Pontifícia Universidade Católica
do Rio de Janeiro

Maboud Farzaneh Kaloorazi

Low-Rank Matrix Approximations and Applications

Tese de Doutorado

Tese apresentada ao Programa de Pós–graduação em Engenharia Elétrica da PUC-Rio como requisito parcial para obtenção do grau de Doutor em Engenharia Elétrica.

Advisor: Prof. Rodrigo Caiado de Lamare

Rio de Janeiro
July 2018

Pontifícia Universidade Católica
do Rio de Janeiro

Maboud Farzaneh Kaloorazi

Low-Rank Matrix Approximations and Applications

Thesis presented to the Programa de Pós–graduação em Engenharia Elétrica of PUC-Rio, in partial fulfillment of the requirements for the degree of Doutor em Engenharia Elétrica. Approved by the undersigned Examination Committee.

Prof. Rodrigo Caiado de Lamare
Adviser
Centro de Estudos e Telecomunicações - PUC-Rio

Prof. Raul Queiroz Feitosa
Departamento de Engenharia Elétrica - PUC-Rio

Prof. Cristiano Augusto Coelho Fernandes
Departamento de Engenharia Elétrica - PUC-Rio

Prof. Lukas Tobias Nepomuk Landau
Centro de Estudos e Telecomunicações - PUC-Rio

Prof. Vitor Heloiz Nascimento
USP

Prof. Jose Carlos Moreira Bermudez
UFSC

Prof. João Terêncio Dias
CEFET/RJ

Prof. Márcio da Silveira Carvalho
Vice Dean of Graduate Studies, Centro Técnico Científico - PUC-Rio

Rio de Janeiro, July 19th, 2018

Maboud Farzaneh Kaloorazi

Graduou-se em engenharia elétrica pela Universidade de Guilan, Rasht, Irã, ele recebeu seu diploma de mestrado do Instituto Blekinge de Tecnologia de Karlskrona, na Suécia. Ele fez o doutorado no departamento de engenharia elétrica (CETUC), trabalhando em algoritmos com randomização e suas aplicações para a ciência de dados.

Ficha Catalográfica

Maboud Farzaneh Kaloorazi

Low-Rank Matrix Approximations and Applications / Maboud Farzaneh Kaloorazi; advisor: Rodrigo Caiado de Lamare. – 2018.

v., 120 f: il. color. ; 30 cm

Tese (doutorado) - Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Engenharia de Elétrica.

Inclui bibliografia

1. Engenharia de Elétrica – Teses. 2. Engenharia de Elétrica – Teses. 3. Cálculos matriciais,. 4. Aproximação de posto reduzido,. 5. Algoritmos com randomização,. 6. Redução de dimensão,. 7. Análise de componentes principais robusta.. I. Rodrigo C. de Lamare. II. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Engenharia de Elétrica. III. Título.

CDD: 621.3

Acknowledgments

I would first and foremost like to thank my adviser, Rodrigo de Lamare, for his encouragement, patience, and guidance during my time as his student. This work would not have been possible without him.

I would like to thank Coordenação de Aperfeicoamento de Pessoal de Nível Superior (CAPES) for funding this wok.

I would also like to thank the members of my dissertation committee, Jose Calrols Bermudez, Vitor Nascimento, Raul feitosa, Cristiano Fernandes, João Dias, and Lukas Landau for their time and effort dedicated to my thesis.

I wish to thank my friends and officemates over the years for their friendship and interesting discussions. Special thanks to Amir for his suggestion.

I would also like to thank my parents for all of their love, encouragements, and on-going support throughout the years. I am greatly indebted to my brother, Mesiam, for his encouragement and support during my graduate studies.

Finally, I would like to thank Feifei for her love and friendship, for always being on my side helping me navigate my way.

Abstract

Maboud Farzaneh Kaloorazi; Rodrigo C. de Lamare. **Low-Rank Matrix Approximations and Applications**. Rio de Janeiro, 2018. 120p. Tese de Doutorado – Departamento de Engenharia de Elétrica, Pontifícia Universidade Católica do Rio de Janeiro.

This dissertation focuses on developing algorithms based on randomized sampling techniques for low-rank matrix approximations.

Low-rank matrix approximations, that is, approximating a given matrix by one of lower rank, play an increasingly important role in numerical linear algebra and signal processing applications. Such a compact representation which retains most important information of a high-dimensional matrix can provide a significant reduction in memory requirements and, more importantly, computational costs when the computational cost scales, e.g., according to a high-degree polynomial, with the dimensionality.

The low-rank approximation algorithms proposed, first, alternately project the matrix onto its row and column space via randomized sampling. Second, approximate bases for the row and column space are computed. Third, the matrix is transformed into a lower dimensional space using the bases obtained. Next, a deterministic method factors the transformed (reduced-size) data, and the final low-rank approximation is computed by projecting it back to the original space. Theoretical lower bounds on the singular values and upper bounds on the error of the low-rank approximations for the algorithm are provided. Due to recently developed Communication-Avoiding QR algorithms, which can perform the computation with optimal/minimum communication costs, the proposed algorithms can exploit modern architectures and, consequently, can be optimized for maximum efficiency.

To demonstrate the efficiency and efficacy of the algorithms, we consider image reconstruction and robust principal component analysis (decomposing a matrix with grossly corrupted entries into a low-rank matrix plus a sparse matrix of outliers) applications. Through numerical experiments with synthetic and real data, we verify that the algorithms are efficient, stable and highly accurate.

Keywords

Matrix computations, low-rank approximation, dimension reduction, matrix decomposition, numerical linear algebra, randomized algorithms, randomized subspace methods, UTV decompositions, PCA, robust PCA, image reconstruction, convex optimization, background subtraction, anomaly detection.

Resumo

Maboud Farzaneh Kaloorazi; Rodrigo C. de Lamare (Advisor). . Rio de Janeiro, 2018. 120p. PhD Thesis – Departamento de Engenharia de Elétrica, Pontifícia Universidade Católica do Rio de Janeiro.

Esta dissertação trata do desenvolvimento de algoritmos baseados em técnicas de randomização para aproximações matriciais de posto reduzido, que servem para representar uma dada matriz de dados por uma matriz de posto reduzido. Essas técnicas desempenham um papel cada vez mais importante em álgebra linear numérica e em aplicações de processamento de sinais. Isto se deve ao fato de que representações compactas que retém os atributos mais importantes de uma matriz de grandes dimensões podem proporcionar uma redução significativa nos requisitos de memória e no custos computacionais, que variam de acordo com um polinômio dependente de uma das dimensões da matriz a ser aproximada.

Os algoritmos de aproximação de posto reduzido se propõe em projetar alternadamente a matriz em seu espaço de linha e coluna por meio de amostragem aleatória. Em segundo lugar, bases aproximadas para o espaço de linha e coluna são calculadas. Em terceiro lugar, a matriz é transformada em um espaço dimensional inferior usando as bases obtidas. Em seguida, um método determinístico fatora os dados transformados em tamanho reduzido e a aproximação de posto reduzido é calculada projetando-a de volta ao espaço original. Os limites inferiores teóricos dos valores singulares e os limites superiores do erro das aproximações de posto reduzido dos algoritmos são estabelecidos. Levando-se em consideração o custo de comunicação das matrizes de dados em processadores e algoritmos QR recentemente desenvolvidos, que podem realizar a computação com custos de comunicação reduzidos, os algoritmos propostos podem explorar arquiteturas modernas e ser otimizados para máxima eficiência. Para ilustrar a eficiência e a eficácia dos algoritmos, consideramos a reconstrução de imagens, a análise de componentes principais robusta, em que uma matriz com aplicações grosseiramente corrompidas em uma matriz de baixa classificação mais uma matriz esparsa de outliers é decomposta, e problemas de detecção de anomalias. Através de experimentos numéricos com dados sintéticos e reais, verificamos que os algoritmos são eficientes, estáveis, e altamente precisos.

Palavras-chave

Cálculos matriciais, Aproximação de posto reduzido, Algoritmos com randomização, Redução de dimensão, Análise de componentes principais robusta.

Summary

Figure list

Table list

List of Abreviations

PCA – Principal Component Analysis
RPCA – Robust Principal Component Analysis
SVD – Singular Value Decomposition
RRR-UZVD – Randomized Rank-Revealing UZV Decomposition
SOR-SVD – Subspace-Orbit Randomized Singular Value Decomposition
CoR-UTV – Compressed Randomized UTV Decompositions
SR-RPCA – Switched-Randomized Robust Principal Component Analysis
RSMs – Randomized Subspace Methods
RBAD – Randomized Basis for Anomaly Detection
SSPBAD – Switched Subspace-Projected Basis for Anomaly Detection
ALM – Augmented Lagrange Multipliers
Flops – Floating-Point Operations

1
Introduction

With recent advances in collecting data and storage capabilities, large volumes of data are produced every day in areas such as engineering, economics, astronomy, biology, remote sensing [98]. Such high-dimensional data, which now are termed big data, present new challenges in data analysis, since the traditional approaches break down partly due to the increase in the number of observations [24]. In dealing with high-dimensional data sets, in many cases, not all the variables (measured with each observation) are important, due to large number of interrelated variables, for understanding the underlying phenomena of interest. Thus, it is of high interest to reduce the dimensionality of the original data set or to approximate it with a lower-dimensional component prior to any modeling or procedure to extract useful information. Representing a high-dimensional data set by a lower dimensional component, which contains as much variation of the data as possible, can significantly reduce memory requirements, and more importantly, computational costs of the data processing [77].

1.1
Motivations, Problems and Background

Computing a low-rank approximation of an input data matrix, i.e., approximating the matrix by one of lower rank, is a fundamental task in numerical linear algebra and signal processing applications. Such a compact representation, which retains most important information of a high-dimensional matrix can provide a significant reduction in memory requirements, and more importantly, computational costs when the computational cost scales, e.g., according to a high-degree polynomial, with the dimensionality. Matrices with low-rank structures have found many applications in background subtraction [5, 53, 69, 93], system identification [30], IP network anomaly detection [52,60], latent variable graphical modeling, [13], ranking and collaborative filtering, [78], subspace clustering [67, 68, 76], sensor and multichannel signal processing [17], biometrics [91,94], statistical process control and multidimensional fault identification [27, 46], quantum state tomography [16], and DNA microarray data [88].

Traditional algorithms such as the singular value decomposition (SVD) [35] and the rank-revealing QR (RRQR) decomposition [11,37] are among the most commonly used algorithms for computing a low-rank approximation of a matrix. A UTV decomposition, proposed in [79, 81], on the other hand, is a compromise between the SVD and the RRQR decomposition, having the virtues of both. Given a matrix $\mathbf{A}$, the UTV algorithm computes a decomposition $\mathbf{A} = \mathbf{UTV}^T$, where $\mathbf{U}$ and $\mathbf{V}$ have orthonormal columns, and $\mathbf{T}$ is triangular (either upper or lower triangular). These deterministic algorithms, however, are computationally expensive for large data sets. Furthermore, standard techniques for their computation are challenging to parallelize in order to utilize advanced computer architectures [18, 36, 39].

Recently developed algorithms for low-rank approximations based on random sampling schemes have been shown to be surprisingly computationally efficient, highly accurate and robust, and are known to outperform the traditional algorithms in many practical situations [25, 33, 36, 39, 71, 74]. These randomized algorithms first form a compressed version of the given matrix through random linear combinations of its rows or columns. Further computations are then performed on the submatrix using deterministic algorithms such as the SVD and the QR decomposition with column pivoting to obtain the final low-rank approximation. The advantage of randomized algorithms over their classical counterparts lies in the fact that (i) they operate on a compressed version of the data matrix rather than a matrix itself, so they are computationally efficient, and (ii) they can be organized to take advantage of modern architectures, performing a decomposition with minimum communication cost [15, 21].

Motivated by recent developments, the scope of this dissertation is to contribute to the aforementioned line of work by developing efficient, accurate and provably correct randomized algorithms for low-rank matrix approximation.

1.2 Structure and Contributions of the Dissertation

In this section, we outline the structure of this work and highlight our contributions. In addition to this introductory chapter, this dissertation consists of six more chapters. We will summarize the content and key contributions of each chapter.

1.2.1 Chapter 2: Literature Review

This chapter surveys prior and related works which based on this dissertation grew out. It discusses deterministic and randomized methods for dimensionality reduction and low-rank matrix approximation, including linear principal component analysis (PCA) technique and non-linear robust PCA. At the end of this chapter, we further present a new fast robust PCA technique applied to background subtraction in surveillance videos.

1.2.2 Chapter 3: Randomized Rank-Revealing UZV Decomposition

This chapter presents an efficient rank-revealing algorithm powered by randomization termed randomized rank-revealing UZV decomposition (RRR-UZVD). For an input matrix, RRR-UZVD delivers information on a specific number of leading singular values and corresponding singular vectors of the matrix. The work of this chapter serves as the basis for the algorithms developed in the next two chapters. Main contributions include:

- Given a matrix $\mathbf{A}$, RRR-UZVD constructs an approximation such as $\mathbf{A} \approx \mathbf{U}\mathbf{Z}\mathbf{V}^T$, where $\mathbf{U}$ and $\mathbf{V}$ have orthonormal columns, leading-diagonal block of $\mathbf{Z}$ is well-conditioned and reveals the numerical rank of $\mathbf{A}$, and its off-diagonal blocks have sufficiently small ℓ_2-norms.
- The rank-revealing property of the proposed algorithm is proved.
- RRR-UZVD is applied to reconstruct a low-rank image as well as to solve the robust PCA problem [9, 14, 93], i.e, to decompose a matrix into its low-rank and sparse components, in applications of background modeling in surveillance video and shadow and specularity removal from face images.

1.2.3 Chapter 4: Subspace-Orbit Randomized SVD

This chapter proposes a new matrix decomposition approach termed Subspace-Orbit Randomized Singular Value Decomposition (SOR-SVD) which based on random sampling techniques approximates the SVD of a given matrix. SOR-SVD is simple, accurate, numerically stable, and provably correct. Main contributions include:

- Given a large and dense matrix of size $m \times n$, SOR-SVD computes a rank-k approximation of the matrix by making a few passes over the data

with an arithmetic cost of $O(mnk)$ floating-point operations. The main operations of the algorithm involve matrix-matrix multiplication and the QR decomposition, and due to recently developed Communication-Avoiding QR (CAQR) algorithms that perform the computation with optimal communication cost [21], SOR-SVD can be optimized for peak machine performance on modern computational platforms.

– Theoretical lower bounds on the singular values and upper bounds on the error of the low-rank approximation for SOR-SVD are provided. It is experimentally shown that the low-rank approximation error bounds provided are empirically sharp for one class of matrices considered.

– SOR-SVD is employed to solve the robust PCA problem [9, 14, 93] and studied in computer vision applications of background/foreground separation in surveillance video and shadow and specularity removal from face images.

1.2.4 Chapter 5: Compressed Randomized UTV Decompositions

This chapter introduces a novel rank-revealing matrix decomposition algorithm termed Compressed Randomized UTV (CoR-UTV) decomposition. CoR-UTV is primarily developed to compute a low-rank approximation of an input matrix by using random sampling schemes. Main contributions include:

– Given a large and dense matrix $\mathbf{A}$ of size $m \times n$ with numerical rank k, CoR-UTV computes a low-rank approximation $\hat{\mathbf{A}}_{\text{CoR}}$ of $\mathbf{A}$ such that

$$\hat{\mathbf{A}}_{\text{CoR}} = \mathbf{UTV}^T, \tag{1-1}$$

where $\mathbf{U}$ and $\mathbf{V}$ have orthonormal columns, and $\mathbf{T}$ is triangular (either upper or lower, whichever is preferred). CoR-UTV only requires a few passes through data, for a matrix stored externally, and runs in $O(mnk)$ floating-point operations. The operations of the algorithm involve matrix-matrix multiplication, the QR and rank revealing QR decompositions. Due to recently developed Communication-Avoiding QR algorithms [20, 21, 26], which perform the computations with optimal/minimum communication costs, CoR-UTV can be optimized for peak machine performance on modern architectures.

