X),1\leq i\leq\min(s,t)$, for any matrix $X\in\mathbb{R}^{m\times n}$ and submatrix $X_1\in\mathbb{R}^{s\times t}$ of X, we obtain that

$$\tau=g_1g_2\frac{\|R_{22}\|_2}{\|R_{22}\|_{1,2}}\frac{\left\|\widetilde{R}^{-T}\right\|_{1,2}^{-1}}{\sigma_{l+1}\left(\widetilde{R}\right)}\frac{\sigma_{l+1}\left(\widetilde{R}\right)}{\sigma_{l+1}(A)}\leq g_1g_2\sqrt{(l+1)(n-l)}.$$

Using the fact that $\sigma_l\left(\begin{bmatrix}R_{11} & R_{12}\end{bmatrix}\right)=\sigma_l\left(\widehat{R}_{11}\right)$ and $\sigma_k(\Sigma_k)=\sigma_k\left(\begin{bmatrix}\widehat{R}_{11} & \widehat{R}_{12}\end{bmatrix}\right)$ by (3.2) and (3.4),

$$\widehat{\tau}=g_1g_2\frac{\|R_{22}\|_2}{\|R_{22}\|_{1,2}}\frac{\left\|R_{11}^{-T}\right\|_{1,2}^{-1}}{\sigma_l(R_{11})}\frac{\sigma_l(R_{11})}{\sigma_l\left(\begin{bmatrix}R_{11} & R_{12}\end{bmatrix}\right)}\frac{\sigma_l\left(\begin{bmatrix}R_{11} & R_{12}\end{bmatrix}\right)}{\sigma_k(\Sigma_k)}\leq g_1g_2\sqrt{l(n-l)}.$$

By the definition of τ,

$$\|R_{22}\|_2=\tau\,\sigma_{l+1}(A).\tag{3.13}$$

Plugging this into (3.6) yields (3.8).

By the definition of $\widehat{\tau}$, we observe that

$$\|R_{22}\|_2\leq\widehat{\tau}\sigma_k(\Sigma_k).\tag{3.14}$$

By (3.5) and (3.14),

(3.15)
$$\sigma_j(\Sigma_k)\geq\frac{\sigma_j(A)}{\sqrt[4]{1+2\left(\frac{\|R_{22}\|_2}{\sigma_j(\Sigma_k)}\right)^4}}\geq\frac{\sigma_j(A)}{\sqrt[4]{1+2\left(\frac{\|R_{22}\|_2}{\sigma_k(\Sigma_k)}\right)^4}}\geq\frac{\sigma_j(A)}{\sqrt[4]{1+2\widehat{\tau}^4}},\quad 1\leq j\leq k.$$

ETNA
Kent State University and
Johann Radon Institute (RICAM)

On the other hand, using (3.5),

$$\begin{aligned}\sigma_j^4(A) &\le \sigma_j^4(\Sigma_k)\left(1+2\frac{\sigma_j^4(A)}{\sigma_j^4(\Sigma_k)}\frac{\|R_{22}\|_2^4}{\sigma_j^4(A)}\right)\\ &\le \sigma_j^4(\Sigma_k)\left(1+2\frac{\left(\sigma_j^4(\Sigma_k)+2\|R_{22}\|_2^4\right)}{\sigma_j^4(\Sigma_k)}\frac{\|R_{22}\|_2^4}{\sigma_j^4(A)}\right)\\ &\le \sigma_j^4(\Sigma_k)\left(1+2\left(1+2\frac{\|R_{22}\|_2^4}{\sigma_k^4(\Sigma_k)}\right)\frac{\|R_{22}\|_2^4}{\sigma_j^4(A)}\right),\end{aligned}$$

that is,

$$\sigma_j(\Sigma_k) \ge \frac{\sigma_j(A)}{\sqrt[4]{1+\left(2+4\frac{\|R_{22}\|_2^4}{\sigma_k^4(\Sigma_k)}\right)\frac{\|R_{22}\|_2^4}{\sigma_j^4(A)}}}.$$

Plugging (3.13) and (3.14) into this above equation yields

$$\sigma_j(\Sigma_k) \ge \frac{\sigma_j(A)}{\sqrt[4]{1+\tau^4\left(2+4\widehat{\tau}^4\right)\left(\frac{\sigma_{l+1}(A)}{\sigma_j(A)}\right)^4}}, \quad 1 \le j \le k. \tag{3.16}$$

Combining (3.15) with (3.16), we arrive at (3.7). ∎

We note that (3.5) and (3.6) still hold true if we replace k by l.

Inequality (3.7) shows that with the definitions (3.12) of τ and $\widehat{\tau}$, Flip-Flop SRQR can reveal at least a dimension-dependent fraction of all the leading singular values of A and indeed approximate them very accurately in case they decay relatively quickly. Moreover, (3.8) shows that Flip-Flop SRQR can compute a rank-k approximation that is up to a factor of $\sqrt[4]{1+2\tau^4\left(\frac{\sigma_{l+1}(A)}{\sigma_{k+1}(A)}\right)^4}$ from optimal. In situations where the singular values of A decay relatively quickly, our rank-k approximation is about as accurate as the truncated SVD with a choice of l, such that

$$\frac{\sigma_{l+1}(A)}{\sigma_{k+1}(A)} = o(1).$$

REMARK 3.3. The "high probability" mentioned in Theorem 3.2 is dependent on the oversampling size p but not on the block size b. We decided not to dive deep into the probability explanation since it will make the proof much longer and more tedious. Theoretically, the computed oversampling size p for certain given high probability would be rather high. For example, to achieve a probability 0.95, we need to choose p to be at least 500. However, in practice, we would never choose such a large p since it will deteriorate the code efficiency. It turns out that a small p like $5 \sim 10$ would achieve good approximation results empirically. The detailed explanation about what does high probability mean here can be found in [75]. In terms of the choice of the block size b, it won't affect the analysis or the error bounds. The choice of b is purely for the performance of the codes. Only a suitable b can take full advantage of the cache hierarchy and BLAS routines.

4. Numerical experiments. In this section, we demonstrate the effectiveness and efficiency of the Flip-Flop SRQR (FFSRQR) algorithm in several numerical experiments. Firstly,

ETNA
Kent State University and
Johann Radon Institute (RICAM)

TABLE 4.1
Methods for approximate SVD.

Method	Description
LANSVD	Approximate SVD using Lanczos bidiagonalization with partial reorthogonalization [42]. Its Fortran and Matlab implementations are from PROPACK [41].
FFSRQR	Flip-Flop Spectrum-revealing QR factorization. Its Matlab implementation is using Mex-wrapped BLAS and LAPACK routines [2].
RSISVD	Approximate SVD with randomized subspace iteration [30, 27]. Its Matlab implementation is from tensorlab toolbox [73].
TUXV	Algorithm 5 in the appendix.
LTSVD	Linear Time SVD [16]. We use its Matlab implementation by Ma et al. [49].

we compare FFSRQR with other approximate SVD algorithms for a matrix approximation. Secondly, we compare FFSRQR with other methods on tensor approximation problems using the tensorlab toolbox [73]. Thirdly, we compare FFSRQR with other methods for the robust PCA problem and matrix completion problem. All experiments are implemented on a laptop with 2.9 GHz Intel Core i5 processor and 8 GB of RAM. The codes based on LAPACK [2] and PROPACK [41] in Section 4.1 are in Fortran. The codes in Sections 4.2, 4.3 are in Matlab, except that FFSRQR is implemented in Fortran and wrapped by mex.

4.1. Approximate truncated SVD. In this section, we compare FFSRQR with other approximate SVD algorithms for low-rank matrix approximation. All tested methods are listed in Table 4.1. Since LTSVD has only a Matlab implementation, this method is only considered in Sections 4.2 and 4.3. The test matrices are:

Type 1: $A \in \mathbb{R}^{m\times n}$ [66] is defined by $A = U\,D\,V^T + 0.1 \cdot d_{ss} \cdot E$, where $U \in \mathbb{R}^{m\times s}, V \in \mathbb{R}^{n\times s}$ are column-orthonormal matrices, and $D \in \mathbb{R}^{s\times s}$ is a diagonal matrix with s geometrically decreasing diagonal entries from 1 to 10^{-3}. d_{ss} is the last diagonal entry of D. $E \in \mathbb{R}^{m\times n}$ is a random matrix where the entries are independently sampled from a normal distribution $\mathcal{N}(0,1)$. In our numerical experiment, we test three different random matrices. The square matrix has a size of 10000×10000, the short-fat matrix has a size of 1000×10000, and the tall-skinny matrix has a size of 10000×1000.

Type 2: $A \in \mathbb{R}^{4929\times 4929}$ is a real data matrix from the University of Florida sparse matrix collection [11]. Its corresponding file name is HB/GEMAT11.

For a given matrix $A \in \mathbb{R}^{m\times n}$, the relative SVD approximation error is measured by $\left\|A - U_k\Sigma_k V_k^T\right\|_F / \|A\|_F$ where Σ_k contains the approximate top k singular values, and U_k, V_k are the corresponding approximate top k singular vectors. The parameters used in FFSRQR, RSISVD, TUXV, and LTSVD are listed in Table 4.2. The additional time required to compute g_2 is negligible. In our experiments, g_2 always remains modest and never triggers subsequent SRQR column swaps. However, computing g_2 is still recommended since g_2 serves as an insurance policy against potential quality degration by TRQRCP. The block size b is chosen to be 32 according to the xGEQP3 LAPACK routine. When k is less than 32, we implement an unblock version of the partial QRCP.

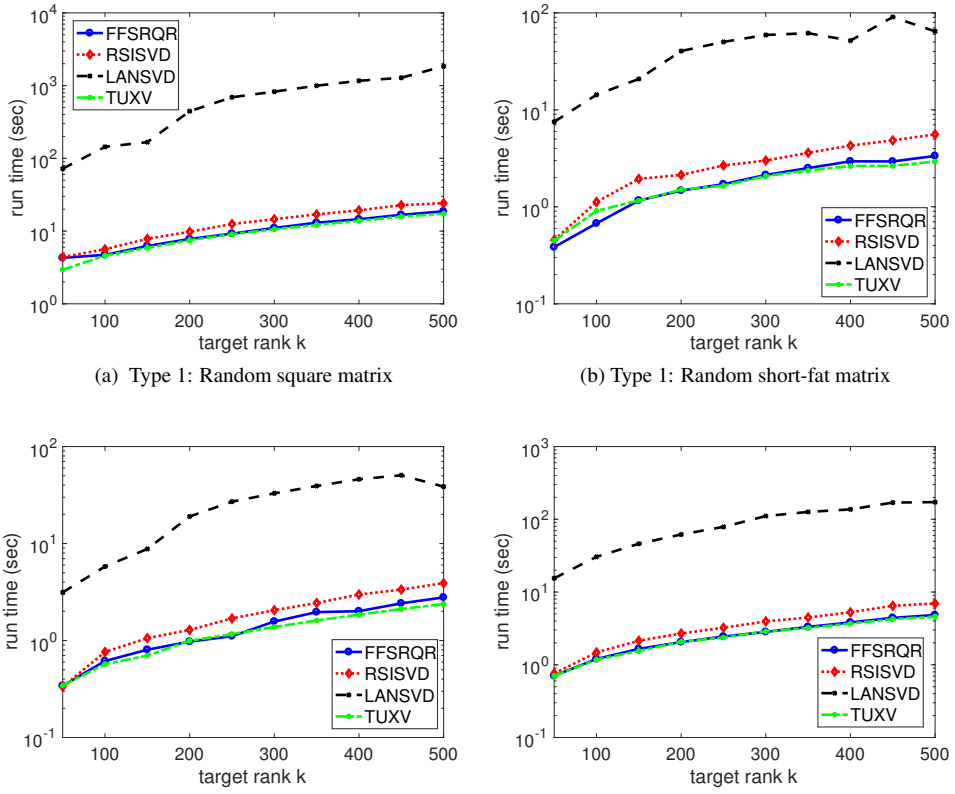
Figures 4.1–4.3 display the run time, the relative approximation error, and the top 20 singular values comparison respectively for four different matrices. In terms of speed, FFSRQR is always faster than LANSVD and RSISVD. Moreover, the efficiency difference is more

ETNA
Kent State University and
Johann Radon Institute (RICAM)

TABLE 4.2
Parameters used in RSISVD, FFSRQR, TUXV and LTSVD.

Method	Parameter
FFSRQR	oversampling size $p = 5$, block size $b = \min\{32, k\}$, $l = k,\ d = 10,\ g = 2.0$
RSISVD	oversampling size $p = 5$, subspace iteration $q = 1$
TUXV	oversampling size $p = 5$, block size $b = \min\{32, k\}$, $l = k$
LTSVD	probabilities $p_i = 1/n$

obvious when k is relatively large. FFSRQR is only slightly slower than TUXV. In terms of the relative approximation error, FFSRQR is comparable to the other three methods. In terms of the top singular values approximation, Figure 4.3 displays the top 20 singular values when $k = 500$ for four different matrices. In Figure 4.3, FFSRQR is superior to TUXV as we take the absolute values of the main diagonal entries of the upper triangle matrix R as the approximate singular values for TUXV.



(a) Type 1: Random square matrix

(b) Type 1: Random short-fat matrix

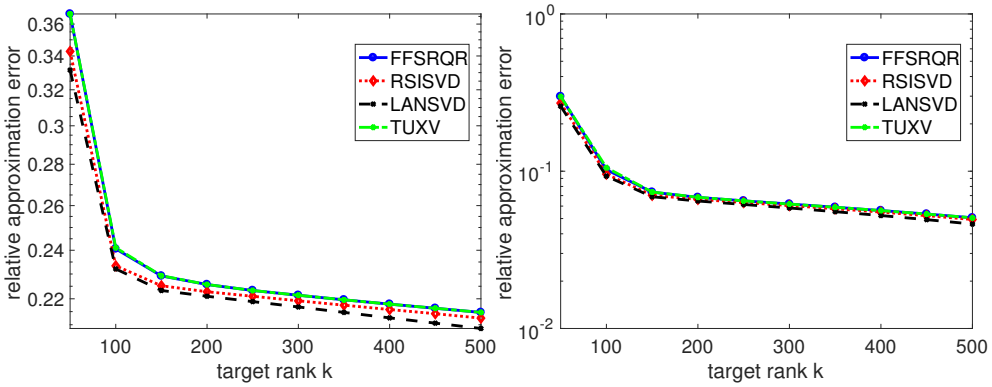
(c) Type 1: Random tall-skinny matrix

(d) Type 2: GEMAT11

FIG. 4.1. *Run time comparison for approximate SVD algorithms.*

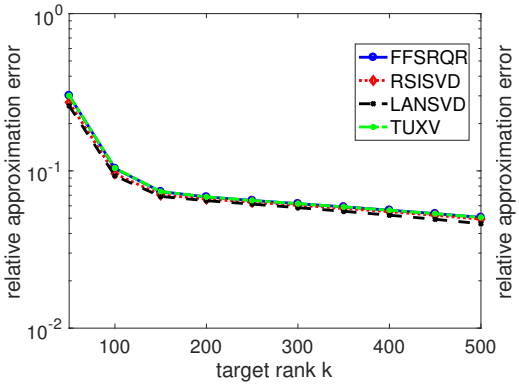
4.2. Tensor approximation. This section illustrates the effectiveness and efficiency of FFSRQR for computing approximate tensors. ST-HOSVD [3, 71] is one of the most efficient

ETNA
Kent State University and
Johann Radon Institute (RICAM)