– The rank-revealing property of CoR-UTV is proved, and upper bounds on the error of the low-rank approximation are given.

– CoR-UTV is applied to treat an image reconstruction problem, as well as the robust PCA problem [9, 14, 93] in applications of background

subtraction in surveillance video and shadow and specularity removal from face images.

1.2.5
Chapter 6: Randomized Subspace Methods

This chapter presents two novel subspace separation methods using randomization, collectively called randomized subspace methods (RSMs), to detect anomalies in Internet Protocol (IP) networks. Main contributions include:

– Given a matrix of link traffic data, RSMs perform a normal-plus-anomalous matrix decomposition and, subsequently, detect traffic anomalies in the anomalous subspace using a statistical test. In contrast to the traditional subspace methods, RSMs do not form the covariance matrix of the traffic data and, as a result, obviate computing the expensive SVD for separating the subspaces.

1.2.6
Chapter 7: Conclusions

This chapter summarizes our work and discusses directions for future research and some open problems that can be further explored.

1.3
Notation

We now introduce the mathematical notation that will be used throughout this thesis.

Bold-face upper-case letters are used to denote matrices. Given a matrix $\mathbf{A}$, $\|\mathbf{A}\|_1$, $\|\mathbf{A}\|_2$, $\|\mathbf{A}\|_F$, $\|\mathbf{A}\|_*$ denote the ℓ_1-norm, the spectral norm, the Frobenius norm, the nuclear norm, respectively. $\sigma_j(\mathbf{A})$ denotes the j-th largest singular value of $\mathbf{A}$, and the numerical range of $\mathbf{A}$ is denoted by $\mathcal{R}(\mathbf{A})$. The symbol $\mathbb{E}$ denotes expected value with respect to random variables. Given a random variable Ω, $\mathbb{E}_\Omega$ denotes expectation with respect to the randomness in Ω, and the dagger $\dagger$ denotes the Moore-Penrose pseudo-inverse.

1.4
List of Publications

The present work has resulted in the following publications.

– M. F. Kaloorazi and R. C. de Lamare, "Subspace-Orbit Randomized Decomposition for Low-Rank Matrix Approximations," *IEEE Transactions on Signal Processing, 66 (2018), pp. 4409-4424.*

– M. F. Kaloorazi and R. C. de Lamare, "Compressed Randomized UTV Decompositions for Low-Rank Matrix Approximations," 2018, *Submitted to IEEE Journal of Selected Topics in Signal Processing.*

– M. F. Kaloorazi and R. C. de Lamare, "Low-Rank Matrix Approximations Using Rank-Revealing UZV Decomposition," 2018, *Submitted.*

– M. F. Kaloorazi and R. C. de Lamare. "Subspace-Orbit Randomized-Based Decomposition for Low-Rank Matrix Approximations," *Accepted for publication at 26th European Signal Processing Conference (EUSIPCO 2018).*

– M. F. Kaloorazi and R. C. de Lamare, "Low-Rank and Sparse Matrix Recovery Based on a Randomized Rank-Revealing Decomposition," in 22nd International Conference on Digital Signal Processing 2017, UK.

– M. F. Kaloorazi and R. C. de Lamare, "Anomaly Detection in IP Networks Based on Randomized Subspace Methods," in 42nd IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Mar 2017, USA.

– M. F. Kaloorazi and R. C. de Lamare, "Switched-Randomized Robust PCA for Background and Foreground Separation in Video Surveillance," in 9th IEEE Sensor Array and Multichannel Signal Processing Workshop (SAM), Jul 2016, Brazil.

2
Literature Review

In this chapter, we carry out a literature review on the most relevant low-rank matrix approximation techniques that are used in the comparisons with the approaches proposed in this thesis. Low rank matrix approximations consist of computing an approximation of a matrix by one of lower rank, which can be used in a variety of signal processing applications such as image reconstruction, background/foreground subtraction, and anomaly detection. The goal is to compactly represent the input matrix with limited loss of information. Such a representation can provide a significant reduction in memory requirements as well as computational costs [77]. In what follows, we will review several algorithms for low-rank matrix approximations that include deterministic algorithms, randomized algorithms, principal component analysis and robust principal component analysis.

2.1
Deterministic Algorithms

2.1.1
Singular Value Decomposition

Given a matrix $\mathbf{A} \in \mathbb{R}^{m\times n}$, where $m \geq n$, with numerical rank k, its singular value decomposition (SVD) [18, 35] is defined as:

$$\begin{aligned}\mathbf{A} =& \mathbf{U}_{\mathrm{A}}\mathbf{\Sigma}_{\mathrm{A}}\mathbf{V}_{\mathrm{A}}^{T} \\ =& \underbrace{\begin{bmatrix}\mathbf{U}_k & \mathbf{U}_0\end{bmatrix}}_{\mathbf{U}_{\mathrm{A}}\in\mathbb{R}^{m\times n}} \underbrace{\begin{bmatrix}\mathbf{\Sigma}_k & 0 \\ 0 & \mathbf{\Sigma}_0\end{bmatrix}}_{\mathbf{\Sigma}_{\mathrm{A}}\in\mathbb{R}^{n\times n}} \underbrace{\begin{bmatrix}\mathbf{V}_k & \mathbf{V}_0\end{bmatrix}^{T}}_{\mathbf{V}_{\mathrm{A}}^{T}\in\mathbb{R}^{n\times n}},\end{aligned} \tag{2-1}$$

where $\mathbf{U}_k \in \mathbb{R}^{m\times k}$, $\mathbf{U}_0 \in \mathbb{R}^{m\times n-k}$ have orthonormal columns, spanning the range of $\mathbf{A}$ and the null space of $\mathbf{A}^T$, respectively, $\mathbf{\Sigma}_k \in \mathbb{R}^{k\times k}$ and $\mathbf{\Sigma}_0 \in \mathbb{R}^{n-k\times n-k}$ are diagonal containing the singular values, i.e., $\mathbf{\Sigma}_k = \mathrm{diag}(\sigma_1, ..., \sigma_k)$ and $\mathbf{\Sigma}_0 = \mathrm{diag}(\sigma_{k+1}, ..., \sigma_n)$, and $\mathbf{V}_k \in \mathbb{R}^{n\times k}$ and $\mathbf{V}_0 \in \mathbb{R}^{n\times n-k}$ have orthonormal columns, spanning the range of $\mathbf{A}^T$ and the null space of $\mathbf{A}$, respectively. $\mathbf{A}$ can be written as $\mathbf{A} = \mathbf{A}_k + \mathbf{A}_0$, where $\mathbf{A}_k = \mathbf{U}_k\mathbf{\Sigma}_k\mathbf{V}_k^T$, and $\mathbf{A}_0 = \mathbf{U}_0\mathbf{\Sigma}_0\mathbf{V}_0^T$. The SVD constructs the optimal rank-k approximation $\mathbf{A}_k$ to

A, as stated in the following theorem.

Theorem 2.1 *(Eckart and Young [28], and Mirsky [64])*

$$\underset{rank(\mathbf{B})\leq k}{minimize} \quad \|\mathbf{A}-\mathbf{B}\|_2 = \|\mathbf{A}-\mathbf{A}_k\|_2 = \sigma_{k+1}. \tag{2-2}$$

$$\underset{rank(\mathbf{B})\leq k}{minimize} \quad \|\mathbf{A}-\mathbf{B}\|_F = \|\mathbf{A}-\mathbf{A}_k\|_F = \sqrt{\sum_{j=k+1}^{n}\sigma_j^2}. \tag{2-3}$$

The SVD is numerically stable and highly accurate and yields detailed information on singular subspaces and singular values, however it is prohibitive to compute i.e., it costs $O(mn^2)$ flops. Moreover, standard techniques for its computation are challenging to parallelize in order to take advantage of modern computational environments [18, 36, 39]. To approximate the SVD, however, a Krylov subspace method such as the Lanczos and Arnoldi algorithm can be used, which constructs a partial SVD of a matrix, for instance **A**, at a cost $O(mnk)$. However, these methods suffer from two drawbacks. First, inherently, they are numerically unstable [8, 18, 35]. Second, they do not lend themselves to parallel implementations [36, 39], which makes them unsuitable for modern computational architectures.

2.1.2 Rank-Revealing QR Decomposition

Another widely used algorithm for low-rank approximations considered as a relatively economic alternative to the SVD is the rank-revealing QR decomposition (RRQR) [11]. The RRQR is a special QR decomposition with column pivoting (QRCP), which reveals the numerical rank of the input matrix. Given the matrix **A**, it takes the following form:

$$\mathbf{AP} = \mathbf{QR} = \mathbf{Q}\begin{bmatrix}\mathbf{R}_{11} & \mathbf{R}_{12}\\ 0 & \mathbf{R}_{22}\end{bmatrix}, \tag{2-4}$$

where **P** is a permutation matrix, $\mathbf{Q} \in \mathbb{R}^{m\times n}$ has orthonormal columns, $\mathbf{R} \in \mathbb{R}^{n\times n}$ is upper triangular where $\mathbf{R}_{11} \in \mathbb{R}^{k\times k}$ is well-conditioned with $\sigma_{\min}(\mathbf{R}_{11}) = O(\sigma_k)$, and the ℓ_2-norm of $\mathbf{R}_{22} \in \mathbb{R}^{n-k\times n-k}$ is sufficiently small, i.e., $\|\mathbf{R}_{22}\|_2 = O(\sigma_{k+1})$ (here we have written the reduced QR decomposition, where the silent columns and rows of **Q** and **R**, respectively, have been removed). If there is an additional requirement that the ℓ_2-norm of $\mathbf{R}_{11}^{-1}\mathbf{R}_{22}$ is small, i.e., a low order polynomial in n, this decomposition is called "strong RRQR decomposition" [37]. The rank-k approximation to **A** is then computed as follows:

$$\hat{\mathbf{A}}_{\text{RRQR}} = \mathbf{Q}(:, 1:k)\mathbf{R}(1:k, :)\mathbf{P}^T, \tag{2-5}$$

where we have used MATLAB notation to indicate submatrices, i.e., $\mathbf{Q}(:, 1:k)$ denotes the first k columns of $\mathbf{Q}$, and $\mathbf{R}(1:k, :)$ denotes the first k rows of $\mathbf{R}$.

2.1.3 UTV Decompositions

A UTV decomposition [79, 81] is a compromise between the SVD and QRCP, which has the virtues of both: UTV (i) is computationally more efficient than the SVD, and (ii) provides information on the numerical null space of the matrix (RRQR does not explicitly furnish the null space information) [40, 79, 81, 82]. For the matrix $\mathbf{A}$, UTV takes the form:

$$\mathbf{A} = \mathbf{U}\mathbf{T}\mathbf{V}^T \tag{2-6}$$

where $\mathbf{U} \in \mathbb{R}^{m\times n}$ and $\mathbf{V} \in \mathbb{R}^{n\times n}$ have orthonormal columns, and $\mathbf{T}$ is triangular. If $\mathbf{T}$ is upper triangular, the decomposition is called URV decomposition:

$$\mathbf{A} = \mathbf{U}\begin{bmatrix}\mathbf{T}_{11} & \mathbf{T}_{12}\\ 0 & \mathbf{T}_{22}\end{bmatrix}\mathbf{V}^T. \tag{2-7}$$

If $\mathbf{T}$ is lower triangular, the decomposition is called ULV decomposition:

$$\mathbf{A} = \mathbf{U}\begin{bmatrix}\mathbf{T}_{11} & 0\\ \mathbf{T}_{21} & \mathbf{T}_{22}\end{bmatrix}\mathbf{V}^T. \tag{2-8}$$

The URV and ULV decompositions are collectively referred to as UTV decompositions [35, 40], and are performed by reduction of the matrix $\mathbf{A}$ using unitary transformations to upper and lower triangular forms, respectively. If there is a well-defined gap in the singular value spectrum of $\mathbf{A}$, i.e., $\sigma_k \gg \sigma_{k+1}$, the UTV decompositions are said to be rank-revealing in the sense that the numerical rank k is revealed in the triangular submatrix $\mathbf{T}_{11} \in \mathbb{R}^{k\times k}$ (2-7), (2-8), and the ℓ_2-norm of off-diagonal submatrices, $[\mathbf{T}_{12}^T \quad \mathbf{T}_{22}^T]^T$ and $[\mathbf{T}_{21} \quad \mathbf{T}_{22}]$, are of the order σ_{k+1} [32, 79, 81], i.e.,

$$\begin{aligned}&\sigma_{\min}(\mathbf{T}_{11}) = O(\sigma_k),\\ &\|[\mathbf{T}_{12}^T \quad \mathbf{T}_{22}^T]^T\|_2 = O(\sigma_{k+1}),\\ &\|\mathbf{T}_{21} \quad \mathbf{T}_{22}\|_2 = O(\sigma_{k+1}).\end{aligned} \tag{2-9}$$

QRCP and UTV decompositions provide highly accurate approximations to $\mathbf{A}$, however they suffer from two drawbacks. First, they are expensive to compute in terms of arithmetic costs, i.e., $O(mn^2)$ flops. Second, methods for their computation are challenging to parallelize, and as a result, they can not exploit modern architectures [18, 36, 39].

2.2 Randomized Algorithms

Recently developed algorithms for low-rank approximations based on randomization [25,33,36,39,71,74,86] have attracted significant attention due to the facts that (i) they are computationally efficient, and (ii) they readily lend themselves to a parallel implementation to exploit advanced computational platforms.

2.2.1 Random Projections

In random projections (RP) [3, 49, 63], the given high-dimensional data matrix is projected onto a lower-dimensional subspace using a random matrix. The RP is a computationally efficient dimensionality reduction technique, but, unlike PCA, it is not optimal in terms of mean-square error since it introduces a trivial distortion in the data. Given $\mathbf{A} \in \mathbb{R}^{m\times n}$, n m-vectors, the m-dimensional data are projected onto a k-dimensional subspace, where $k \ll m$, using a random matrix $\mathbf{\Phi} \in \mathbb{R}^{k\times m}$:

$$\mathbf{B}_{\text{ran}} = \mathbf{\Phi}\mathbf{A}. \tag{2-10}$$

The key idea behind the RP is the Johnson-Lindenstrauss (JL) lemma [49]: n points in Euclidean space can be projected to k dimensions, where $k = c\epsilon^{-2}\log(n)$, c is a positive constant, and $\epsilon > 0$ while introducing a distortion of at most $1+\epsilon$. To be more precise, for any m-vectors x and y, the following holds with constant probability:

$$\sqrt{\frac{k}{m}}\|x-y\|_2(1-\epsilon) \le \|\mathbf{\Phi}x - \mathbf{\Phi}y\|_2 \le \sqrt{\frac{k}{m}}\|x-y\|_2(1+\epsilon). \tag{2-11}$$

The original projection matrix $\mathbf{\Phi}$ is characterized by the following three properties:

- Orthogonality: The columns of $\mathbf{\Phi}$ are orthogonal to each other,
- Normality: The columns of $\mathbf{\Phi}$ have unit length,
- Spherical symmetry: For orthogonal matrix $\mathbf{A}$, $\mathbf{\Phi}\mathbf{A}$ and $\mathbf{A}$ have the same distribution.

However, researchers showed that the JL guarantee (2-11) still holds by dropping these properties: orthogonality and the normality conditions were dropped by [44], and spherical symmetry condition by [1].