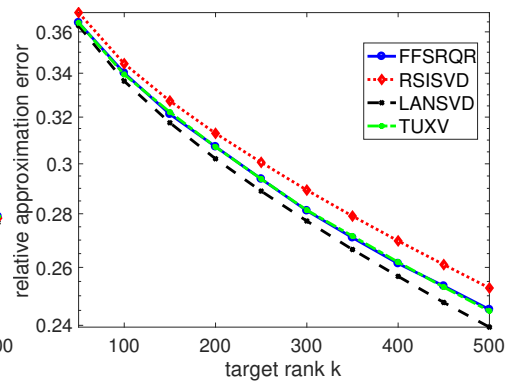


(a) Type 1: Random square matrix

(b) Type 1: Random short-fat matrix



(c) Type 1: Random tall-skinny matrix



(d) Type 2: GEMAT11

FIG. 4.2. *Relative approximation error comparison for approximate SVD algorithms.*

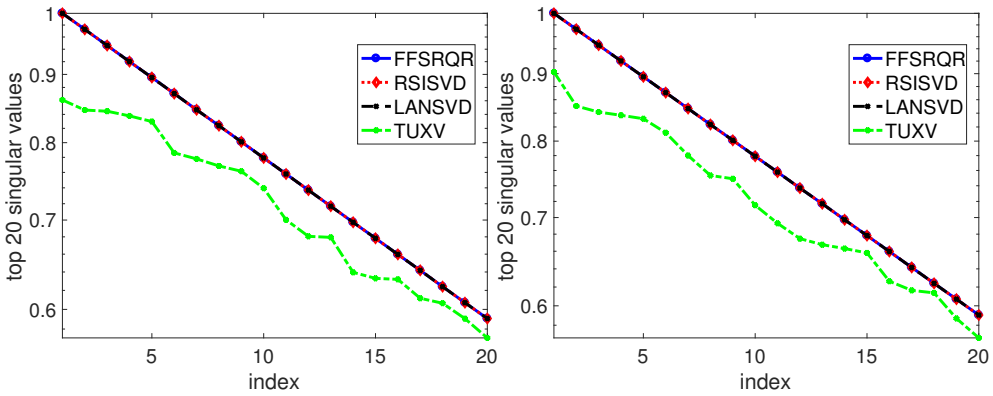
algorithms to compute a Tucker decomposition of tensors, and the most costly part of this algorithm is to compute an SVD or approximate SVD of the tensor unfoldings. The truncated SVD and RSISVD are used in the routines MLSVD and MLSVD_RSI, respectively, in the Matlab tensorlab toolbox [73]. Based on this Matlab toolbox, we implement ST-HOSVD using FFSRQR or LTSVD to do the SVD approximation. We name these two new routines by MLSVD_FFSRQR and MLSVD_LTSVD respectively, and compare these four routines in the numerical experiment. We also have Python codes for this tensor approximation experiment. We don't list the results of the python programs here as they are similar to those for Matlab.

4.2.1. A sparse tensor example. We test a sparse tensor $\mathcal{X} \in \mathbb{R}^{n\times n\times n}$ of the following format [64, 59],

$$\mathcal{X} = \sum_{j=1}^{10} \frac{1000}{j} x_j \circ y_j \circ z_j + \sum_{j=11}^{n} \frac{1}{j} x_j \circ y_j \circ z_j,$$

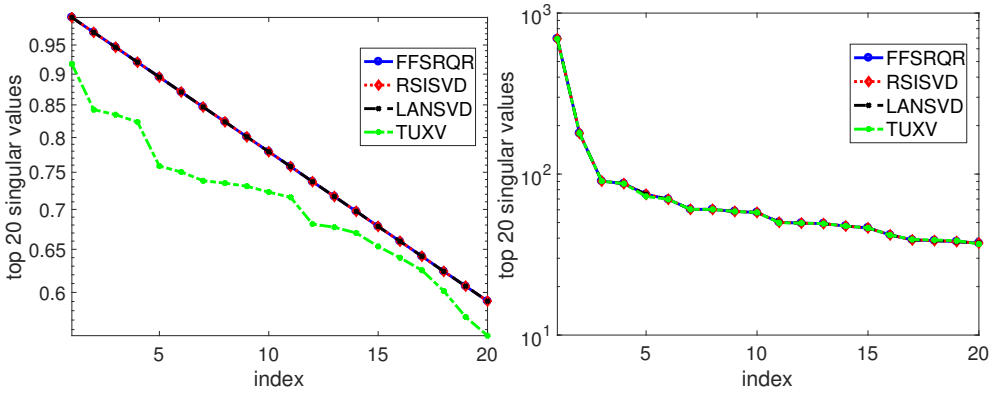
where $x_j, y_j, z_j \in \mathbb{R}^n$ are sparse vectors with nonnegative entries. The symbol "$\circ$" represents the vector outer product. We compute a rank-(k,k,k) Tucker decomposition $[\mathcal{G}; U_1, U_2, U_3]$ using MLSVD, MLSVD_FFSRQR, MLSVD_RSI, and MLSVD_LTSVD respectively. The relative approximation error is measured by $\|\mathcal{X} - \mathcal{X}_k\|_F / \|\mathcal{X}\|_F$ where $\mathcal{X}_k = \mathcal{G} \times_1 U_1 \times_2 U_2 \times_3 U_3$.

ETNA
Kent State University and
Johann Radon Institute (RICAM)



(a) Type 1: Random square matrix

(b) Type 1: Random short-fat matrix



(c) Type 1: Random tall-skinny matrix

(d) Type 2: GEMAT11

FIG. 4.3. *Top 20 singular values comparison for approximate SVD algorithms.*

Figure 4.4 displays a comparison of the efficiency and accuracy of different methods for a $400 \times 400 \times 400$ sparse tensor approximation problem. MLSVD_LTSVD is the fastest but the least accurate one. The other three methods have similar accuracy while MLSVD_FFSRQR is faster when the target rank k is larger.

4.2.2. Handwritten digits classification. MNIST is a handwritten digits image data set created by Yann LeCun [43]. Every digit is represented by a 28×28 pixel image. Handwritten digits classification has the objective of training a classification model to classify new unlabeled images. A HOSVD algorithm is proposed by Savas and Eldén [60] to classify handwritten digits. To reduce the training time, a more efficient ST-HOSVD algorithm is introduced in [71].

We do handwritten digits classification using MNIST which consists of $60,000$ training images and $10,000$ test images. The number of training images in each class is restricted to 5421 so that the training set is equally distributed over all classes. The training set is represented by a tensor $\mathcal{X}$ of size $786 \times 5421 \times 10$. The classification relies on Algorithm 2 in [60]. We use various algorithms to obtain an approximation $\mathcal{X} \approx \mathcal{G} \times_1 U_1 \times_2 U_2 \times_3 U_3$ where the core tensor $\mathcal{G}$ has size $65 \times 142 \times 10$.

The results are summarized in Table 4.3. In terms of run time, we observe that our method MLSVD_FFSRQR is comparable to MLSVD_RSI, while MLSVD is the most expensive

ETNA
Kent State University and
Johann Radon Institute (RICAM)

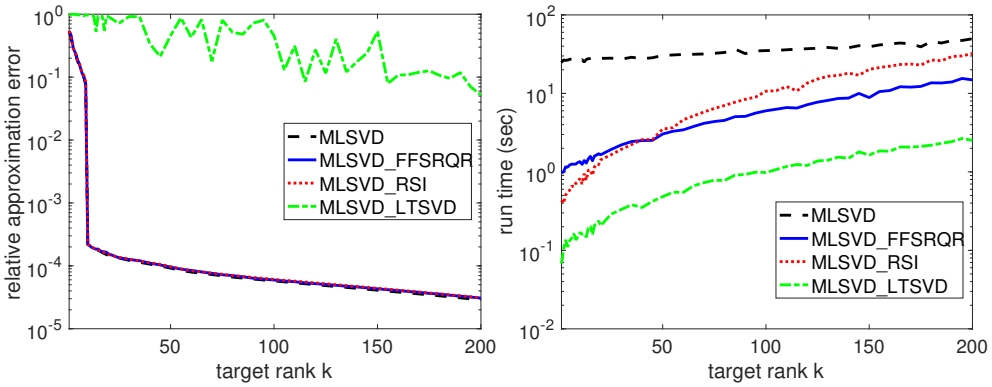


FIG. 4.4. *Run time and relative approximation error comparison on a sparse tensor.*

TABLE 4.3
Comparison on handwritten digits classification.