Beginning with [33], many randomized algorithms have been proposed for low-rank matrix approximations. The algorithms in [22, 25, 73], built on Frieze et al.'s idea [33], first sample columns of an input matrix with a probability proportional to either their magnitudes or leverage scores,

representing the matrix in a compressed form. The submatrix is then used for further computation (post-processing step) using deterministic algorithms such as the SVD and pivoted QR decomposition [35] to obtain the final low-rank approximation. Sarlós [74] proposes a different method based on results of the Johnson-Lindenstrauss (JL) lemma [49]. He showed that random linear combinations of rows, i.e., projecting the data matrix onto a structured random subspace, render a good approximation to a low-rank matrix. The works in [15, 66] have extended and improved Sarlós's idea and construct a low-rank approximation based on subspace embedding.

2.2.2 A Randomized Algorithm for PCA

Rokhlin et al. [71] employ random projections in order to approximate the matrix $\mathbf{A}$; first, the input matrix is projected onto a low dimensional random subspace by means of a random Gaussian matrix and, next, the low-rank approximation $\hat{\mathbf{A}}_{\text{ranPCA}}$ is given through computations on the reduced-sized matrix. Given $\mathbf{A}$ and an integer $k \leq \ell \leq \min\{m, n\}$, the proposed method [71] approximates $\mathbf{A}$ by taking the steps described in Algorithm 1.

Algorithm 1 Randomized PCA

Input: Matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$, integers k and ℓ.
Output: A low-rank approximation.

1: Draw a random matrix $\mathbf{G} \in \mathbb{R}^{\ell \times m}$ from a standard Gaussian distribution,
2: Compute $\mathbf{R} = \mathbf{GA}$,
3: Compute an SVD $\mathbf{R}^T = \mathbf{QSH}^T$,
4: Compute $\mathbf{T} = \mathbf{AQ}$,
5: Compute an SVD $\mathbf{T} = \mathbf{U\Sigma W}^T$,
6: Compute $\mathbf{V} = \mathbf{QW}$,
7: $\hat{\mathbf{A}}_{\text{ranPCA}} = \mathbf{U\Sigma V}^T$.

To obtain more accurate approximation, specifically for a matrix with slowly decaying singular values, the authors incorporate q steps of a power iteration [72]. Thus, the matrix $\mathbf{R}$, step 2 of Algorithm 1, is defined as follows:

$$\mathbf{R} = \mathbf{G}(\mathbf{AA}^T)^q\mathbf{A}. \tag{2-12}$$

and the rank-k approximation $\mathbf{A}_{\text{ran}}$ satisfies:

$$\|\mathbf{A} - \hat{\mathbf{A}}_{\text{ranPCA}}\|_2 \leq \beta m^{1/(4q+2)}\sigma_{k+1}, \tag{2-13}$$

with high probability, where β is a constant, and σ_{k+1} is the $(k+1)$-th singular value of $\mathbf{A}$. To approximate $\mathbf{A}$, this approach requires $2(q+1)$ passes over the data, for matrices stored out-of-core, and the flop counts satisfy

$$C_{\text{ranPCA}} \sim (2+2q)\ell C_{\text{mult}} + 2\ell^2(m+2n), \tag{2-14}$$

where C_{mult} is the cost of a matrix-vector multiplication with $\mathbf{A}$ or $\mathbf{A}^T$. A pass over the data is defined as visiting the data matrix by the algorithm to carry our the computations.

2.2.3 Randomized Algorithms for SVD

Halko et al. [39] propose two methods in order to approximate the SVD of a given matrix based on randomization. The first method, *randomized SVD*, for which the authors provide theoretical analysis and extensive numerical experiments, for the matrix $\mathbf{A}$ and integers $k \leq \ell < \min\{m,n\}$ and q, is described in Algorithm 2.

Algorithm 2 Randomized SVD (R-SVD)

Input: Matrix $\mathbf{A} \in \mathbb{R}^{m\times n}$, integers k, ℓ and q.
Output: A rank-ℓ approximation.

1: Draw a Gaussian random matrix $\mathbf{\Omega} \in \mathbb{R}^{n\times\ell}$;
2: Compute $\mathbf{Y} = (\mathbf{A}\mathbf{A}^T)^q\mathbf{A}\mathbf{\Omega}$;
3: Compute a QR decomposition $\mathbf{Y} = \mathbf{Q}\mathbf{R}$;
4: Compute $\mathbf{B} = \mathbf{Q}^T\mathbf{A}$;
5: Compute an SVD $\mathbf{B} = \widetilde{\mathbf{U}}\mathbf{\Sigma}\mathbf{V}^T$;
6: $\hat{\mathbf{A}}_{\text{RSVD}} = (\mathbf{Q}\widetilde{\mathbf{U}})\mathbf{\Sigma}\mathbf{V}^T$.

The R-SVD approximates $\mathbf{A}$ as follows: (i) a compressed matrix $\mathbf{Y}$, through random linear combinations of columns of $\mathbf{A}$ is formed, (ii) a QR decomposition is performed on $\mathbf{Y}$, where the $\mathbf{Q}$ factor constructs an approximate basis for $\mathcal{R}(\mathbf{A})$, (iii) $\mathbf{A}$ is projected onto a subspace spanned by columns of $\mathbf{Q}$, forming $\mathbf{B}$, (iv) a full SVD of $\mathbf{B}$ is computed. In Algorithm 2, q is the number of steps of a power method [39, 71]. $\mathbf{A}_{\text{RSVD}}$ satisfies

$$\mathbb{E}\|\mathbf{A} - \mathbf{A}_{\text{RSVD}}\|_2 \leq \left[1 + 4\sqrt{\frac{2\min\{m,n\}}{k-1}}\right]^{1/(2q+1)} \sigma_{k+1}, \tag{2-15}$$

where $\mathbb{E}$ denotes the expectation operator, and σ_{k+1} is the $(k+1)$-th singular value of $\mathbf{A}$. To decompose $\mathbf{A}$, the R-SVD algorithm requires $2(q+1)$ passes over the data, for matrices stored externally, and the flop counts satisfy

$$C_{\text{RSVD}} \sim (2q+2)\ell C_{\text{mult}} + 2\ell^2(m+n), \tag{2-16}$$

where C_{mult} is the cost of a matrix-vector multiplication with $\mathbf{A}$ or $\mathbf{A}^T$. The cost in (2-16) results from a dense matrix $\mathbf{A}$. If $\mathbf{A}$ is sparse, the arithmetic cost is proportional to the number s of non-zero entries of $\mathbf{A}$, satisfying

$$C_{\text{RSVD}} \sim (2q+2)\ell s + 2\ell^2(m+n). \tag{2-17}$$

If the Gaussian random matrix Ω is replaced by a random matrix with internal structure such as the the subsampled randomized Hadamard transform (SRHT) [87], the number of flops will be reduced. An SRHT matrix $\mathbf{\Omega} \in \mathbb{R}^{n\times\ell}$ has the following form:

$$\mathbf{\Omega} = \sqrt{\frac{n}{\ell}}\mathbf{RHD}, \tag{2-18}$$

where

- $\mathbf{D} \in \mathbb{R}^{n\times n}$ is diagonal whose entries are independently drawn from $\{-1, 1\}$,
- $\mathbf{H} \in \mathbb{R}^{n\times n}$ is a Walsh-Hadamard matrix scaled by $n^{-1/2}$,
- $\mathbf{R} \in \mathbb{R}^{\ell\times n}$ is a sparse matrix whose rows are samples, without replacement, from the standard basis of $\mathbb{R}^n$.

By defining $\mathbf{\Omega}$ as in (2-18), the product $\mathbf{A\Omega}$ is computed in $O(mn\log(\ell))$ flops [87]. As a result, flop counts of the RSVD satisfy

$$C_{\text{RSVD}} \sim mn\log(\ell) + (2q+1)\ell C_{\text{mult}} + 2\ell^2(m+n). \tag{2-19}$$

Gu [36] applies a slightly modified version of the R-SVD algorithm to improve subspace iteration methods, and presents a new error analysis. The second method proposed in [39, Section 5.5] is a *single-pass* algorithm, i.e., it requires only one pass through data, to compute a low-rank approximation of a given matrix. For the matrix $\mathbf{A}$, the decomposition, which we call two-sided randomized SVD (TSR-SVD), is computed as described in Algorithm 3.

Algorithm 3 Two-Sided Randomized SVD (TSR-SVD)

Input: Matrix $\mathbf{A} \in \mathbb{R}^{m\times n}$, integers k and ℓ.
Output: A rank-ℓ approximation.

1: Draw random matrices $\mathbf{\Psi}_1 \in \mathbb{R}^{n\times\ell}$ and $\mathbf{\Psi}_2 \in \mathbb{R}^{m\times\ell}$;
2: Compute $\mathbf{Y}_1 = \mathbf{A\Psi}_1$ and $\mathbf{Y}_2 = \mathbf{A}^T\mathbf{\Psi}_2$ in a single pass through $\mathbf{A}$;
3: Compute QR decompositions $\mathbf{Y}_1 = \mathbf{Q}_1\mathbf{R}_1$, $\mathbf{Y}_2 = \mathbf{Q}_2\mathbf{R}_2$;
4: Compute $\mathbf{B}_{\text{approx}} = \mathbf{Q}_1^T\mathbf{Y}_1(\mathbf{Q}_2^T\mathbf{\Psi}_1)^\dagger$;
5: Compute an SVD $\mathbf{B}_{\text{approx}} = \widetilde{\mathbf{U}}\widetilde{\mathbf{\Sigma}}\widetilde{\mathbf{V}}$;
6: $\hat{\mathbf{A}}_{\text{TSR}} = (\mathbf{Q}_1\widetilde{\mathbf{U}})\widetilde{\mathbf{\Sigma}}(\mathbf{Q}_2\widetilde{\mathbf{V}})^T$.

In Algorithm 3, $\mathbf{B}_{\text{approx}}$ is an approximation to $\mathbf{B} = \mathbf{Q}_1^T\mathbf{AQ}_2$, $\mathbf{Q}_1\widetilde{\mathbf{U}}$ is an approximation to the left subspace, $\mathbf{Q}_2\widetilde{\mathbf{V}}$ is an approximation to the right subspace, and $\widetilde{\mathbf{\Sigma}}$ is an approximation to the first ℓ singular values of $\mathbf{A}$.

2.2.4 Sketching-based Fixed-Rank Approximation

Tropp et al. [86] propose a suite of low-rank approximation methods of a given matrix by making use of a sketch of the matrix. For the matrix $\mathbf{A}$, its sketch is formed in a single pass through $\mathbf{A}$ to capture the action of the matrix, and further processing is performed on the sketch by means of deterministic methods to construct the low-rank approximation. The algorithm proposed in [86, Algorithm 7], which we call sketching fixed-rank approximation (SFRA), is presented in Algorithm 4.

Algorithm 4 Sketching-based Fixed-Rank Approximation (SFRA)

Input: Matrix $\mathbf{A} \in \mathbb{R}^{m\times n}$, target rank k, sketch size parameters (p_1, p_2).
Output: A rank-k approximation.

1: Draw two random matrices $\mathbf{\Omega} \in \mathbb{R}^{n\times p_1}$, $\mathbf{\Psi} \in \mathbb{R}^{p_2\times m}$;
2: Form sketches of $\mathbf{A}$: $\mathbf{Y} = \mathbf{A\Omega}$, $\mathbf{W} = \mathbf{\Psi A}$;
3: Compute a QR decomposition $\mathbf{Y} = \mathbf{Q}_{\mathrm{y}}\mathbf{R}_{\mathrm{y}}$;
4: Compute a QR decomposition $\mathbf{\Psi Q}_{\mathrm{y}} = \mathbf{QR}$;
5: Form $\mathbf{X} = \mathbf{R}^{\dagger}(\mathbf{Q}^T\mathbf{W})$
6: Compute a rank-k truncated SVD $\mathbf{X} = \mathbf{U}_k\mathbf{\Sigma}_k\mathbf{V}_k^T$;
7: $\hat{\mathbf{A}}_{\mathrm{SFRA}} = (\mathbf{QU}_k)\mathbf{\Sigma}_k\mathbf{V}_k^T$.

The key difference between TSR-SVD and SFRA, however, is that in order to capture the action of $\mathbf{A}^T$, the former projects $\mathbf{A}$ onto a subspace spanned by its rows, whereas the latter projects $\mathbf{A}$ onto a random subspace, i.e., a subspace spanned by columns of $\mathbf{\Psi}$. The limitation of SFRA, as pointed out by authors [86], is that it can not treat all low-rank matrix approximation problems, rather it can be applied in situations where it is only possible to make a single pass through the input matrix.

2.3 Comparison of Deterministic and Randomized Algorithms for Image Reconstruction

In this section, we assess the quality of low-rank approximation computed by the algorithms discussed by reconstructing a gray-scale image of a differential gear of size 1280×804, taken from [26]. The results are shown in Figures 2.1 and 2.2; Figure 2.1 shows the reconstructed images of the differential gear with $rank = 70$, and Figure 2.2 displays the Frobenius-norm approximation error against the corresponding approximation rank, where the error is calculated as:

$$e_{\mathrm{approx}} = \|\mathbf{A} - \hat{\mathbf{A}}_{\mathrm{approx}}\|_F, \tag{2-20}$$

where $\hat{\mathbf{A}}_{\text{approx}}$ is the approximation computed by each algorithm. Judging from Figure 2.1, with a careful scrutiny, small defects appear in reconstructions by randomized algorithms with no power iteration, i.e., randomized PCA, R-SVD, TSR-SVD, as well as SFRA. While reconstructed images by deterministic algorithms (the SVD, QRCP, UTV) as well as randomized algorithms with one step of power iteration are visually indistinguishable from the original.

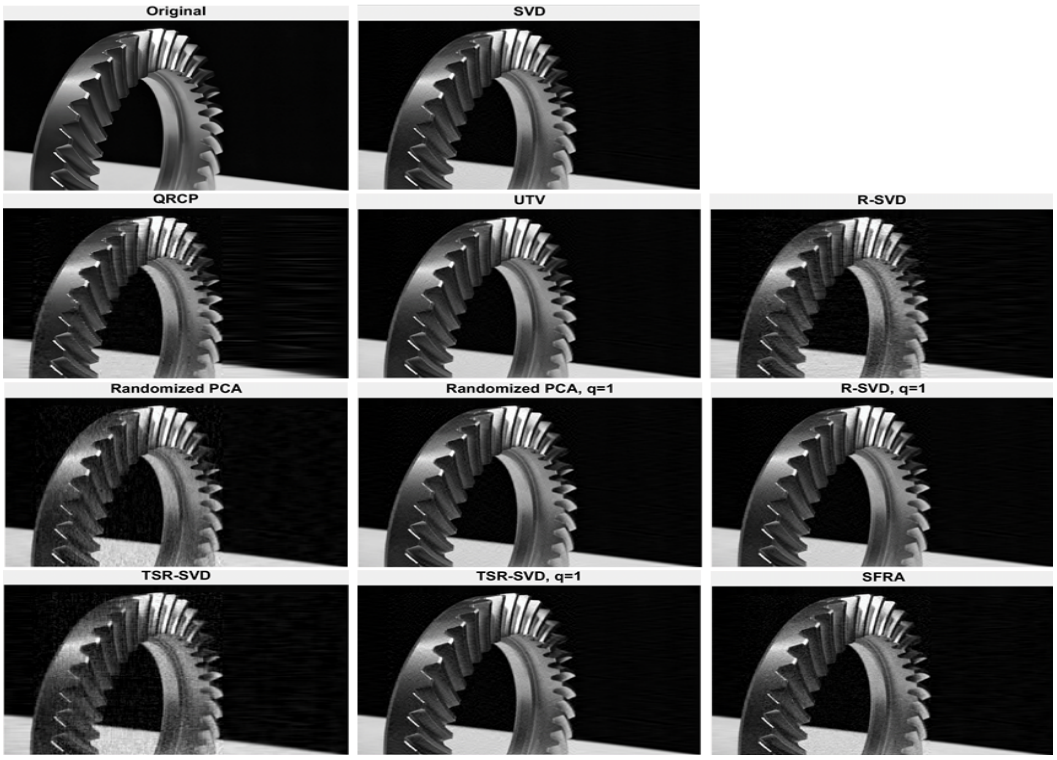


Figure 2.1: Low-rank image reconstruction. Approximations of a 1280×804 differential gear image are computed with $rank = 70$.

2.4 Computational and communication costs

In this section, we briefly describe the costs associated with an algorithm. The cost of any algorithm involves [21]:

1. Arithmetic which is floating-point arithmetic operations such as addition, multiplication, or division of two floating-point numbers.

2. Communication which is data movement between different levels of a memory hierarchy on a sequential machine (see Figure 2.3a), or data movement between processors working in parallel on a parallel machine (see Figure 2.3b).

Communication costs involve both bandwidth costs, which are proportional to the number of words of data sent, and latency costs, which are pro-

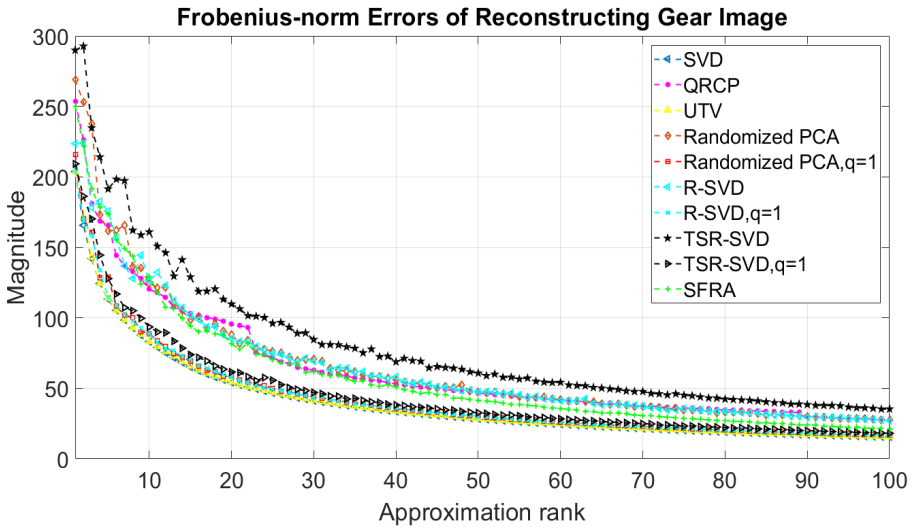


Figure 2.2: Errors incurred by different algorithms in reconstructing the differential gear image.

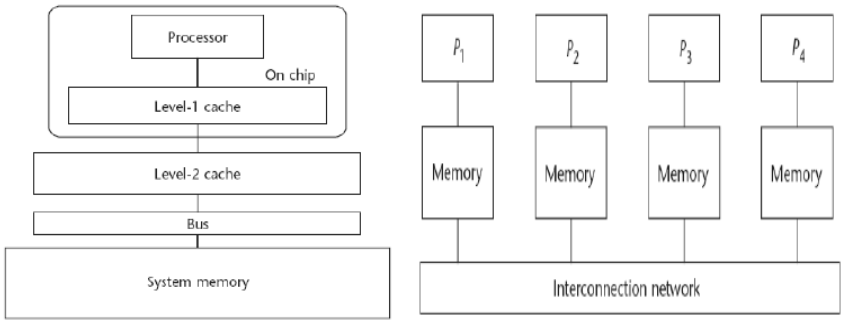
portional to the number of messages in which the data is sent. On high performance computing architectures, for a data matrix stored externally, communication costs become substantially more expensive compared to the arithmetic, and the gap is increasing rapidly for technological reasons [21, 23]. Therefore, developing new algorithms or redesigning existing algorithms to solve a problem in hand with minimum communication costs is highly desirable.

In this thesis we focus on developing randomized methods for low-rank matrix approximations, and providing mathematical analysis for them. We furnish arithmetic costs for the proposed algorithms, and comment on their communication costs. However, a detailed study of communication costs of the algorithms and implementing them on advanced computational platforms is beyond the scope of this work.

2.5 Principal Component Analysis

Principal component analysis (PCA) [46, 50] is a linear dimensionality reduction technique that tranforms a data matrix to a lower-dimensional subspace that captures most features of the data. In particular, PCA seeks to reduce the dimensionality of a data matrix, containing a large number of interrelated variables, by finding a few orthogonal linear combinations of the original variables with the largest variance. Given an input matrix $\mathbf{A} \in \mathbb{R}^{m\times n}$, where $m \geq n$, first the covariance matrix is formed by

$$\mathbf{\Sigma}_{n\times n} = \frac{1}{m}(\mathbf{A} - \mu)^T(\mathbf{A} - \mu), \tag{2-21}$$



(a) A uniprocessor memory hierarchy (b) A distributed-memory architecture

Figure 2.3: Memory architectures [45]. (a) The processor and a level-1 cache memory are on one chip, and a level-2 cache lies between the chip and the memory. (b) Each processor has a memory, and communication between processors is done over an interconnection network.

where μ contains the mean of $\mathbf{A}$. Next, using the spectral decomposition theorem [18, 35], $\mathbf{\Sigma}$ is written as:

$$\mathbf{\Sigma} = \mathbf{W}\mathbf{\Lambda}\mathbf{W}^{\mathbf{T}}, \tag{2-22}$$

where $\mathbf{W}$ has orthonormal columns containing eigenvectors of $\mathbf{\Sigma}$, and $\mathbf{\Lambda} = \text{diag}(\lambda_1, ..., \lambda_n)$ contains the eigenvalues of $\mathbf{\Sigma}$. The (full) principal components (PCs) are then given by

$$\mathbf{B} = \mathbf{AW}. \tag{2-23}$$

It has been shown [46,50] that, given $k < n$, the first k PCs ($\mathbf{B}_k = \mathbf{AW}_k$) capture the most important information in $\mathbf{A}$. The computational cost for PCA is $O(mn^2 + n^3)$ flops.

2.6 Robust PCA

PCA is well-known to be very sensitive to grossly corrupted observations; a single grossly corrupted element in the observation matrix can render the approximated matrix far from true. After a long line of research to robustifying PCA against grossly perturbed observations, robust PCA [9, 14, 93] was proposed. Robust PCA assumes that the data matrix $\mathbf{X} \in \mathbb{R}^{m \times n}$ consists of a linear superposition of two matrices such that

$$\mathbf{X} = \mathbf{L} + \mathbf{S}, \tag{2-24}$$

where $\mathbf{L}$ is a low-rank matrix, i.e., $\text{rank}(\mathbf{L}) \ll \min\{m, n\}$, and $\mathbf{S}$ is a sparse matrix of corrupted entries, i.e, $\|\mathbf{S}\|_0 \ll mn$, where $\|\cdot\|_0$ denotes the ℓ_0-norm of the matrix (number of nonzero entries). Robust PCA has been initially proposed in [93] to solve the following optimization problem:

$$\begin{aligned} &\text{minimize}_{(\mathbf{L},\mathbf{S})} \text{rank}(\mathbf{L}) + \eta\|\mathbf{S}\|_0 \\ &\text{subject to } \mathbf{L}+\mathbf{S}=\mathbf{X}, \end{aligned} \tag{2-25}$$

where $\eta > 0$ is a weighting parameter. Unfortunately, both the rank minimization and the ℓ_0-norm minimization problems are NP-hard [89], [65]. Therefore, to get a tractable optimization problem, (2-25) is relaxed by replacing the rank with the nuclear norm [29], and the ℓ_0-norm with the ℓ_1-norm [10], leading to the convex program:

$$\begin{aligned} &\text{minimize}_{(\mathbf{L},\mathbf{S})} \ \|\mathbf{L}\|_* + \lambda\|\mathbf{S}\|_1 \\ &\text{subject to } \mathbf{L}+\mathbf{S}=\mathbf{X}, \end{aligned} \tag{2-26}$$

where, for any matrix $\mathbf{B}$, $\|\mathbf{B}\|_* \triangleq \sum_i \sigma_i(\mathbf{B})$ is the nuclear norm of $\mathbf{B}$ (sum of the singular values), $\|\mathbf{B}\|_1 \triangleq \sum_{ij} |\mathbf{B}_{ij}|$ is the ℓ_1-norm of $\mathbf{B}$, and $\lambda > 0$ is a weighting parameter. The iterative thresholding (IT) algorithm [93] solves the following relaxed version of (2-26):

$$\begin{aligned} &\text{minimize}_{(\mathbf{L},\mathbf{S})} \ \|\mathbf{L}\|_* + \lambda\|\mathbf{S}\|_1 + \frac{1}{2\gamma}\|\mathbf{L}\|_F^2 + \frac{1}{2\gamma}\|\mathbf{S}\|_F^2 \\ &\text{subject to } \mathbf{L}+\mathbf{S}=\mathbf{X}, \end{aligned} \tag{2-27}$$

where $\|\mathbf{B}\|_F \triangleq \sqrt{\text{trace}(\mathbf{B}^T\mathbf{B})}$ is the Frobenius norm of the matrix $\mathbf{B}$ and γ is a large positive scalar. The solution pair $(\mathbf{L}^*, \mathbf{S}^*)$ is given after iteratively minimizing the Lagrangian function of (2-27) with respect to $\mathbf{L}$, $\mathbf{S}$ and $\mathbf{Y}$:

$$\mathcal{L}(\mathbf{L},\mathbf{S},\mathbf{Y}) \triangleq \|\mathbf{L}\|_* + \lambda\|\mathbf{S}\|_1 + \frac{1}{2\gamma}\|\mathbf{L}\|_F^2 + \frac{1}{2\gamma}\|\mathbf{S}\|_F^2 + \frac{1}{\gamma}\langle\, \mathbf{Y}, \mathbf{X}-\mathbf{L}\ -\mathbf{S}\rangle, \tag{2-28}$$

where $\mathbf{Y} \in \mathbb{R}^{m\times n}$ is a matrix of Lagrange multipliers, and $\langle \mathbf{A},\mathbf{B}\rangle \triangleq \text{trace}(\mathbf{B}^T\mathbf{A})$ is the inner product of matrices $\mathbf{A}$ and $\mathbf{B}$. The work in [9] solves (2-26) via the method of augmented Lagrange multipliers (ALM) [59, 95], and terms the approach Principal Component Pursuit (PCP). The ALM method operates on the augmented Lagrangian function of (2-26):

$$\mathcal{L}(\mathbf{L},\mathbf{S},\mathbf{Y},\mu) \triangleq \|\mathbf{L}\|_* + \lambda\|\mathbf{S}\|_1 + \langle\, \mathbf{Y}, \mathbf{X}-\mathbf{L}\ -\mathbf{S}\rangle + \frac{\mu}{2}\|\mathbf{X}-\ \mathbf{L}\ -\mathbf{S}\|_F^2, \tag{2-29}$$

where $\mathbf{Y} \in \mathbb{R}^{m\times n}$ is a matrix of Lagrange multipliers, $\mu > 0$ is a penalty parameter. The optimal solution pair $(\mathbf{L}^*, \mathbf{S}^*)$ is given after iteratively minimizing (2-29) with respect to $\mathbf{L}$ (while fixing $\mathbf{S}$), and then with respect to $\mathbf{S}$ (while fixing $\mathbf{L}$), i.e., the following two equations:

$$\mathbf{L}_{j+1} = \arg\min_{\mathbf{L}} \mathcal{L}(\mathbf{L},\mathbf{S},\mathbf{Y}) = \mathcal{D}_{\frac{1}{\mu}}(\mathbf{X}-\mathbf{S}+\frac{1}{\mu}\mathbf{Y}), \tag{2-30}$$

$$\mathbf{S}_{j+1} = \arg\min_{\mathbf{S}} \mathcal{L}(\mathbf{L},\mathbf{S},\mathbf{Y}) = \mathcal{S}_{\frac{\lambda}{\mu}}(\mathbf{X}-\mathbf{L}+\frac{1}{\mu}\mathbf{Y}), \tag{2-31}$$

where $\mathcal{D}_\varepsilon(\mathbf{B}) = \mathbf{U}\mathcal{S}_\varepsilon(\mathbf{\Sigma})\mathbf{V}^T$ is a singular-value thresholding operator [7], where $\mathbf{B} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$ is a singular value decomposition, and $\mathcal{S}_\varepsilon(x) = \text{sgn}(x)\text{max}(|x| - \varepsilon, 0)$ is a soft-thresholding (shrinkage) operator [38,84]. The pseudocode of the ALM method for RPCA is given in Table 2.1.

Table 2.1: Pseudo-code of robust PCA solved by ALM.

Input: Matrix $\mathbf{X}, \lambda, \mu, \mathbf{Y}_0 = \mathbf{S}_0 = 0, j = 0$;
Output: Low-rank plus sparse matrix
1: **while** the algorithm does not converge **do**
2: Compute $\mathbf{L}_{j+1} = \mathcal{D}_{\mu^{-1}}(\mathbf{X} - \mathbf{S}_j + \mu^{-1}\mathbf{Y}_j)$;
3: Compute $\mathbf{S}_{j+1} = \mathcal{S}_{\lambda\mu^{-1}}(\mathbf{X} - \mathbf{L}_{j+1} + \mu^{-1}\mathbf{Y})$;
4: Compute $\mathbf{Y}_{j+1} = \mathbf{Y}_j + \mu(\mathbf{X} - \mathbf{L}_{j+1} - \mathbf{S}_{j+1})$;
5: **end while**
6: **return** $\mathbf{L}^*$ and $\mathbf{S}^*$

2.7 Comparison of PCA and Robust PCA for Background Modeling in Surveillance Video

In this section, we compare PCA and robust PCA methods for separating background and foreground in a video stream taken from [58] (for the PCA method 5 PCs have been used). The results are shown in Figure 2.4. In the background recovered by PCA ghostly artifacts appear showing that PCA can not completely separate moving objects from the background. Furthermore, substantial defects appear in the foreground. While robust PCA successfully model the background and foreground.

2.8 Switched-Randomized Robust PCA

This section presents a new fast robust PCA technique termed switched-randomized robust PCA (SR-RPCA) [51] and applies it in application of background subtraction in surveillance videos.

The ALM method applied to solve the robust PCA problem yields the optimal solution, however, its major bottleneck is computing a computationally demanding SVD at each iteration to approximate the low-rank component $\mathbf{L}$ of $\mathbf{X}$. To address this concern and speed up the convergence of the ALM

(a) Background modeling with PCA (b) Background modeling with robust PCA

Figure 2.4: Background modeling in surveillance video. In (a) and (b), images in column 1 are frames of the surveillance video, images in column 2 are recovered backgrounds, and column 3 corresponds to foregrounds recovered by the algorithms.

method, the work in [59] proposes a few techniques including predicting the principal singular space dimension, a continuation technique [85], and a truncated SVD by using PROPACK package [57]. The modified algorithm [59], called `InexactALM` hereafter in this dissertation, substantially improves the convergence speed, however the bottleneck is that the truncated SVD [57] employed uses the lanczos algorithm that is inherently unstable and, moreover, due to the limited data reuse in its operations it has very poor performance on modern architectures [8, 35, 36, 39].

To address this issue, we thus, by retaining the original objective function proposed in [9, 14, 59, 93], replace the SVD with an approximation; the approximate left and right singular vectors of the input matrix are drawn from the column and row spaces, respectively, using the bilateral projections technique [97], through switching among different random matrices. Furthermore, to obtain the corresponding singular values, a technique that employs the Weibull distribution [48] is used to estimate the singular values of the matrix. After convergence of the proposed robust PCA algorithm, to guarantee a sparse error matrix SR-RPCA uses a hard thresholding operator [4] to keep only the largest elements in the sparse matrix. The SR-RPCA method is applied for background modeling in surveillance videos. We also apply the method on the data matrix partitioned with two different schemes.