	MLSVD	MLSVD_FFSRQR	MLSVD_RSI	MLSVD_LTSVD
Training Time [sec]	27.2121	1.5455	1.9343	0.4266
Relative Model Error	0.4099	0.4273	0.4247	0.5162
Classification Accur.	95.19%	94.98%	95.05%	92.59%

one and MLSVD_LTSVD is the fastest one. In terms of classification quality, MLSVD, MLSVD_FFSRQR, and MLSVD_RSI are comparable, while MLSVD_LTSVD is the least accurate one.

4.3. Solving nuclear norm minimization problem. To show the effectiveness of the FFSRQR algorithm for the nuclear norm minimization problems, we investigate two scenarios: the robust PCA (2.7) and the matrix completion problem (2.8). The test matrix used for the robust PCA is introduced in [45], and the test matrices used in the matrix completion are two real data sets. We use the IALM method [45] to solve both problems; the IALM code can be downloaded[1].

4.3.1. Robust PCA. To solve the robust PCA problem, we replace the approximate SVD part in the IALM method [45] by various methods. We denote the actual solution to the robust PCA problem by a matrix pair $(X^*, E^*) \in \mathbb{R}^{m\times n} \times \mathbb{R}^{m\times n}$. The matrix X^* is $X^* = X_L X_R^T$, with $X_L \in \mathbb{R}^{m\times k}$, $X_R \in \mathbb{R}^{n\times k}$ being random matrices whose entries are independently sampled from a normal distribution. The sparse matrix E^* is a random matrix, where its non-zero entries are independently sampled from a uniform distribution over the interval $[-500, 500]$. The input to the IALM algorithm has the form $M = X^* + E^*$ and the output is denoted by $\left(\widehat{X}, \widehat{E}\right)$. In this numerical experiment, we use the same parameter settings as the IALM code for robust PCA: rank k is $0.1m$ and the number of non-zero entries in E is $0.05m^2$. We choose the trade-off parameter $\lambda = 1/\sqrt{\max(m,n)}$ as suggested by Candès et al. [8]. The solution quality is measured by the normalized root mean square error $\|\widehat{X} - X^*\|_F / \|X^*\|_F$.

Table 4.4 includes relative error, run time, the number of non-zero entries in $\widehat{E}$ ($\|\widehat{E}\|_0$), the iteration count, and the number of non-zero singular values (#sv) in $\widehat{X}$ of the IALM

[1] http://perception.csl.illinois.edu/matrix-rank/sample_code.html

ETNA
Kent State University and
Johann Radon Institute (RICAM)

algorithm using different approximate SVD methods. We observe that IALM_FFSRQR is faster than all the other three methods, while its error is comparable to IALM_LANSVD and IALM_RSISVD. IALM_LTSVD is relatively slow and not effective.

TABLE 4.4
Comparison on robust PCA.

Size	Method	Error	Time (sec)	$\|\widehat{E}\|_0$	Iter	#sv
1000×1000	IALM_LANSVD	3.33e-07	5.79e+00	50000	22	100
	IALM_FFSRQR	2.79e-07	1.02e+00	50000	25	100
	IALM_RSISVD	3.36e-07	1.09e+00	49999	22	100
	IALM_LTSVD	9.92e-02	3.11e+00	999715	100	100
2000×2000	IALM_LANSVD	2.61e-07	5.91e+01	199999	22	200
	IALM_FFSRQR	1.82e-07	6.93e+00	199998	25	200
	IALM_RSISVD	2.63e-07	7.38e+00	199996	22	200
	IALM_LTSVD	8.42e-02	2.20e+01	3998937	100	200
4000×4000	IALM_LANSVD	1.38e-07	4.65e+02	799991	23	400
	IALM_FFSRQR	1.39e-07	4.43e+01	800006	26	400
	IALM_RSISVD	1.51e-07	5.04e+01	799990	23	400
	IALM_LTSVD	8.94e-02	1.54e+02	15996623	100	400
6000×6000	IALM_LANSVD	1.30e-07	1.66e+03	1799982	23	600
	IALM_FFSRQR	1.02e-07	1.42e+02	1799993	26	600
	IALM_RSISVD	1.44e-07	1.62e+02	1799985	23	600
	IALM_LTSVD	8.58e-02	5.55e+02	35992605	100	600

4.3.2. Matrix completion. We solve the matrix completion problems for two real data sets used in [68]: the Jester joke data set [24] and the MovieLens data set [32]. The Jester joke data set consists of 4.1 million ratings for 100 jokes from $73,421$ users and can be downloaded from the website http://goldberg.berkeley.edu/jester-data/. We test with the following data matrices:

- **jester-**1: Data from $24,983$ users who have rated 36 or more jokes,
- **jester-**2: Data from $23,500$ users who have rated 36 or more jokes,
- **jester-**3: Data from $24,938$ users who have rated between 15 and 35 jokes,
- **jester-all**: The combination of **jester-**1, **jester-**2, and **jester-**3.

The MovieLens data set can be downloaded from

https://grouplens.org/datasets/movielens/.

We test with the following data matrices:

- **movie-**100**K**: $100,000$ ratings of 943 users for 1682 movies,
- **movie-**1**M**: 1 million ratings of 6040 users for 3900 movies,
- **movie-latest-small**: $100,000$ ratings of 700 users for 9000 movies.

For each data set, we let M be the original data matrix where M_{ij} stands for the rating of joke (movie) j by user i and Γ be the set of indices where M_{ij} is known. The matrix completion algorithm quality is measured by the Normalized Mean Absolute Error (NMAE) defined by

$$\text{NMAE} \stackrel{def}{=} \frac{\frac{1}{|\Gamma|}\sum_{(i,j)\in\Gamma}|M_{ij}-X_{ij}|}{r_{\max}-r_{\min}},$$

ETNA
Kent State University and
Johann Radon Institute (RICAM)

where X_{ij} is the prediction of the rating of joke (movie) j given by user i, and $r_{\min}, r_{\max}$ are the lower and upper bounds of the ratings, respectively. For the Jester joke data sets we set $r_{\min} = -10$ and $r_{\max} = 10$. For the MovieLens data sets we set $r_{\min} = 1$ and $r_{\max} = 5$.

Since $|\Gamma|$ is large, we randomly select a subset Ω from Γ and then use Algorithm 8 to solve the problem (2.8). We randomly select 10 ratings for each user in the Jester joke data sets, while we randomly choose about 50% of the ratings for each user in the MovieLens data sets. Table 4.5 includes the parameter settings in the algorithms. The maximum iteration number is 100 in IALM and all other parameters are the same as those used in [45].

The numerical results are included in Table 4.6. We observe that IALM_FFSRQR achieves almost the same recoverability as other methods (except IALM_LTSVD) and is slightly faster than IALM_RSISVD for these two data sets.

TABLE 4.5
Parameters used in the IALM method on matrix completion.