2.8.1 The Bilateral Projections Technique

The bilateral projections technique [97], termed bilateral random projections (BRP), is a fast method to approximate a rank-r matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$ as

described in Algorithm 5.

Algorithm 5 The Bilateral Projections Technique

Input: Matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$, integer r.
Output: A rank-k approximation.

1: Draw a random matrix $\mathbf{B}_2 \in \mathbb{R}^{n \times r}$;
2: **for** $j = 1$: $q + 1$ **do**
3: Compute $\mathbf{B}_1 = \mathbf{A}\mathbf{B}_2$;
4: Compute $\mathbf{B}_2 = \mathbf{A}^T\mathbf{B}_1$;
5: **end for**
6: Compute QR decompositions $\mathbf{B}_1 = \mathbf{Q}_1\mathbf{R}_1$, $\mathbf{B}_2 = \mathbf{Q}_2\mathbf{R}_2$;
7: Form the rank-k approximation $\hat{\mathbf{A}}_{\text{BRP}} = \mathbf{Q}_1[\mathbf{R}_1(\mathbf{B}_1{}^T\mathbf{B}_1)^{-1}\mathbf{R}_2{}^T]^{1/2q+1}\mathbf{Q}_2{}^T$.

In Algorithm 5, $\mathbf{T}_1 \in \mathbb{R}^{m \times r}$ is obtained by a right multiplication of $\mathbf{A}$ with a random matrix $\mathbf{B}_2 \in \mathbb{R}^{n \times r}$, and $\mathbf{B}_2$ is then updated by a left multiplication of $\mathbf{B}_1$. $\mathbf{Q}_1$ and $\mathbf{Q}_2$ are approximations to the left and right singular vectors of $\mathbf{A}$, respectively. The integer q corresponds to the number of steps of a power iteration scheme [71, 72].

2.8.2 Singular Values Estimation Technique

To compute an SVD-like low-rank approximation of a given matrix $\mathbf{A}$, we first compute $\mathbf{Q}_1$ and $\mathbf{Q}_2$ via the bilateral projections technique (Algorithm 5). Next, we estimate the singular values of $\mathbf{A}$ which are then incorporated with $\mathbf{Q}_1$ and $\mathbf{Q}_2$ to approximate the SVD of $\mathbf{A}$. The proposed technique is based on the observation of data from surveillance videos that if the matrix is decomposable into a low-rank and a sparse component, the singular value distribution follows the Weibull distribution [48]. First, random numbers are generated using the Weibull distribution and normalized to be between 0 and 1. Then, they are multiplied by the largest singular value of the data matrix obtained via the R-SVD Algorithm 2. Experimental results show that the estimated singular values are fairly close to the leading singular values of $\mathbf{A}$. For our experiment, we determine the rank r by applying the following inequality [2], which relates the numerical rank r of any matrix $\mathbf{B}$ with the ℓ_2 and Frobenius norms:

$$\|\mathbf{A}\|_2 \geq \frac{\|\mathbf{A}\|_F}{\sqrt{r}} \tag{2-32}$$

In order to obtain more accurate approximations $\mathbf{Q}_1$ and $\mathbf{Q}_2$, we use four different random matrices $\mathbf{B}_2$ generated as follows:

- a matrix with i.i.d Gaussian entries i.e., $\mathcal{N}(0, 1)$,
- a matrix whose entries are i.i.d. random variables drawn from a uniform distribution in the interval (0, 1),

– a Markov matrix whose entries are all non-negative and entries of each column add up to 1,

– a matrix whose entries are independently drawn from {-1, 1}.

The SR-RPCA method switches among different random matrices and chooses the best one in order to obtain the solution pair ($\mathbf{L}^*$, $\mathbf{S}^*$) with lower distortion. To guarantee the sparse structure of the chosen error matrix $\mathbf{S}^*$, the SR-RPCA approach uses a hard thresholding operator $\mathcal{H}_s(\cdot)$ [4]. $\mathcal{H}_s(\cdot)$ is a nonlinear operator that keeps the largest s entries (in magnitude) of a matrix it operates on, and sets all other entries to zero. The pseudo-code of the proposed method is given in Table 2.2.

Table 2.2: Pseudo-code for the SR-RPCA algorithm.

Input: Matrix $\mathbf{X} \in \mathbb{R}^{m \times n}, \lambda, \mu_0, \bar{\mu}, \rho, \mathbf{Y}_0, \mathbf{S}_0, k = 0$;

Generate four random matrices;
for each random matrix **do**
  Compute $\mathbf{Q}_1$ and $\mathbf{Q}_2$ using the bilateral projections technique;
  while the algorithm does not converge **do**
    Estimate singular values $\mathbf{\Lambda}$;
    Determine the numerical rank r;
    $\mathbf{L}_{k+1} = \mathbf{Q}_1(:, 1:r)\mathbf{\Lambda}_r\mathbf{Q}_2(:, 1:r)^T; \rightarrow \mathbf{\Lambda}_r = \text{diag}(\mathbf{\Lambda}(1:r))$
    $\mathbf{S}_{k+1} = \mathcal{S}_{\lambda\mu_k^{-1}}(\mathbf{X} - \mathbf{L}_{k+1} + \mu_k^{-1}\mathbf{Y})$;
    $\mathbf{Y}_{k+1} = \mathbf{Y}_k + \mu_k(\mathbf{X} - \mathbf{L}_{k+1} - \mathbf{S}_{k+1})$;
    $\mu_{k+1} \leftarrow \max(\rho\mu_k, \bar{\mu})$;
  end while
end for
Choose the lowest error & corresponding random projection matrix & return ($\mathbf{L}^\circ$, $\mathbf{S}^\circ$);
Apply the hard-thresholding operator: $\mathcal{H}_s(\mathbf{S}^\circ)$;
return $\mathbf{L}^\circ$ and $\mathbf{S}^\circ$.

2.8.3 Partitioning Input Matrix

We can further perform SR-RPCA on the partitioned input matrices. The advantage of partitioning the data matrix into submatrices are twofold (i) it enables to parallelize the operations on the matrix, and (ii) it reduce memory requirements. The disadvantage, however, is that the recovery guarantee provided in [9, 14] is less likely to be satisfied for each block meaning that the probability of obtaining the exact solution by concatenating the solution of each block is reduced. For partitioning, we use two partitioning schemes,

column-wise and row-wise. In both cases, $\mathbf{X}$ is partitioned into small blocks, and we apply SR-RPCA to each block, and combine the solution of the corresponding blocks afterwards to recover the original matrix. For column-wise partitioning, consider Φ_i as a subset of columns of the data matrix $\mathbf{X}$ such that the entries of X_{Φ_i}, the ith block, are chosen between $1+(i-1)n/K$ and in/K where K is the number of submatrices as described by

$$\mathbf{X} = \left[\Phi_1|\Phi_2|\dots|\Phi_K\right] \tag{2-33}$$

For row-wise partitioning, consider Υ_j as a subset of rows of $\mathbf{X}$ such that the entries of X_{Υ_j}, the jth block, are chosen between $1+(j-1)m/P$ and jm/P, where P is the number of submatrices as given by

$$\mathbf{X} = \left[\Upsilon_1^T|\Upsilon_2^T|\dots|\Upsilon_P^T\right]^T \tag{2-34}$$

2.8.4 Experiments

We conduct experiments on two real-time videos introduced in [58]. Both video streams have 200 grayscale frames. One has dimensions 176×144 in each frame taken in an airport, and the other one has dimensions 120×160 in each frame taken in a buffet restaurant. The data matrices are obtained through concatenating 200 frames, i.e., $\mathbf{X} \in \mathbb{R}^{25344\times 200}$ and $\mathbf{X} \in \mathbb{R}^{19200\times 200}$.

We set the initial values of the SR-RPCA method as suggested by [59], and compare the results with those of [59]. Both algorithms stop when the following stopping condition holds:

$$\frac{\|\mathbf{X} - \mathbf{L}^{\text{sol}} - \mathbf{S}^{\text{sol}}\|_F}{\|\mathbf{X}\|_F} < 10^{-7}, \tag{2-35}$$

where ($\mathbf{L}^{\text{sol}}$,$\mathbf{S}^{\text{sol}}$) is the pair of solution of either algorithm. Figure 2.5 shows the recovered backgrounds and foregrounds of two sample frames of video streams by the SR-RPCA method. We do not show the results of the RPCA algorithm, as well as the SR-RPCA on partitioned data matrix since they are visually identical to those presented.

If the SR-RPCA method is used in full power, i.e., all four random matrices are used, the computational time for the algorithm to converge, say t, is close to the work in [59]. However, if only one of the random matrices is used, roughly speaking, the computational time should be divided by 4, yielding $t/4$. Table 2.3 summarizes the computational time for the two methods.

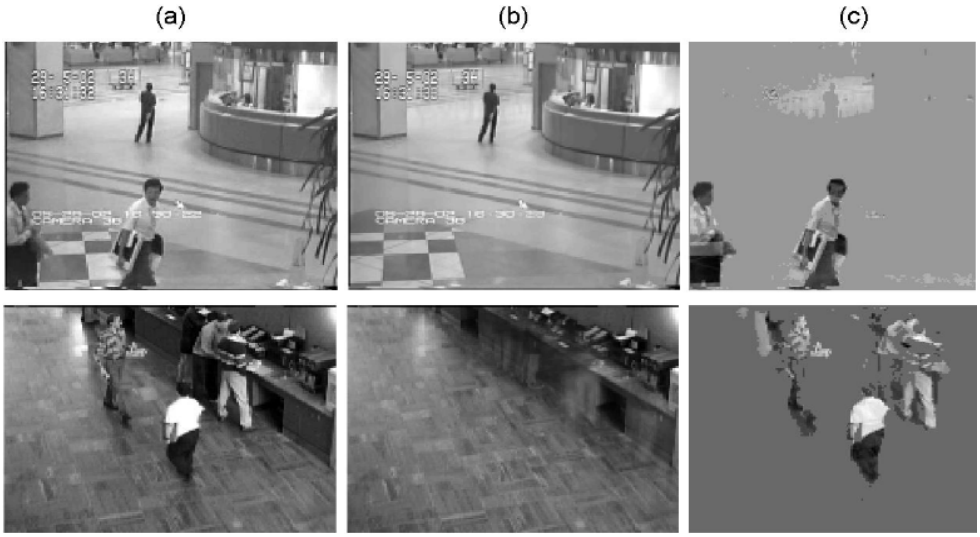


Figure 2.5: Background modeling in surveillance videos. Images in (a) are frames of the video streams. Images in (b) and (c) are recovered backgrounds and foregrounds by the SR-RPCA method, respectively.

Table 2.3: Computational time (in seconds). For the SR-RPCA method, all four random matrices are used.

Methods	Airport hall	Buffet restaurant
RPCA [59]	41	31
SR-RPCA	46	32
SR-RPCA on column-wise partitioned data matrix ($K = 5$)	42	31
SR-RPCA on row-wise partitioned data matrix ($P = 6$)	39	29

3
Randomized Rank-Revealing UZV Decomposition

This chapter presents a new rank-revealing algorithm termed randomized rank-revealing UZV decomposition (RRR-UZVD) [53]. The work of this chapter serves as the basis for the algorithms presented in the next two chapters.

The RRR-UZVD first, through randomization, constructs orthonormal bases for the column and row space of the input matrix. Second, the matrix is compressed by multiplying on the right and the left by the approximate bases. Third, columns of the compressed matrix and, accordingly, columns of the approximate bases are permuted. Finally, the low-rank approximation is given by projecting the small projected matrix back to the original space. The rank-revealing property of the proposed algorithm is proved. The RRR-UZVD is applied to reconstruct a low-rank image as well as to solve the robust PCA problem.

3.1
The RRR-UZVD Algorithm

The RRR-UZVD delivers information on singular values and singular subspaces of a matrix using randomization. Given a matrix $\mathbf{A} \in \mathbb{R}^{m\times n}$, where $m \geq n$, with numerical rank k and an integer $k \leq \ell < n$, RRR-UZVD constructs an approximation $\hat{\mathbf{A}}_{\text{UZV}}$ to $\mathbf{A}$ which takes the following form:

$$\hat{\mathbf{A}}_{\text{UZV}} = \mathbf{U}\mathbf{Z}\mathbf{V}^T = \mathbf{U}\begin{bmatrix}\mathbf{Z}_k & \mathbf{G}\\ \mathbf{H} & \mathbf{E}\end{bmatrix}\mathbf{V}^T, \tag{3-1}$$

where $\mathbf{U} \in \mathbb{R}^{m\times\ell}$ and $\mathbf{V} \in \mathbb{R}^{n\times\ell}$ have orthonormal columns. The matrix $\mathbf{Z}_k \in \mathbb{R}^{k\times k}$ is well-conditioned, and its diagonal elements are approximations of leading singular values of $\mathbf{A}$, and matrices $\mathbf{G} \in \mathbb{R}^{k\times\ell-k}$, $\mathbf{H} \in \mathbb{R}^{\ell-k\times k}$ and $\mathbf{E} \in \mathbb{R}^{\ell-k\times\ell-k}$ have sufficiently small ℓ_2-norms. We call diagonals of $\mathbf{Z} \in \mathbb{R}^{\ell\times\ell}$, $\mathbf{Z}$-*values* of $\mathbf{A}$.

The RRR-UZVD has the rank-revealing property in the sense that the rank k of $\mathbf{A}$ is revealed in the submatrix $\mathbf{Z}_k$, and the ℓ_2-norm of other submatrices are of the order σ_{k+1}; see Theorem 3.1. This is analogous to definitions of rank-revealing decompositions in the literature [11, 12, 37, 41, 79, 81]. The RRR-UZVD for the matrix $\mathbf{A}$ is computed as follows:

1. Generate a random test matrix $\mathbf{\Phi} \in \mathbb{R}^{n\times\ell}$,

2. Compute the matrix product:

$$\mathbf{Z}_1 = \mathbf{A}\mathbf{\Phi}. \tag{3-2}$$

3. Compute the matrix product:

$$\mathbf{Z}_2 = \mathbf{A}^T\mathbf{Z}_1. \tag{3-3}$$

4. Compute QR decompositions of $\mathbf{Z}_1$ and $\mathbf{Z}_2$:

$$\mathbf{Z}_1 = \mathbf{U}\mathbf{R}_1, \quad \text{and} \quad \mathbf{Z}_2 = \mathbf{V}\mathbf{R}_2. \tag{3-4}$$

5. Compute the matrix product:

$$\mathbf{Z} = \mathbf{U}^T\mathbf{A}\mathbf{V}. \tag{3-5}$$

6. Project the compressed data back to the original space, delivering a low-rank approximation:

$$\hat{\mathbf{A}}_{\text{UZV}} = \mathbf{U}\mathbf{Z}\mathbf{V}^T. \tag{3-6}$$

The matrix $\mathbf{Z}_1 \in \mathbb{R}^{m\times\ell}$ (3-2) is constructed by linear combinations of columns of $\mathbf{A}$ by $\mathbf{\Phi}$. The matrix $\mathbf{Z}_2 \in \mathbb{R}^{n\times\ell}$ (3-3) is formed by linear combinations of rows of $\mathbf{A}$ by $\mathbf{Z}_1$. The matrices $\mathbf{U}$ and $\mathbf{V}$ (3-4) are approximate bases for $\mathcal{R}(\mathbf{A})$ and $\mathcal{R}(\mathbf{A}^T)$, respectively, where $\mathcal{R}(\cdot)$ denotes the range of a matrix. The matrix $\mathbf{Z} \in \mathbb{R}^{\ell\times\ell}$ (3-5) is formed by compression of $\mathbf{A}$ through left and right multiplications by the approximate bases, and the diagonal of $\mathbf{Z}$ provides an approximation to singular values of $\mathbf{A}$.

The RRR-UZVD, described in its basic form, requires three passes through data. However, it can be modified to revisit $\mathbf{A}$ only once. To this end, the compressed matrix $\mathbf{Z}$ (3-5) can be computed by using currently available matrices as follows: both sides of the currently unknown $\mathbf{Z}$ are postmultiplied by $\mathbf{V}^T\mathbf{\Phi}$, i.e.,

$$\mathbf{Z}\mathbf{V}^T\mathbf{\Phi} = \mathbf{U}^T\mathbf{A}\mathbf{V}\mathbf{V}^T\mathbf{\Phi}. \tag{3-7}$$

Having defined $\mathbf{A} \approx \mathbf{A}\mathbf{V}\mathbf{V}^T$ and $\mathbf{Z}_1 = \mathbf{A}\mathbf{\Phi}$, an approximation to $\mathbf{Z}$ can be obtained by

$$\mathbf{Z}_{\text{approx}} = \mathbf{U}^T\mathbf{Z}_1(\mathbf{V}^T\mathbf{\Phi})^\dagger, \tag{3-8}$$

where † denotes the pseudo-inverse.

The RRR-UZVD may produce poor approximate bases and fuzzy singular values that deviate significantly from the exact ones (computed by the SVD), especially in applications where the matrix has slowly decaying singular values. Moreover, the orthonormal columns of $\mathbf{U}$ and $\mathbf{V}$ may not be necessar-

ily in a contributing order and, as a result, the $\mathbf{Z}$-*values* may not be in a non-increasing order. To address these concerns, we propose two techniques:

1. *Power iterations.* A few steps of a power method [39,71] can significantly improve the performance of the algorithm, due to alternately applying the sketch of the input matrix for projections.
2. *Column permutation.* The column reordering technique is implemented as follows: (i) sort the diagonal of $\mathbf{Z}$ according to their magnitudes (decreasing order), returning a permutation matrix $\mathbf{\Pi}$, (ii) post-multiply $\mathbf{U}$ and $\mathbf{V}$ by $\mathbf{\Pi}$: $\mathbf{U}_s = \mathbf{U}\mathbf{\Pi}$, and $\mathbf{V}_s = \mathbf{V}\mathbf{\Pi}$.

The modified RRR-UZVD is described in Algorithm 6.

Algorithm 6 The RRR-UZVD algorithm

Input: Matrix $\mathbf{A} \in \mathbb{R}^{m\times n}$, integers k, ℓ and q,
Output: A rank-ℓ approximation.

Draw a random test matrix $\mathbf{Z}_2 \in \mathbb{R}^{n\times\ell}$;
for $i = 1$: $q+1$ **do**
    Compute $\mathbf{Z}_1 = \mathbf{A}\mathbf{Z}_2$;
    Compute $\mathbf{Z}_2 = \mathbf{A}^T\mathbf{Z}_1$;
end for
Compute QR decompositions $\mathbf{Z}_1 = \mathbf{U}\mathbf{R}_1$ and $\mathbf{Z}_2 = \mathbf{V}\mathbf{R}_2$;
Compute $\mathbf{Z} = \mathbf{U}^T\mathbf{A}\mathbf{V}$ or $\mathbf{Z}_{\text{approx}} = \mathbf{U}^T\mathbf{Z}_1(\mathbf{V}^T\mathbf{Z}_2)^\dagger$;
Perform the column reordering technique, returning $\mathbf{U}_s$, $\mathbf{V}_s$, $\mathbf{Z}_{\text{approx}} = \mathbf{U}_s^T\mathbf{Z}_1(\mathbf{V}_s^T\mathbf{Z}_2)^\dagger$;
Form the low-rank approximation of $\hat{\mathbf{A}}_{\text{UZV}} = \mathbf{U}_s\mathbf{Z}_{\text{approx}}\mathbf{V}_s^T$.

3.2 Analysis of RRR-UZVD

In this section, we discuss the rank-revealing property and computational complexity of RRR-UZVD.

3.2.1 Rank-Revealing Property

For the matrix $\mathbf{A}$, and integers $k \le \ell \le n$ and q, the partitioned RRR-UZVD has the following form:

$$\hat{\mathbf{A}}_{\text{UZV}} = \mathbf{U}\mathbf{Z}\mathbf{V}^T = \begin{bmatrix}\mathbf{U}_1 & \mathbf{U}_2\end{bmatrix}\begin{bmatrix}\mathbf{Z}_k & \mathbf{G}\\ \mathbf{H} & \mathbf{E}\end{bmatrix}\begin{bmatrix}\mathbf{V}_1 & \mathbf{V}_2\end{bmatrix}^T, \tag{3-9}$$

where $\mathbf{U}_1 \in \mathbb{R}^{m\times k}$, $\mathbf{U}_2 \in \mathbb{R}^{m\times\ell-k}$, $\mathbf{V}_1 \in \mathbb{R}^{n\times k}$, $\mathbf{V}_2 \in \mathbb{R}^{n\times\ell-k}$, $\mathbf{Z}_k \in \mathbb{R}^{k\times k}$ is well-conditioned and its diagonals are approximations of leading singular values of $\mathbf{A}$. We show that $\mathbf{Z}_k$ reveals the rank, and submatrices $\mathbf{G} \in \mathbb{R}^{k\times\ell-k}$,

$\mathbf{H} \in \mathbb{R}^{\ell-k\times k}$, $\mathbf{E} \in \mathbb{R}^{\ell-k\times \ell-k}$ have small ℓ_2-norms. The following theorem states the rank-revealing property of RRR-UZVD. This result is new.

Theorem 3.1 *Let $\mathbf{A} \in \mathbb{R}^{m\times n}$, where $m \geq n$, be a matrix with numerical rank k whose SVD is defined in* (2-1)*, and its RRR-UZVD is defined in* (3-9)*. Then, we have*

$$\sigma_{min}(\mathbf{Z}_k) = O(\sigma_k), \tag{3-10}$$

$$\|[\mathbf{H} \quad \mathbf{E}]\|_2 = O(\sigma_{k+1}), \tag{3-11}$$

$$\|[\mathbf{G}^T \quad \mathbf{E}^T]^T\|_2 = O(\sigma_{k+1}). \tag{3-12}$$

Proof. The proof is given in Appendix 8.1.

3.2.2 Computational Complexity

To factor $\mathbf{A}$, the simple version of RRR-UZVD incurs the following costs: Step 1 costs $O(n\ell)$, Step 2 costs $O(mn\ell)$, Step 3 costs $O(mn\ell)$, Step 4 costs $O(m\ell^2+n\ell^2)$, Step 5 costs $O(mn\ell+m\ell^2)$ (if the matrix $\mathbf{Z}$ is approximated by $\mathbf{Z}_{\text{approx}}$ of equation (3-8) in this step, the cost would be $O(m\ell^2+n\ell^2+\ell^3)$). The column reordering technique costs $O(m\ell)$. The dominant cost of Step 1-6 occurs when multiplying $\mathbf{A}$ and $\mathbf{A}^T$ with the corresponding matrices. Thus

$$C_{\text{UZV}} = O(mn\ell). \tag{3-13}$$

The sample size parameter ℓ is typically close to the rank k. RRR-UZVD requires either three or two passes (when $\mathbf{Z}$ is approximated by $\mathbf{Z}_{\text{approx}}$) over data to factor $\mathbf{A}$. When the power method is used (Algorithm 6), RRR-UZVD requires either $(2q+3)$ or $(2q+2)$ passes (when $\mathbf{Z}$ is approximated by $\mathbf{Z}_{\text{approx}}$) over data with arithmetic costs of $(2q+3)C_{\text{UZV}}$ or $(2q+2)C_{\text{UZV}}$, respectively.

The RRR-UZVD, except for matrix-matrix multiplications which are readily parallelizable, performs two QR decompositions on matrices of size $m \times \ell$ and $n \times \ell$, whereas the R-SVD performs one QR decomposition on an $m \times \ell$ matrix and one SVD on a $n \times \ell$ matrix. While recently developed Communication-Avoiding QR (CAQR) algorithms [21] are optimal in terms of communication costs, standard techniques to compute an SVD are challenging for parallelization [18, 61]. Thus, the operations of RRR-UZVD can be organized to provide a low-rank approximation with the optimal communication cost.

3.3 Simulations

In this section, we evaluate the performance of RRR-UZVD. We illustrate through numerical examples that RRR-UZVD (i) is a rank revealer, and (ii) provides estimates of singular values, i.e., **Z**-*values*, that with remarkable fidelity track singular values of the matrix. We compare the performance of RRR-UZVD against those of the optimal SVD, QR with column pivoting (QRCP) and R-SVD. We next consider an image reconstruction problem in which a low-rank image of a differential gear of size 1280×804 is reconstructed using RRR-UZVD. Finally, we develop an algorithm to solve the robust PCA problem by making use of RRR-UZVD, and experimentally investigate the effectiveness of the proposed method on synthetic and real data.

3.3.1 Rank-Revealing Property and Singular Value Estimation

For the first example, we generate a noisy rank-k matrix $\mathbf{A} \in \mathbb{R}^{1000\times1000}$ as $\mathbf{A} = \mathbf{A_1} + \mathbf{A_2}$. $\mathbf{A}_1 = \mathbf{U\Sigma V}^T$, where $\mathbf{U}$ and $\mathbf{V}$ are orthonormal matrices, and $\mathbf{\Sigma}$ is diagonal containing the singular values σ_is whose entries decrease linearly from 1 to 10^{-9}, and $\sigma_{k+1} = ... = \sigma_{1000} = 0$. $\mathbf{A}_2$ is a Gaussian matrix normalized to have ℓ_2-norm gap $\times\ \sigma_k$. In MATLAB notation, we have `smax = 1; smin = 1e-9; s = linspace(smax,smin,n); s(k+1:n) = 0; G = randn(n); E = G/norm(G); A = orth(rand(n)) * diag(s) * orth(rand(n)) + gap * s(k) * E`. We set $k = 20$, $\ell = 2k$, and gap $= 0.15$.

For the second example a challenging matrix $\mathbf{A} \in \mathbb{R}^{1000\times1000}$ with multiple gaps in its singular value spectrum, the devil's stairs [80], is generated. The singular values of $\mathbf{A}$ are arranged akin to a descending staircase, where each step consists of $d = 10$ equal singular values. We set $q = 1$ for R-SVD and RRR-UZVD.

We compare the quality of singular values computed by RRR-UZVD (Algorithm 6) against that of the SVD, QRCP, and R-SVD. The results are shown in Figure 3.1. We make the following observations: (i) RRR-UZVD strongly reveals the gap between σ_{20} and σ_{21} in the first matrix, and estimates singular values with no loss of accuracy compared to the SVD, while QRCP fails to reveal the rank, and its approximations to the leading singular values significantly deviate from those of the SVD, and (ii) For the Devil's stairs matrix, RRR-UZVD reveals multiple gaps in its singular values and, further, perfectly tracks the singular values. RRR-UZVD provides excellent approximations to the singular values of the matrix, while QRCP fails in revealing the gaps, estimating and tracking the singular values of the matrix.

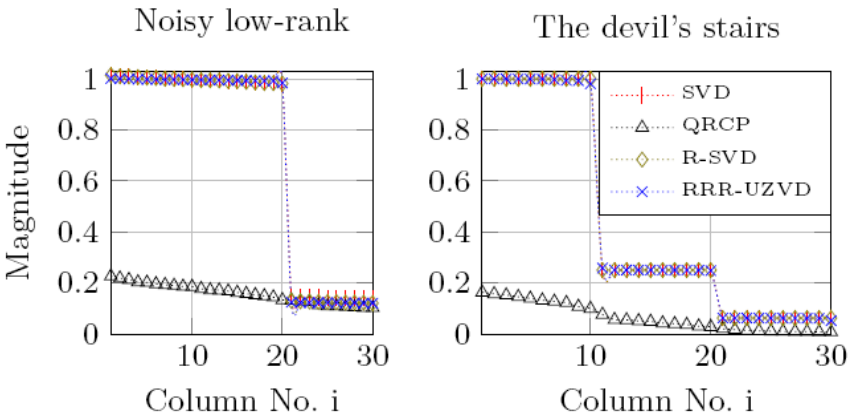


Figure 3.1: Comparison of singular values.

3.3.2 Image Reconstruction

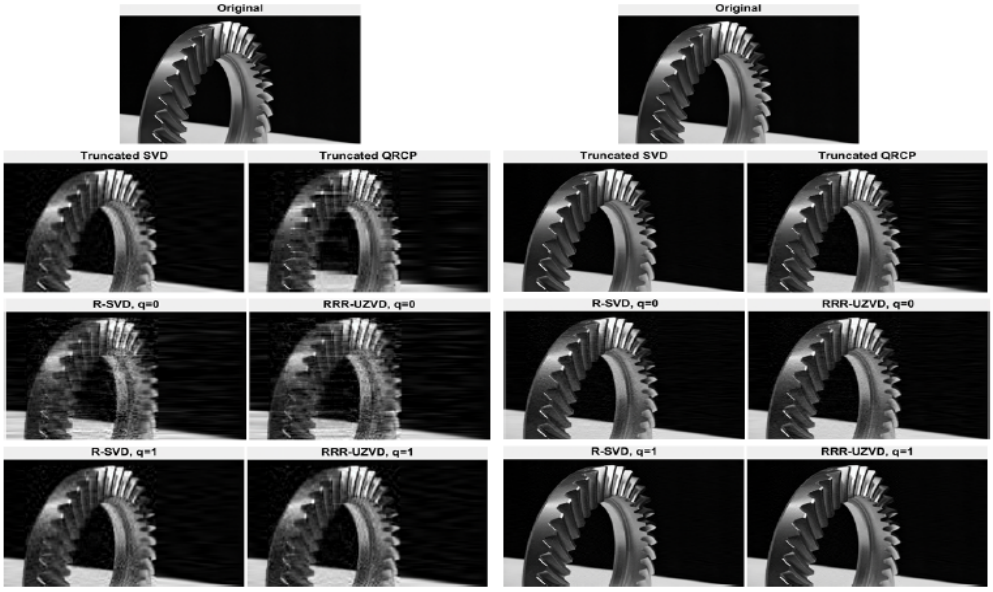
We assess the quality of the low-rank approximation produced by RRR-UZVD by reconstructing a gray-scale image of a differential gear of size 1280×804, taken from [26]. We compare the results with those of the truncated QRCP, R-SVD, and the truncated SVD by using (widely recommended) PROPACK package [57].

The results are shown in Figure 3.2; Figures 3.2a and 3.2b show the reconstructed images with $rank = 25$ and $rank = 85$, respectively, using the algorithms mentioned. Figure 3.3a displays the Frobenius-norm approximation error against the corresponding approximation rank, where the error is calculated as:

$$e_{\text{approx}} = \|\mathbf{A} - \hat{\mathbf{A}}_{\text{approx}}\|_F, \tag{3-14}$$

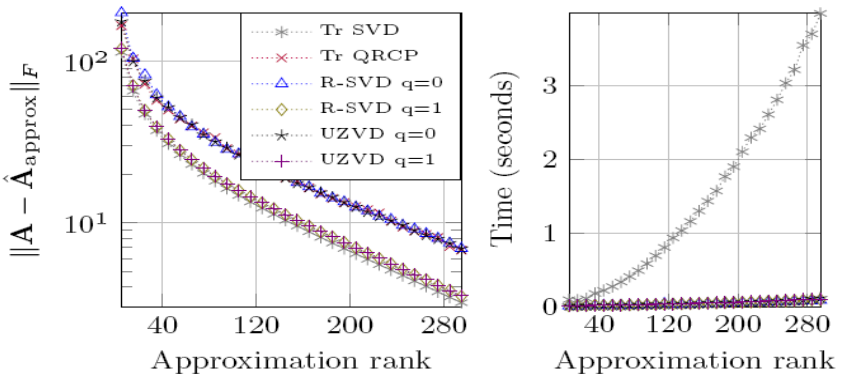
where $\hat{\mathbf{A}}_{\text{approx}}$ is the approximation computed by each algorithm. Figure 3.3b compares the runtime of RRR-UZVD, R-SVD and the truncated SVD against the corresponding approximation rank. We have discarded truncated QRCP because there is no optimized LAPACK function for QRCP with a specified rank.