Data set	m	n	$\lvert\Gamma\rvert$	$\lvert\Omega\rvert$
jester-1	24983	100	1.81e+06	249830
jester-2	23500	100	1.71e+06	235000
jester-3	24938	100	6.17e+05	249384
jester-all	73421	100	4.14e+06	734210
movie-100K	943	1682	1.00e+05	49918
movie-1M	6040	3706	1.00e+06	498742
movie-latest-small	671	9066	1.00e+05	52551

5. Conclusions. We have presented the Flip-Flop SRQR factorization, a variant of the QLP factorization, to compute low-rank matrix approximations. The Flip-Flop SRQR algorithm uses a SRQR factorization to initialize the truncated version of a column-pivoted QR factorization and then forms an LQ factorization. For the numerical results presented, the errors in the proposed algorithm were comparable to those obtained from the other state-of-the-art algorithms. This new algorithm is cheaper to compute and produces quality low-rank matrix approximations. Furthermore, we prove singular value lower bounds and residual error upper bounds for the Flip-Flop SRQR factorization. In situations where singular values of the input matrix decay relatively quickly, the low-rank approximation computed by Flip-Flop SRQR is guaranteed to be as accurate as the truncated SVD. We also perform a complexity analysis to show that Flip-Flop SRQR is faster than the approximate SVD with randomized subspace iteration. Future work includes reducing the overhead cost in Flip-Flop SRQR and implementing the Flip-Flop SRQR algorithm on distributed memory machines for popular applications such as distributed PCA.

Acknowledgments. The authors would like to thank Yousef Saad, the editor, and three anonymous referees who kindly reviewed the earlier version of this manuscript and provided valuable suggestions and comments.

Appendix A. Partial QR factorization with column pivoting.
The details of partial QRCP are covered in Algorithm 4.

Appendix B. TUXV algorithm.
The details of TUXV algorithm are presented in Algorithm 5.

Appendix C. Computing Tucker decomposition of higher-order tensors.
Algorithm 6 computes the Tucker decomposition of higher-order tensors, i.e., sequentially truncated higher-order SVD.

ETNA
Kent State University and
Johann Radon Institute (RICAM)

TABLE 4.6
Comparison on matrix completion.

Data set	Method	Iter	Time	NMAE	#sv	$\sigma_{\max}$	σ_{min}
jester-1	IALM-LANSVD	12	7.06e+00	1.84e-01	100	2.14e+03	1.00e+00
	IALM-FFSRQR	12	3.44e+00	1.69e-01	100	2.28e+03	1.00e+00
	IALM-RSISVD	12	3.75e+00	1.89e-01	100	2.12e+03	1.00e+00
	IALM-LTSVD	100	2.11e+01	1.74e-01	62	3.00e+03	1.00e+00
jester-2	IALM-LANSVD	12	6.80e+00	1.85e-01	100	2.13e+03	1.00e+00
	IALM-FFSRQR	12	2.79e+00	1.70e-01	100	2.29e+03	1.00e+00
	IALM-RSISVD	12	3.59e+00	1.91e-01	100	2.12e+03	1.00e+00
	IALM-LTSVD	100	2.03e+01	1.75e-01	58	2.96e+03	1.00e+00
jester-3	IALM-LANSVD	12	7.05e+00	1.26e-01	99	1.79e+03	1.00e+00
	IALM-FFSRQR	12	3.03e+00	1.22e-01	100	1.71e+03	1.00e+00
	IALM-RSISVD	12	3.85e+00	1.31e-01	100	1.78e+03	1.00e+00
	IALM-LTSVD	100	2.12e+01	1.33e-01	55	2.50e+03	1.00e+00
jester-all	IALM-LANSVD	12	2.39e+01	1.72e-01	100	3.56e+03	1.00e+00
	IALM-FFSRQR	12	1.12e+01	1.62e-01	100	3.63e+03	1.00e+00
	IALM-RSISVD	12	1.34e+01	1.82e-01	100	3.47e+03	1.00e+00
	IALM-LTSVD	100	6.99e+01	1.68e-01	52	4.92e+03	1.00e+00
movie-100K	IALM-LANSVD	29	2.86e+01	1.83e-01	285	1.21e+03	1.00e+00
	IALM-FFSRQR	30	4.55e+00	1.67e-01	295	1.53e+03	1.00e+00
	IALM-RSISVD	29	4.82e+00	1.82e-01	285	1.29e+03	1.00e+00
	IALM-LTSVD	48	1.42e+01	1.47e-01	475	1.91e+03	1.00e+00
movie-1M	IALM-LANSVD	50	7.40e+02	1.58e-01	495	4.99e+03	1.00e+00
	IALM-FFSRQR	53	2.07e+02	1.37e-01	525	6.63e+03	1.00e+00
	IALM-RSISVD	50	2.23e+02	1.57e-01	495	5.35e+03	1.00e+00
	IALM-LTSVD	100	8.50e+02	1.17e-01	995	8.97e+03	1.00e+00
movie-latest -small	IALM-LANSVD	31	1.66e+02	1.85e-01	305	1.13e+03	1.00e+00
	IALM-FFSRQR	31	1.96e+01	2.00e-01	305	1.42e+03	1.00e+00
	IALM-RSISVD	31	2.85e+01	1.91e-01	305	1.20e+03	1.00e+00
	IALM-LTSVD	63	4.02e+01	2.08e-01	298	1.79e+03	1.00e+00

Appendix D. IALM for solving two special nuclear norm minimization problems.

Algorithm 7 and Algorithm 8 solve the robust PCA and matrix completion problems, respectively. In Algorithm 7, $\mathcal{S}_\omega(x) = \operatorname{sgn}(x) \cdot \max(|x| - \omega, 0)$ is the soft shrinkage operator [29] with $x \in \mathbb{R}^n$ and $\omega > 0$. In Algorithm 8, $\overline{\Omega}$ is the complement of Ω.

Appendix E. Approximate SVD with randomized subspace iteration.

The randomized subspace iteration was proposed in [30, Algorithm 4.4] to compute an orthonormal matrix whose range approximates the range of A. An approximate SVD can be computed using the aforementioned orthonormal matrix [30, Algorithm 5.1]. A randomized subspace iteration is used in routine MLSVD_RSI in Matlab toolbox tensorlab [73], and MLSVD_RSI is by far the most efficient function that we can find in Matlab to compute ST-HOSVD. We summarize the approximate SVD with randomized subspace iteration pseudocode in Algorithm 9.

Now we perform a complexity analysis for the approximate SVD with randomized subspace iteration. We first note that:

1. The cost of generating a random matrix is negligible.
2. The cost of computing $B = A\Omega$ is $2mn(k+p)$.

ETNA
Kent State University and
Johann Radon Institute (RICAM)

Algorithm 4 Partial QRCP.

Inputs:
Matrix $A \in \mathbb{R}^{m\times n}$. Target rank k.
Outputs:
Orthogonal matrix $Q \in \mathbb{R}^{m\times m}$.
Upper trapezoidal matrix $R \in \mathbb{R}^{m\times n}$.
Permutation matrix $\Pi \in \mathbb{R}^{n\times n}$ such that $A\Pi = QR$.
Algorithm:
Initialize $\Pi = I_n$. Compute column norms $r_s = \|A(1:m,s)\|_2 \ (1 \le s \le n)$.
for $j = 1:k$ **do**
  Find $i = \arg\max_{j\le s\le n} r_s$. Exchange r_j and r_i columns in A and Π.
  Form Householder reflection Q_j from $A(j:m,j)$.
  Update trailing matrix $A(j:m,j:n) \leftarrow Q_j^T A(j:m,j:n)$.
  Update $r_s = \|A(j+1:m,s)\|_2 \ (j+1 \le s \le n)$.
end for
$Q = Q_1Q_2\cdots Q_k$ is the product of all reflections. $R =$ upper trapezoidal part of A.

Algorithm 5 TUXV Algorithm.

Inputs:
Matrix $A \in \mathbb{R}^{m\times n}$. Target rank k. Block size b. Oversampling size $p \ge 0$.
Outputs:
Column orthonormal matrices $U \in \mathbb{R}^{m\times k}$, $V \in \mathbb{R}^{n\times k}$, and upper triangular matrix $R \in \mathbb{R}^{k\times k}$ such that $A \approx URV^T$.
Algorithm:
Do TRQRCP on A to obtain $Q \in \mathbb{R}^{m\times k}$, $R \in \mathbb{R}^{k\times n}$, and $\Pi \in \mathbb{R}^{n\times n}$.
$R = R\Pi^T$ and do LQ factorization, i.e., $[V,R] = qr(R^T,0)$.
Compute $Z = AV$ and do QR factorization, i.e., $[U,R] = qr(Z,0)$.