In Figure 3.2a (rank-25 approximation), RRR-UZVD and R-SVD with $q = 0$ show the poorest reconstruction qualities. The truncated QRCP shows a better approximation, while RRR-UZVD and R-SVD with $q = 1$ produce approximations as good as the truncated SVD, outperforming the truncated QRCP. In Figure 3.2b (rank-85 approximation), with a careful scrutiny tiny artifacts appear in the reconstructed images by truncated QRCP as well as RRR-UZVD and R-SVD with $q = 0$, while reconstructed images by truncated SVD, RRR-UZVD and R-SVD with $q = 1$ are visually indistinguishable from the original.



(a) Reconstruction with $rank = 25$ (b) Reconstruction with $rank = 85$

Figure 3.2: Low-rank image reconstruction.



(a) Approximation error (b) Computational time

Figure 3.3: (a) Errors incurred by the algorithms considered in reconstructing the differential gear image. (b) Computational time in seconds for different algorithms.

Figure 3.3b illustrates how the execution time for truncated SVD substantially grows as the approximation rank increases. The results show that one step of the power method hardly adds to the execution time of more efficient RRR-UZVD. This shows that RRR-UZVD produces comparable results with truncated SVD at a much lower cost. However, we expect that since RRR-UZVD's operations can be performed with minimum communication costs (see subsection 3.2.2), on current and future advanced computers RRR-UZVD to be faster than the truncated SVD as well as R-SVD, where communication cost is a major bottleneck on the performance of an algorithm.

3.3.3 Robust PCA Using RRR-UZVD

We now apply RRR-UZVD to solve the robust PCA problem [9, 14, 93]; see Section 2.6 of Chapter 2 for a detailed description of robust PCA. We retain the original objective function proposed in [9,14,59,93], and apply RRR-UZVD as a surrogate to the SVD to solve the optimization problem (2-26). We adopt the continuation technique [59, 85], which increases μ in each iteration. The pseudocode of the proposed method, called `ALM-UZVD` hereafter, is given in Table 3.1.

Table 3.1: Pseudo-code of robust PCA solved by `ALM-UZVD`.

Input: Matrix $\mathbf{X}, \lambda, \mu_0, \bar{\mu}, \rho, \mathbf{Y}_0, \mathbf{S}_0, j = 0$;
Output: Low-rank plus sparse matrix

while the algorithm does not converge **do**
  Compute $\mathbf{L}_{j+1} = \mathcal{Z}_{\mu_j^{-1}}(\mathbf{X} - \mathbf{S}_j + \mu_j^{-1}\mathbf{Y}_j)$;
  Compute $\mathbf{S}_{j+1} = \mathcal{S}_{\lambda\mu_j^{-1}}(\mathbf{X} - \mathbf{L}_{j+1} + \mu_j^{-1}\mathbf{Y})$;
  Compute $\mathbf{Y}_{j+1} = \mathbf{Y}_j + \mu_j(\mathbf{X} - \mathbf{L}_{j+1} - \mathbf{S}_{j+1})$;
  Update $\mu_{j+1} = \max(\rho\mu_j, \bar{\mu})$;
end while
return $\mathbf{L}^*$ and $\mathbf{S}^*$

In Table 3.1, for a matrix $\mathbf{B}$ having a RRR-UZV decomposition described in Section 3.1, $\mathcal{Z}_\delta(\mathbf{B})$ refers to a UZV thresholding operator defined as:

$$\mathcal{Z}_\delta(\mathbf{B}) = \mathbf{U}(:, 1:r)\mathbf{Z}(1:r, :)\mathbf{V}^T, \tag{3-15}$$

where r is the number of diagonals of $\mathbf{Z}$ greater than δ, $\mathcal{S}_\delta(x) = \text{sgn}(x)\max(|x| - \delta, 0)$ is a shrinkage operator and λ, μ_0, $\bar{\mu}$, ρ, $\mathbf{Y}_0$, and $\mathbf{S}_0$ are initial values. We compare the results of `ALM-UZVD` with those of `InexactALM` [59] (See Section 2.8 of Chapter 2 for more details on `InexactALM`).

3.3.3.1 Synthetic Data Recovery

We construct a rank-k matrix $\mathbf{X} = \mathbf{L} + \mathbf{S}$ as a sum of a low-rank matrix $\mathbf{L} \in \mathbb{R}^{n\times n}$ and a sparse error matrix $\mathbf{S} \in \mathbb{R}^{n\times n}$. $\mathbf{L}$ is generated as $\mathbf{L} = \mathbf{U}\mathbf{V}^T$, where $\mathbf{U}$, $\mathbf{V} \in \mathbb{R}^{n\times k}$ are standard normal matrices, and $\mathbf{S}$ has s non-zero entries independently drawn from the set {-100, 100}. We consider $k = \text{rank}(\mathbf{L}) = 0.05 \times n$ and $s = \|\mathbf{S}\|_0 = 0.05 \times n^2$, where $\|\cdot\|_0$ denotes the ℓ_0-norm.

We apply the `ALM-UZVD` and `InexactALM` algorithms to $\mathbf{X}$ to recover $\mathbf{L}$ and $\mathbf{S}$. The numerical results are summarized in Table 3.2. In our experiments, we adopt the initial values suggested in [59], and algorithms are terminated when $\|\mathbf{X} - \mathbf{L}^{sol} - \mathbf{S}^{sol}\|_F < 10^{-4}\|\mathbf{X}\|_F$ is satisfied, where $(\mathbf{L}^{sol}, \mathbf{S}^{sol})$ is the pair of solution of either algorithm. In Table 3.2, *Time* refers to the computational time in seconds, *Iter.* refers to the number of iterations, and $\zeta = \|\mathbf{X} - \mathbf{L}^{sol} - \mathbf{S}^{sol}\|_F/\|\mathbf{X}\|_F$ refers to relative error. RRR-UZVD requires a prespecified rank ℓ to perform the factorization. We thus set $\ell = 2k$, as a random start, and $q = 2$. Judging from the results in Table 3.2, we make several observations on `ALM-UZVD`: (i) it successfully detects the exact numerical rank k of the input matrix in all cases, (ii) it provides the exact optimal solution, while it requires one more iteration compared to `InexactALM`, and (iii) outperforms `InexactALM` in terms of runtime, with speedups of up to 5×.

Table 3.2: Numerical results for synthetic matrix recovery.

n	$r(\mathbf{L})$	$\|\mathbf{S}\|_0$	Methods	$r(\hat{\mathbf{L}})$	$\|\hat{\mathbf{S}}\|_0$	Time	Iter.	ξ
1000	50	5e4	`InexactALM`	50	5e4	2.5	9	3.1e-5
			`ALM-UZVD`	50	5e4	0.6	10	4.2e-5
2000	100	2e5	`InexactALM`	100	2e5	17.6	9	4.9e-5
			`ALM-UZVD`	100	2e5	4.4	10	4.1e-5
3000	150	45e4	`InexactALM`	150	45e4	52.9	9	5.2e-5
			`ALM-UZVD`	150	45e4	10.5	10	5.3e-5

3.3.3.2 Background Modeling in Surveillance Video

In this experiment, we apply `ALM-UZVD` to a surveillance video introduced in [58]. The video consists of 200 grayscale frames of size 256 × 320, taken in a

(a) Shopping mall (b) Yale B02 face

Figure 3.4: (a) Images in columns 1, 2, and 3 are frames of the video, recovered backgrounds $\mathbf{L}^*$ and foregrounds $\mathbf{S}^*$, respectively, by `ALM-UZVD`. (b) Images in column 1 are cropped images of a face under varying illuminations. Images in column 2 and 3 are recovered images by `ALM-UZVD` and errors corresponding to the shadows and specularities, respectively.

shopping mall. We form a matrix $\mathbf{X} \in \mathbb{R}^{81920\times 200}$ by stacking individual frames as its columns.

The RRR-UZVD, used in `ALM-UZVD`, requires a prespecified rank ℓ which we determine by using the following bound that relates the rank k of any matrix $\mathbf{B}$ with the nuclear and Frobenius norms [35]:

$$\|\mathbf{B}\|_* \leq \sqrt{k}\|\mathbf{B}\|_F. \tag{3-16}$$

We set $\ell = k + p$, where k is the minimum value satisfying (3-16), and $p = 2$ is an oversampling parameter. Again, we set $q = 2$ for RRR-UZVD.

Some frames of the surveillance video with recovered foregrounds and backgrounds are displayed in Figure 3.4a. As seen, `ALM-UZVD` successfully recovers the low-rank and sparse components of the video. Table 3.3 presents the numerical results, which shows `ALM-UZVD` outperform `InexactALM` in terms of runtime.

3.3.3.3 Shadow and Specularity Removal from Face Images

In this experiment, we use face images from the Yale B face database [34]. Each image has the size 192×168 with a total of 64 different illuminations. The images are stacked as columns of a matrix $\mathbf{X} \in \mathbb{R}^{32256\times 64}$. The recovered images are displayed in Figure 3.4b. We observe that the shadows and specularities have been effectively extracted in the sparse components by `ALM-UZVD`. Table 3.3 summarizes the numerical results.

We conclude that `ALM-UZVD` successfully recovers the face images under

Table 3.3: Comparison of the `InexactALM` and `ALM-UZVD` methods for real-time data recovery.

Dataset	`InexactALM`			`ALM-UZVD`		
	Time	Iter.	ξ	Time	Iter.	ξ
Shopping mall 81920×200	47.8	23	7.1e-5	15.8	23	8.1e-5
Yale B02 32256×64	3.5	21	7.5e-5	2.0	21	9.1e-5

different illuminations from the dataset studied nearly two times faster than `InexactALM`.

4
Subspace-Orbit Randomized Singular Value Decomposition

This chapter introduces a new matrix decomposition approach termed Subspace-Orbit Randomized Singular Value Decomposition (SOR-SVD), which makes use of random sampling techniques to give a low-rank approximation to an input matrix.

The SOR-SVD algorithm for a matrix $\mathbf{A} \in \mathbb{R}^{m\times n}$ constructs a rank-k approximation in $O(mnk)$ flops by making only a few passes through $\mathbf{A}$. Built on RRR-UZVD [53], the difference between the two algorithms is that in SOR-SVD an SVD is applied on the compressed matrix rather than column permutation technique. This allows further exploration of the properties of SOR-SVD. Theoretical lower bounds on the singular values and upper bounds on the error of the low-rank approximation for SOR-SVD are provided. We experimentally show that the low-rank approximation error bounds provided are empirically sharp for one class of matrices considered. To demonstrate the effectiveness of SOR-SVD, we conduct experiments on synthetic data as well as real data in computer vision applications of background/foreground separation in surveillance video and shadow and specularity removal from face images.

4.1
Proposed SOR-SVD Algorithm

The SOR-SVD computes a *fixed-rank* approximation of a given matrix. Given a matrix $\mathbf{A} \in \mathbb{R}^{m\times n}$, where $m \geq n$, with numerical rank k and an integer $k \leq \ell < n$, SOR-SVD is computed as follows: using a random number generator, we form a real matrix $\mathbf{\Omega} \in \mathbb{R}^{n\times \ell}$ with entries being independent, identically distributed (i.i.d.) Gaussian random variables of zero mean and unit variance. We then compute the matrix product:

$$\mathbf{T}_1 = \mathbf{A}\mathbf{\Omega}, \tag{4-1}$$

where $\mathbf{T}_1 \in \mathbb{R}^{m\times \ell}$ is formed by linear combinations of columns of $\mathbf{A}$ by the random Gaussian matrix. $\mathbf{T}_1$ is nothing but a projection onto the subspace spanned by columns of $\mathbf{A}$. Having $\mathbf{T}_1$, we form the matrix $\mathbf{T}_2 \in \mathbb{R}^{n\times \ell}$:

$$\mathbf{T}_2 = \mathbf{A}^T\mathbf{T}_1, \tag{4-2}$$

where $\mathbf{T}_2$ is constructed by linear combinations of rows of $\mathbf{A}$ by $\mathbf{T}_1$. $\mathbf{T}_2$ is nothing but a projection onto the subspace spanned by rows of $\mathbf{A}$. Using the QR decomposition algorithm, we factor $\mathbf{T}_1$ and $\mathbf{T}_2$ such that:

$$\mathbf{T}_1 = \mathbf{Q}_1\mathbf{R}_1 \quad \text{and} \quad \mathbf{T}_2 = \mathbf{Q}_2\mathbf{R}_2, \tag{4-3}$$

where $\mathbf{Q}_1$ and $\mathbf{Q}_2$ are approximate bases for $\mathcal{R}(\mathbf{A})$ and $\mathcal{R}(\mathbf{A}^T)$, respectively. We now form a matrix $\mathbf{M} \in \mathbb{R}^{\ell\times\ell}$ by compression of $\mathbf{A}$ through left and right multiplications by orthonormal bases:

$$\mathbf{M} = \mathbf{Q}_1^T\mathbf{A}\mathbf{Q}_2, \tag{4-4}$$

We then compute the rank-k truncated SVD of $\mathbf{M}$:

$$\mathbf{M}_k = \widetilde{\mathbf{U}}_k\widetilde{\mathbf{\Sigma}}_k\widetilde{\mathbf{V}}_k, \tag{4-5}$$

Finally, we form the SOR-SVD-based low-rank approximation of $\mathbf{A}$:

$$\hat{\mathbf{A}}_{\text{SOR}} = (\mathbf{Q}_1\widetilde{\mathbf{U}}_k)\widetilde{\mathbf{\Sigma}}_k(\mathbf{Q}_2\widetilde{\mathbf{V}}_k)^T, \tag{4-6}$$

where $\mathbf{Q}_1\widetilde{\mathbf{U}}_k \in \mathbb{R}^{m\times k}$ and $\mathbf{Q}_2\widetilde{\mathbf{V}}_k \in \mathbb{R}^{n\times k}$ are approximations to the k leading left and right singular vectors of $\mathbf{A}$, respectively, and $\widetilde{\mathbf{\Sigma}}_k$ contains an approximation to the k leading singular values. The algorithm is presented in Algorithm 7.

Algorithm 7 Subspace-Orbit Randomized SVD (SOR-SVD)

Input: Matrix $\mathbf{A} \in \mathbb{R}^{m\times n}$, integers k and ℓ.
Output: A rank-k approximation.

1: Draw a standard Gaussian matrix $\mathbf{\Omega} \in \mathbb{R}^{n\times\ell}$;
2: Compute $\mathbf{T}_1 = \mathbf{A}\mathbf{\Omega}$;
3: Compute $\mathbf{T}_2 = \mathbf{A}^T\mathbf{T}_1$;
4: Compute QR decompositions $\mathbf{T}_1 = \mathbf{Q}_1\mathbf{R}_1$, $\mathbf{T}_2 = \mathbf{Q}_2\mathbf{R}_2$;
5: Compute $\mathbf{M} = \mathbf{Q}_1^T\mathbf{A}\mathbf{Q}_2$;
6: Compute the rank-k truncated SVD $\mathbf{M}_k = \widetilde{\mathbf{U}}_k\widetilde{\mathbf{\Sigma}}_k\widetilde{\mathbf{V}}_k$;
7: Form the low-rank approximation of $\mathbf{A}$:$\hat{\mathbf{A}}_{\text{SOR}} = (\mathbf{Q}_1\widetilde{\mathbf{U}}_k)\widetilde{\mathbf{\Sigma}}_k(\mathbf{Q}_2\widetilde{\mathbf{V}}_k)^T$.