3. In each QR step $[Q,\sim] = qr(B,0)$, the cost of computing the QR factorization of B is $2m(k+p)^2 - \frac{2}{3}(k+p)^3$ (cf. [69]), and the cost of forming the first $(k+p)$ columns in the full Q matrix is $m(k+p)^2 + \frac{1}{3}(k+p)^3$.

Now we count the flops for each i in the **for** loop:

1. The cost of computing $B = A^T * Q$ is $2mn(k+p)$.
2. The cost of computing $[Q,\sim] = qr(B,0)$ is $2n(k+p)^2 - \frac{2}{3}(k+p)^3$, and the cost of forming the first $(k+p)$ columns in the full Q matrix is $n(k+p)^2 + \frac{1}{3}(k+p)^3$.
3. The cost of computing $B = A * Q$ is $2mn(k+p)$.
4. The cost of computing $[Q,\sim] = qr(B,0)$ is $2m(k+p)^2 - \frac{2}{3}(k+p)^3$, and the cost of forming the first $(k+p)$ columns in the full Q matrix is $m(k+p)^2 + \frac{1}{3}(k+p)^3$.

Together, the cost of running the **for** loop q times is

$$q\left(4mn(k+p) + 3(m+n)(k+p)^2 - \frac{2}{3}(k+p)^3\right).$$

Additionally, the cost of computing $B = Q^T * A$ is $2mn(k+p)$, the cost of doing SVD of B is $O(n(k+p)^2)$, and the cost of computing $U = Q * U$ is $2m(k+p)^2$.

ETNA
Kent State University and
Johann Radon Institute (RICAM)

Algorithm 6 ST-HOSVD.

Inputs:
Tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times \cdots \times I_d}$, desired rank $(k_1, \ldots, k_d)$, and processing order $p = (p_1, \cdots, p_d)$.
Outputs:
Tucker decomposition $[\mathcal{G}; U_1, \cdots, U_d]$.
Algorithm:
Define tensor $\mathcal{G} \leftarrow \mathcal{X}$.
for $j = 1 : d$ **do**
  $r = p_j$.
  Compute exact or approximate rank k_r SVD of the tensor unfolding $G_{(r)} \approx \widehat{U}_r \widehat{\Sigma}_r \widehat{V}_r^T$.
  $U_r \leftarrow \widehat{U}_r$.
  Update $G_{(r)} \leftarrow \widehat{\Sigma}_r \widehat{V}_r^T$, i.e., applying $\widehat{U}_r^T$ to $\mathcal{G}$.
end for

Algorithm 7 Robust PCA Using IALM.

Inputs:
Measured matrix $M \in \mathbb{R}^{m \times n}$, positive number $\lambda,\ \mu_0,\ \overline{\mu}$, tolerance tol, $\rho > 1$.
Outputs:
Matrix pair (X_k, E_k).
Algorithm:
$k = 0$; $J(M) = \max\left(\|M\|_2, \lambda^{-1}\|M\|_F\right)$; $Y_0 = M/J(M)$; $E_0 = 0$;
while not converged **do**
  $(\mathbf{U}, \mathbf{\Sigma}, \mathbf{V}) = \mathbf{svd}\left(\mathbf{M} - \mathbf{E_k} + \mu_\mathbf{k}^{-1}\mathbf{Y_k}\right)$;
  $X_{k+1} = U\mathcal{S}_{\mu_k^{-1}}(\Sigma)V^T$;
  $E_{k+1} = \mathcal{S}_{\lambda\mu_k^{-1}}\left(M - X_{k+1} + \mu_k^{-1}Y_k\right)$;
  $Y_{k+1} = Y_k + \mu_k(M - X_{k+1} - E_{k+1})$;
  Update $\mu_{k+1} = \min(\rho\mu_k, \overline{\mu})$;
  $k = k + 1$;
  if $\|M - X_k - E_k\|_F / \|M\|_F < tol$ **then**
    Break;
  end if
end while

Now assume $k + p \ll \min(m, n)$ and omit the lower-order terms, then we arrive at $(4q + 4)mn(k + p)$ as the complexity of approximate SVD with randomized subspace iteration. In practice, q is usually chosen to be an integer between 0 and 2.

REFERENCES

[1] O. ALTER, P. O. BROWN, AND D. BOTSTEIN, *Singular value decomposition for genome-wide expression data processing and modeling*, Proc. Natl. Acad. Sci. USA, 97 (2000), pp. 10101–10106.
[2] E. ANDERSON, Z. BAI, C. BISCHOF, L. S. BLACKFORD, J. DEMMEL, J. DONGARRA, J. DU CROZ, A. GREENBAUM, S. HAMMARLING, A. MCKENNEY, AND D. SORENSEN, *LAPACK Users' Guide*, 3rd ed., SIAM, Philadelphia, 1999.
[3] C. A. ANDERSSON AND R. BRO, *Improving the speed of multi-way algorithms: Part I. Tucker3*, Chemometrics Intell. Lab. Syst., 42 (1998), pp. 93–103.
[4] M. W. BERRY, S. T. DUMAIS, AND G. W. O'BRIEN, *Using linear algebra for intelligent information retrieval*, SIAM Rev., 37 (1995), pp. 573–595.
[5] C. BISCHOF AND C. VAN LOAN, *The WY representation for products of Householder matrices*, SIAM J. Sci.

ETNA
Kent State University and
Johann Radon Institute (RICAM)

Algorithm 8 Matrix Completion Using IALM.

Inputs:
Sampled set Ω, sampled entries $\pi_\Omega(M)$, positive number λ, μ_0, $\overline{\mu}$, tolerance tol, $\rho > 1$.
Outputs:
Matrix pair (X_k, E_k).
Algorithm:
$k = 0$; $Y_0 = 0$; $E_0 = 0$;
while not converged **do**
  $(\mathbf{U}, \mathbf{\Sigma}, \mathbf{V}) = \mathbf{svd}\left(\mathbf{M} - \mathbf{E_k} + \mu_\mathbf{k}^{-1}\mathbf{Y_k}\right)$;
  $X_{k+1} = U\mathcal{S}_{\mu_k^{-1}}(\Sigma)V^T$;
  $E_{k+1} = \pi_{\overline{\Omega}}\left(M - X_{k+1} + \mu_k^{-1}Y_k\right)$;
  $Y_{k+1} = Y_k + \mu_k(M - X_{k+1} - E_{k+1})$;
  Update $\mu_{k+1} = \min(\rho\mu_k, \overline{\mu})$;
  $k = k + 1$;
  if $\|M - X_k - E_k\|_F / \|M\|_F < tol$ **then**
    Break;
  end if
end while

Statist. Comput., 8 (1987), pp. S2–S13.

[6] P. BUSINGER AND G. H. GOLUB, *Handbook series linear algebra. Linear least squares solutions by Householder transformations*, Numer. Math., 7 (1965), pp. 269–276.

[7] J.-F. CAI, E. J. CANDÈS, AND Z. SHEN, *A singular value thresholding algorithm for matrix completion*, SIAM J. Optim., 20 (2010), pp. 1956–1982.

[8] E. J. CANDÈS, X. LI, Y. MA, AND J. WRIGHT, *Robust principal component analysis?*, J. ACM, 58 (2011), Art. 11, 37 pages.

[9] E. J. CANDÈS AND B. RECHT, *Exact matrix completion via convex optimization*, Found. Comput. Math., 9 (2009), pp. 717–772.

[10] J. D. CARROLL AND J.-J. CHANG, *Analysis of individual differences in multidimensional scaling via an N-way generalization of "Eckart-Young" decomposition*, Psychometrika, 35 (1970), pp. 283–319.

[11] T. A. DAVIS AND Y. HU, *The University of Florida sparse matrix collection*, ACM Trans. Math. Software, 38 (2011), pp. Art. 1, 25.

[12] L. DE LATHAUWER AND B. DE MOOR, *From matrix to tensor: Multilinear algebra and signal processing*, in Mathematics in Signal Processing IV, Inst. Math. Appl. Conf. Ser., Oxford Univ. Press, Oxford, 1998, pp. 1–16.

[13] L. DE LATHAUWER, B. DE MOOR, AND J. VANDEWALLE, *A multilinear singular value decomposition*, SIAM J. Matrix Anal. Appl., 21 (2000), pp. 1253–1278.

[14] ———, *On the best rank-1 and rank-$(R_1, R_2, \cdots, R_N)$ approximation of higher-order tensors*, SIAM J. Matrix Anal. Appl., 21 (2000), pp. 1324–1342.

[15] L. DE LATHAUWER AND J. VANDEWALLE, *Dimensionality reduction in higher-order signal processing and rank-$(R_1, R_2, \ldots, R_N)$ reduction in multilinear algebra*, Linear Algebra Appl., 391 (2004), pp. 31–55.

[16] P. DRINEAS, R. KANNAN, AND M. W. MAHONEY, *Fast Monte Carlo algorithms for matrices. II. Computing a low-rank approximation to a matrix*, SIAM J. Comput., 36 (2006), pp. 158–183.

[17] J. A. DUERSCH AND M. GU, *Randomized QR with column pivoting*, SIAM J. Sci. Comput., 39 (2017), pp. C263–C291.

[18] C. ECKART AND G. YOUNG, *The approximation of one matrix by another of lower rank*, Psychometrika, 1 (1936), pp. 211–218.

[19] L. ELDÉN, *Matrix Methods in Data Mining and Pattern Recognition*, SIAM, Philadelphia, 2007.

[20] M. FAZEL, H. HINDI, AND S. P. BOYD, *Log-det heuristic for matrix rank minimization with applications to Hankel and Euclidean distance matrices*, in Proceedings of the 2003 American Control Conference, 2003, IEEE Conference Proceedings, IEEE, Piscataway, pp. 2156–2162.

[21] M. FAZEL, T. K. PONG, D. SUN, AND P. TSENG, *Hankel matrix rank minimization with applications to system identification and realization*, SIAM J. Matrix Anal. Appl., 34 (2013), pp. 946–977.

[22] C. FRANK AND E. NÖTH, *Optimizing eigenfaces by face masks for facial expression recognition*, in Computer Analysis of Images and Patterns, N. Petkov and M. A. Westenberg, eds., vol. 2756 of Lecture Notes in Comput. Sci., Springer, Berlin, 2003, pp. 646–654.

ETNA
Kent State University and
Johann Radon Institute (RICAM)

Algorithm 9 Approximate SVD with Randomized Subspace Iteration.

Inputs:
Matrix $A \in \mathbb{R}^{m\times n}$. Target rank k. Oversampling size $p \geq 0$. Number of iterations $q \geq 1$.
Outputs:
$U \in \mathbb{R}^{m\times k}$ contains the approximate top k left singular vectors of A.
$\Sigma \in \mathbb{R}^{k\times k}$ contains the approximate top k singular values of A.
$V \in \mathbb{R}^{n\times k}$ contains the approximate top k right singular vectors of A.
Algorithm:
Generate i.i.d Gaussian matrix $\Omega \in \mathcal{N}(0,1)^{n\times(k+p)}$.
Compute $B = A\Omega$.
$[Q, \sim] = qr(B, 0)$;
for $i = 1 : q$ **do**
  $B = A^T * Q$;
  $[Q, \sim] = qr(B, 0)$;
  $B = A * Q$;
  $[Q, \sim] = qr(B, 0)$;
end for
$B = Q^T * A$;
$[U, \Sigma, V] = svd(B)$;
$U = Q * U$;
$U = U(:, 1 : k)$;
$\Sigma = \Sigma(1 : k, 1 : k)$;
$V = V(:, 1 : k)$;

[23] G. W. FURNAS, S. DEERWESTER, S. T. DUMAIS, T. K. LANDAUER, R. A. HARSHMAN, L. A. STREETER, AND K. E. LOCHBAUM, *Information retrieval using a singular value decomposition model of latent semantic structure*, in Proceedings of the 11th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Y. Chiaramella, ed., ACM, New York, 1988, pp. 465–480.
[24] K. GOLDBERG, T. ROEDER, D. GUPTA, AND C. PERKINS, *Eigentaste: A constant time collaborative filtering algorithm*, Inf. Retrieval, 4 (2001), pp. 133–151.
[25] G. GOLUB, *Numerical methods for solving linear least squares problems*, Numer. Math., 7 (1965), pp. 206–216.
[26] G. H. GOLUB AND C. F. VAN LOAN, *Matrix Computations*, 4th ed., Johns Hopkins University Press, Baltimore, 2013.
[27] M. GU, *Subspace iteration randomization and singular value problems*, SIAM J. Sci. Comput., 37 (2015), pp. A1139–A1173.
[28] M. GU AND S. C. EISENSTAT, *Efficient algorithms for computing a strong rank-revealing QR factorization*, SIAM J. Sci. Comput., 17 (1996), pp. 848–869.
[29] E. T. HALE, W. YIN, AND Y. ZHANG, *Fixed-point continuation for l_1-minimization: methodology and convergence*, SIAM J. Optim., 19 (2008), pp. 1107–1130.
[30] N. HALKO, P. G. MARTINSSON, AND J. A. TROPP, *Finding structure with randomness: probabilistic algorithms for constructing approximate matrix decompositions*, SIAM Rev., 53 (2011), pp. 217–288.
[31] R. A. HARSHMAN, *Foundations of the PARAFAC procedure: Models and conditions for an "explanatory" multimodal factor analysis*, UCLA Working Papers in Phonetics, 16 (1970), pp. 1–84.
[32] J. L. HERLOCKER, J. A. KONSTAN, A. BORCHERS, AND J. RIEDL, *An algorithmic framework for performing collaborative filtering*, in Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, F. Gey, M. Heaast, and R. Tong, eds., ACM, New York, 1999, pp. 230–237.
[33] R. A. HORN AND C. R. JOHNSON, *Topics in Matrix Analysis*, Cambridge University Press, Cambridge, 1991.
[34] D. A. HUCKABY AND T. F. CHAN, *On the convergence of Stewart's QLP algorithm for approximating the SVD*, Numer. Algorithms, 32 (2003), pp. 287–316.
[35] ———, *Stewart's pivoted QLP decomposition for low-rank matrices*, Numer. Linear Algebra Appl., 12 (2005), pp. 153–159.
[36] W. B. JOHNSON AND J. LINDENSTRAUSS, *Extensions of Lipschitz mappings into a Hilbert space*, in Conference in Modern Analysis and Probability (New Haven, Conn., 1982), R. Beals, A. Beck, A. Bellow, and A. Hajian, eds., vol. 26 of Contemp. Math., Amer. Math. Soc., Providence, 1984, pp. 189–206.

ETNA
Kent State University and
Johann Radon Institute (RICAM)

[37] I. T. JOLLIFFE, *Principal Component Analysis*, Springer, New York, 1986.
[38] J. M. KLEINBERG, *Authoritative sources in a hyperlinked environment*, J. ACM, 46 (1999), pp. 604–632.
[39] T. G. KOLDA, *Orthogonal tensor decompositions*, SIAM J. Matrix Anal. Appl., 23 (2001), pp. 243–255.
[40] T. G. KOLDA AND B. W. BADER, *Tensor decompositions and applications*, SIAM Rev., 51 (2009), pp. 455–500.
[41] R. M. LARSEN, *PROPACK-software for large and sparse SVD calculations*, available at http://sun.stanford.edu/rmunk/PROPACK.
[42] ———, *Lanczos bidiagonalization with partial reorthogonalization*, Tech. Report, PB-537, Department of Computer Science, Aarhus University, Aarhus, Denmark, 1998.
[43] Y. LECUN, L. BOTTOU, Y. BENGIO, AND P. HAFFNER, *Gradient-based learning applied to document recognition*, Proc. IEEE, 86 (1998), pp. 2278–2324.
[44] R. B. LEHOUCQ, D. C. SORENSEN, AND C. YANG, *ARPACK Users' Guide: Solution of Large-Scale Eigenvalue Problems with Implicitly Restarted Arnoldi Methods*, SIAM, Philadelphia, 1998.
[45] Z. LIN, M. CHEN, AND Y. MA, *The augmented Lagrange multiplier method for exact recovery of corrupted low-rank matrices*, Preprint on arXiv, https://arxiv.org/abs/1009.5055, 2010.
[46] N. LINIAL, E. LONDON, AND Y. RABINOVICH, *The geometry of graphs and some of its algorithmic applications*, Combinatorica, 15 (1995), pp. 215–245.
[47] Z. LIU, A. HANSSON, AND L. VANDENBERGHE, *Nuclear norm system identification with missing inputs and outputs*, Systems Control Lett., 62 (2013), pp. 605–612.
[48] Z. LIU AND L. VANDENBERGHE, *Interior-point method for nuclear norm approximation with application to system identification*, SIAM J. Matrix Anal. Appl., 31 (2009), pp. 1235–1256.
[49] S. MA, D. GOLDFARB, AND L. CHEN, *Fixed point and Bregman iterative methods for matrix rank minimization*, Math. Program., 128 (2011), pp. 321–353.
[50] C. B. MELGAARD, *Randomized Pivoting and Spectrum-Revealing Bounds in Numerical Linear Algebra*, Ph.D. Thesis, University of California, Berkeley, 2015.
[51] M. MESBAHI AND G. P. PAPAVASSILOPOULOS, *On the rank minimization problem over a positive semidefinite linear matrix inequality*, IEEE Trans. Automat. Control, 42 (1997), pp. 239–243.
[52] N. MULLER, L. MAGAIA, AND B. M. HERBST, *Singular value decomposition, eigenfaces, and 3D reconstructions*, SIAM Rev., 46 (2004), pp. 518–545.
[53] M. NARWARIA AND W. LIN, *SVD-based quality metric for image and video using machine learning*, IEEE Trans. Syst,, Man, and Cybernetics, Part B (Cybernetics), 42 (2012), pp. 347–364.
[54] T.-H. OH, Y. MATSUSHITA, Y.-W. TAI, AND I. SO KWEON, *Fast randomized singular value thresholding for nuclear norm minimization*, in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition 2015, IEEE Conference Proceedings, IEEE, Los Alamitos, 2015, pp. 4484–4493.
[55] G. QUINTANA-ORTÍ, X. SUN, AND C. H. BISCHOF, *A BLAS-3 version of the QR factorization with column pivoting*, SIAM J. Sci. Comput., 19 (1998), pp. 1486–1494.
[56] B. RECHT, M. FAZEL, AND P. A. PARRILO, *Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization*, SIAM Rev., 52 (2010), pp. 471–501.
[57] J. D. RENNIE AND N. SREBRO, *Fast maximum margin matrix factorization for collaborative prediction*, in Proceedings of the 22nd International Conference on Machine Learning, L. De Raedt and S. Wrobel, eds., ACM Press, Neow York, 2005, pp. 713–719.
[58] V. ROKHLIN, A. SZLAM, AND M. TYGERT, *A randomized algorithm for principal component analysis*, SIAM J. Matrix Anal. Appl., 31 (2009), pp. 1100–1124.
[59] A. K. SAIBABA, *HOID: higher order interpolatory decomposition for tensors based on Tucker representation*, SIAM J. Matrix Anal. Appl., 37 (2016), pp. 1223–1249.
[60] B. SAVAS AND L. ELDÉN, *Handwritten digit classification using higher order singular value decomposition*, Pattern Recognition, 40 (2007), pp. 993–1003.
[61] R. SCHREIBER AND C. VAN LOAN, *A storage-efficient WY representation for products of Householder transformations*, SIAM J. Sci. Statist. Comput., 10 (1989), pp. 53–57.
[62] A. SHASHUA AND T. HAZAN, *Non-negative tensor factorization with applications to statistics and computer vision*, in Proceedings of the 22nd International Conference on Machine Learning, S. Dzeroski, L. De Raedt, and S. Wrobel, eds., ACM, New York, 2005, pp. 792–799.
[63] N. D. SIDIROPOULOS, R. BRO, AND G. B. GIANNAKIS, *Parallel factor analysis in sensor array processing*, IEEE Trans. Signal Process, 48 (2000), pp. 2377–2388.
[64] D. C. SORENSEN AND M. EMBREE, *A DEIM induced CUR factorization*, SIAM J. Sci. Comput., 38 (2016), pp. A1454–A1482.
[65] N. SREBRO, J. RENNIE, AND T. S. JAAKKOLA, *Maximum-margin matrix factorization*, in NIPS'04 Proceedings of the 17th International Conference on Neural Information Processing Systems, L. K. Saul, Y. Weiss, and L. Bottou, eds., MIT Press, Cambridge, 2004, pp. 1329–1336.
[66] G. W. STEWART, *The QLP approximation to the singular value decomposition*, SIAM J. Sci. Comput., 20 (1999), pp. 1336–1348.

ETNA
Kent State University and
Johann Radon Institute (RICAM)

[67] S. TAHERI, Q. QIU, AND R. CHELLAPPA, *Structure-preserving sparse decomposition for facial expression analysis*, IEEE Trans. Image Process., 23 (2014), pp. 3590–3603.
[68] K.-C. TOH AND S. YUN, *An accelerated proximal gradient algorithm for nuclear norm regularized linear least squares problems*, Pac. J. Optim., 6 (2010), pp. 615–640.
[69] L. N. TREFETHEN AND D. BAU III, *Numerical Linear Algebra*, SIAM, Philadelphia, 1997.
[70] L. R. TUCKER, *Some mathematical notes on three-mode factor analysis*, Psychometrika, 31 (1966), pp. 279–311.
[71] N. VANNIEUWENHOVEN, R. VANDEBRIL, AND K. MEERBERGEN, *A new truncation strategy for the higher-order singular value decomposition*, SIAM J. Sci. Comput., 34 (2012), pp. A1027–A1052.
[72] M. A. O. VASILESCU AND D. TERZOPOULOS, *Multilinear analysis of image ensembles: Tensorfaces*, in European Conference on Computer Vision 2002, A. Heyden, G. Sparr, M. Nielsen, and P. Johansen, eds, Lecture Notes in Computer Science vol. 2350, Springer, Berlin, 2002, pp. 447–460.
[73] N. VERVLIET, O. DEBALS, L. SORBER, M. V. BAREL, AND L. DE LATHAUWER, *Tensorlab user guide*, available at http://www.tensorlab.net.
[74] J. XIAO AND M. GU, *Spectrum-revealing Cholesky factorization for kernel methods*, in Proceedings of the 16th IEEE International Conference on Data Mining (ICDM), IEEE Conference Proceedings, Los Alamitos, 2016, pp. 1293–1298.
[75] J. XIAO, M. GU, AND J. LANGOU, *Fast parallel randomized QR with column pivoting algorithms for reliable low-rank matrix approximations*, in Proceedings of the 24th IEEE International Conference on High Performance Computing (HiPC), IEEE Conference Proceedings, Los Alamitos, 2017, pp. 233–242.
[76] T. ZHANG AND G. H. GOLUB, *Rank-one approximation to high order tensors*, SIAM J. Matrix Anal. Appl., 23 (2001), pp. 534–550.