The SOR-SVD requires three passes through data, for matrices stored-out-of-core, but it can be modified to revisit the data only once. To this end, the compressed matrix $\mathbf{M}$ (4-4) can be computed by making use of currently available matrices as follows: both sides of the currently unknown $\mathbf{M}$ are postmultiplied by $\mathbf{Q}_2^T\mathbf{\Omega}$:

$$\mathbf{M}\mathbf{Q}_2^T\mathbf{\Omega} = \mathbf{Q}_1^T\mathbf{A}\mathbf{Q}_2\mathbf{Q}_2^T\mathbf{\Omega}. \tag{4-7}$$

Having defined $\mathbf{A} \approx \mathbf{A}\mathbf{Q}_2\mathbf{Q}_2^T$ and $\mathbf{T}_1 = \mathbf{A}\mathbf{\Omega}$, an approximation to $\mathbf{M}$ is obtained by:

$$\mathbf{M}_{\text{approx}} = \mathbf{Q}_1^T\mathbf{T}_1(\mathbf{Q}_2^T\mathbf{\Omega})^\dagger. \tag{4-8}$$

Remark 4.1 *In the SOR-SVD algorithm, projecting* $\mathbf{A}$ *onto a subspace spanned by its rows using a matrix containing random linear combinations of its columns, i.e., equation* (4-2)*, significantly improves the quality of the approximate basis* $\mathbf{Q}_2$*, compared to that of the TSR-SVD (Algorithm 3). This results in (i) an accurate approximation* $\mathbf{M}_{approx}$ *to* $\mathbf{M}$*, and (ii) tighter bounds for the singular values.*

The two key differences between our proposed SOR-SVD and TSR-SVD are (i) to use a sketch of the input matrix in order to project it onto its row space, rather than using a random matrix, and (ii) to apply truncated SVD on the reduced-size matrix.

SOR-SVD may be sufficiently accurate for matrices whose singular values display some decay, however in applications where the data matrix has a slowly decaying singular values, it may produce singular vectors and singular values that significantly deviate from those of the optimal SVD. Thus, we incorporate q steps of a power iteration [39, 71] to improve the accuracy of the algorithm in these circumstances. Using power iterations, to obtain the approximate bases, the algorithm consecutively projects the input matrix onto is subspaces by making use of compressed versions of the input matrix. This substantially improves the quality of left and right approximate bases. The resulting algorithm is described in Algorithm 8. Note that in implementing Algorithm 8, a non-updated $\mathbf{T}_2$ must be used to form $\mathbf{M}_{\text{approx}}$.

Algorithm 8 SOR-SVD with Power Method

Input: Matrix $\mathbf{A} \in \mathbb{R}^{m\times n}$, integers k, ℓ and q.
Output: A rank-k approximation.

1: Draw a standard Gaussian matrix $\mathbf{T}_2 \in \mathbb{R}^{n\times \ell}$;
2: **for** $i = 1$: $q+1$ **do**
3: Compute $\mathbf{T}_1 = \mathbf{A}\mathbf{T}_2$;
4: Compute $\mathbf{T}_2 = \mathbf{A}^T\mathbf{T}_1$;
5: **end for**
6: Compute QR decompositions $\mathbf{T}_1 = \mathbf{Q}_1\mathbf{R}_1$, $\mathbf{T}_2 = \mathbf{Q}_2\mathbf{R}_2$;
7: Compute $\mathbf{M} = \mathbf{Q}_1^T\mathbf{A}\mathbf{Q}_2$ or $\mathbf{M}_{\text{approx}} = \mathbf{Q}_1^T\mathbf{T}_1(\mathbf{Q}_2^T\mathbf{T}_2)^\dagger$;
8: Compute the rank-k truncated SVD $\mathbf{M}_k = \widetilde{\mathbf{U}}_k\widetilde{\mathbf{\Sigma}}_k\widetilde{\mathbf{V}}_k$ or $\mathbf{M}_{\text{approx-k}} = \widetilde{\mathbf{U}}_k\widetilde{\mathbf{\Sigma}}_k\widetilde{\mathbf{V}}_k$;
9: Form the low-rank approximation of $\mathbf{A}$: $\hat{\mathbf{A}}_{\text{SOR}} = (\mathbf{Q}_1\widetilde{\mathbf{U}}_k)\widetilde{\mathbf{\Sigma}}_k(\mathbf{Q}_2\widetilde{\mathbf{V}}_k)^T$.

4.2 Analysis of SOR-SVD

In this section, we provide a detailed analysis of the performance of the SOR-SVD algorithms, the basic version in Algorithm 7 as well as the one in

Algorithm 8. In particular, we develop lower bounds on singular values, as well as upper bounds on the rank-k approximation error in terms of the spectral and Frobenius norms.

The TSR-SVD (Algorithm 3) results in the approximation $\mathbf{A} \approx \mathbf{Q}_1\mathbf{M}\mathbf{Q}_2^T$, where $\mathbf{M}$ is defined in (4-4). However, the SOR-SVD procedure results in $\mathbf{A} \approx \mathbf{Q}_1\mathbf{M}_k\mathbf{Q}_2^T$, where $\mathbf{M}_k$ is the rank-k truncated SVD of $\mathbf{M}$. The following theorem is a generalization of Theorem 2.1 for SOR-SVD.

Theorem 4.2 *Let $\mathbf{Q}_1 \in \mathbb{R}^{m\times\ell}$ and $\mathbf{Q}_2 \in \mathbb{R}^{n\times\ell}$ be matrices with orthonormal columns, and $1 \le k \le \ell$. Let $\mathbf{M}_k$ be the rank-k truncated SVD of $\mathbf{Q}_1^T\mathbf{A}\mathbf{Q}_2$. Then $\mathbf{M}_k$ is an optimal solution to the following optimization problem:*

$$\underset{\mathbf{M}\in\mathbb{R}^{\ell\times\ell}, rank(\mathbf{M})\le k}{minimize} \|\mathbf{A} - \mathbf{Q}_1\mathbf{M}\mathbf{Q}_2^T\|_F = \|\mathbf{A} - \mathbf{Q}_1\mathbf{M}_k\mathbf{Q}_2^T\|_F, \tag{4-9}$$

and

$$\|\mathbf{A}-\mathbf{Q}_1\mathbf{M}_k\mathbf{Q}_2^T\|_F \le \|\mathbf{A}_0\|_F + \|\mathbf{A}_k - \mathbf{Q}_1\mathbf{Q}_1^T\mathbf{A}_k\|_F + \|\mathbf{A}_k - \mathbf{A}_k\mathbf{Q}_2\mathbf{Q}_2^T\|_F, \tag{4-10}$$

and we also have

$$\|\mathbf{A} - \mathbf{Q}_1\mathbf{M}_k\mathbf{Q}_2^T\|_2 \le \|\mathbf{A}_0\|_2 + \|\mathbf{A}_k - \mathbf{Q}_1\mathbf{Q}_1^T\mathbf{A}_k\|_F + \|\mathbf{A}_k - \mathbf{A}_k\mathbf{Q}_2\mathbf{Q}_2^T\|_F. \tag{4-11}$$

Proof. The proof is given in Appendix 8.2.1.

4.2.1 Deterministic Error Bounds

In this section, we make use of techniques from linear algebra to give generic error bounds which depend on the interaction between the standard Gaussian matrix $\mathbf{\Omega}$ and the right singular vectors of the data matrix $\mathbf{A}$.

To derive lower bounds on approximated singular values, we begin by stating two key results that are used in the analysis later on.

Theorem 4.3 *(Thompson [83]) Let the matrix $\mathbf{A}$ have singular values as defined in (2-1), and $\mathbf{M} \in \mathbb{R}^{\ell\times\ell}$ be a submatrix of $\mathbf{A}$. Then for $j = 1, ..., \ell$, we have*

$$\sigma_j \ge \sigma_j(\mathbf{M}). \tag{4-12}$$

The relation in (4-12) can be easily proven by allowing $\mathbf{M}$ to be $\mathbf{M} = \mathbf{Q}_1^T\mathbf{A}\mathbf{Q}_2$, where $\mathbf{Q}_1 \in \mathbb{R}^{m\times\ell}$ and $\mathbf{Q}_2 \in \mathbb{R}^{n\times\ell}$ are orthonormal matrices.

Remark 4.4 *Since the matrices $\mathbf{M}$ and $\mathbf{M}_k$ have the same singular values σ_j, for $j = 1, ..., k$, and moreover, singular values of $\mathbf{Q}_1^T\mathbf{A}\mathbf{Q}_2$ and $\mathbf{Q}_1\mathbf{Q}_1^T\mathbf{A}\mathbf{Q}_2\mathbf{Q}_2^T$ coincide [83], we have*

$$\sigma_j \geq \sigma_j(\mathbf{Q}_1\mathbf{M}_k\mathbf{Q}_2^T) = \sigma_j(\mathbf{Q}_1\mathbf{Q}_1^T\mathbf{A}\mathbf{Q}_2\mathbf{Q}_2^T). \tag{4-13}$$

Lemma 4.5 *(Gu [36]). Let* $\mathbf{B} \in \mathbb{R}^{m\times n}$ *be any matrix,* $\mathbf{\Omega} \in \mathbb{R}^{n\times \ell}$ *be full column rank, and* $\mathbf{X} \in \mathbb{R}^{\ell\times\ell}$ *be a non-singular matrix. Let* $\mathbf{B\Omega} = \mathbf{Q}_b\mathbf{R}_b$ *and* $\mathbf{B\Omega X} = \mathbf{Q}_x\mathbf{R}_x$ *be QR decompositions of the matrix products. Then*

$$\mathbf{Q}_b\mathbf{Q}_b^T = \mathbf{Q}_x\mathbf{Q}_x^T. \tag{4-14}$$

For the matrix $\mathbf{T}_2$, equation (4-2), and by the SVD of $\mathbf{A}$ given in (2-1), we have

$$\mathbf{T}_2 = \mathbf{A}^T\mathbf{T}_1 = \mathbf{A}^T\mathbf{A\Omega} = \mathbf{V\Sigma}^2\mathbf{V}^T\mathbf{\Omega}. \tag{4-15}$$

By partitioning $\mathbf{\Sigma}^2$, we have

$$\mathbf{T}_2 = \mathbf{V}\begin{bmatrix}\mathbf{\Sigma}_1^2 & \mathbf{0} & \mathbf{0}\\ \mathbf{0} & \mathbf{\Sigma}_2^2 & \mathbf{0}\\ \mathbf{0} & \mathbf{0} & \mathbf{\Sigma}_3^2\end{bmatrix}\begin{bmatrix}\mathbf{V}_1^T\mathbf{\Omega}\\ \\ \mathbf{V}_2^T\mathbf{\Omega}\end{bmatrix} = \mathbf{Q}_2\mathbf{R}_2, \tag{4-16}$$

where $\mathbf{\Sigma}_1 \in \mathbb{R}^{k\times k}$, $\mathbf{\Sigma}_2 \in \mathbb{R}^{\ell-p-k\times\ell-p-k}$, $\mathbf{\Sigma}_3 \in \mathbb{R}^{n-\ell+p\times n-\ell+p}$. We define $\mathbf{\Omega}_1 \in \mathbb{R}^{\ell-p\times\ell}$ and $\mathbf{\Omega}_2 \in \mathbb{R}^{n-\ell+p\times\ell}$ as follows:

$$\mathbf{\Omega}_1 \triangleq \mathbf{V}_1^T\mathbf{\Omega}, \quad \text{and} \quad \mathbf{\Omega}_2 \triangleq \mathbf{V}_2^T\mathbf{\Omega}. \tag{4-17}$$

We assume that $\mathbf{\Omega}_1$ is full row rank and its pseudo-inverse satisfies

$$\mathbf{\Omega}_1\mathbf{\Omega}_1^\dagger = \mathbf{I}. \tag{4-18}$$

To understand the behavior of singular values and the low-rank approximation, we choose a matrix $\mathbf{X} \in \mathbb{R}^{\ell\times\ell}$, which orients the first k columns of $\mathbf{T}_2\mathbf{X}$ in the directions of the k leading singular vectors in $\mathbf{V}$. Thus we choose

$$\mathbf{X} = \left[\mathbf{\Omega}_1^\dagger\begin{pmatrix}\mathbf{\Sigma}_1^2 & \mathbf{0}\\ \mathbf{0} & \mathbf{\Sigma}_2^2\end{pmatrix}^{-1}, \tilde{\mathbf{X}}\right], \tag{4-19}$$

where the $\tilde{\mathbf{X}} \in \mathbb{R}^{\ell\times p}$ is chosen such that $\mathbf{X} \in \mathbb{R}^{\ell\times\ell}$ is non-singular, and $\mathbf{\Omega}_1\tilde{\mathbf{X}} = \mathbf{0}$. Now we write

$$\mathbf{T}_2\mathbf{X} = \mathbf{A}^T\mathbf{A\Omega X} = \mathbf{V}\begin{bmatrix}\begin{pmatrix}\mathbf{\Sigma}_1^2 & \mathbf{0}\\ \mathbf{0} & \mathbf{\Sigma}_2^2\end{pmatrix}\mathbf{\Omega}_1\\ \mathbf{\Sigma}_3^2\mathbf{\Omega}_2\end{bmatrix}\mathbf{X} = \mathbf{V}\begin{bmatrix}\mathbf{I} & \mathbf{0} & \mathbf{0}\\ \mathbf{0} & \mathbf{I} & \mathbf{0}\\ \mathbf{W}_1 & \mathbf{W}_2 & \mathbf{W}_3\end{bmatrix}, \tag{4-20}$$

where $\mathbf{W}_1 = \mathbf{\Sigma}_3^2\mathbf{\Omega}_2\mathbf{\Omega}_1^\dagger\mathbf{\Sigma}_1^{-2} \in \mathbb{R}^{n-\ell+p\times k}$, $\mathbf{W}_2 = \mathbf{\Sigma}_3^2\mathbf{\Omega}_2\mathbf{\Omega}_1^\dagger\mathbf{\Sigma}_2^{-2} \in \mathbb{R}^{n-\ell+p\times\ell-p-k}$, and $\mathbf{W}_3 = \mathbf{\Sigma}_3^2\mathbf{\Omega}_2\tilde{\mathbf{X}} \in \mathbb{R}^{n-\ell+p\times p}$.

Now by a QR decomposition of equation (4-20), with partitioned matrices, we have

$$\begin{aligned}\mathbf{V}\begin{bmatrix}\mathbf{I} & \mathbf{0} & \mathbf{0}\\ \mathbf{0} & \mathbf{I} & \mathbf{0}\\ \mathbf{W}_1 & \mathbf{W}_2 & \mathbf{W}_3\end{bmatrix} &= \tilde{\mathbf{Q}}\tilde{\mathbf{R}} = [\tilde{\mathbf{Q}}_1 \quad \tilde{\mathbf{Q}}_2 \quad \tilde{\mathbf{Q}}_3]\begin{bmatrix}\tilde{\mathbf{R}}_{11} & \tilde{\mathbf{R}}_{12} & \tilde{\mathbf{R}}_{13}\\ \mathbf{0} & \tilde{\mathbf{R}}_{22} & \tilde{\mathbf{R}}_{23}\\ \mathbf{0} & \mathbf{0} & \tilde{\mathbf{R}}_{33}\end{bmatrix}\\ &= \begin{bmatrix}(\tilde{\mathbf{Q}}_1\tilde{\mathbf{R}}_{11})^T\\ (\tilde{\mathbf{Q}}_1\tilde{\mathbf{R}}_{12}+\tilde{\mathbf{Q}}_2\tilde{\mathbf{R}}_{22})^T\\ (\tilde{\mathbf{Q}}_1\tilde{\mathbf{R}}_{13}+\tilde{\mathbf{Q}}_2\tilde{\mathbf{R}}_{23}+\tilde{\mathbf{Q}}_3\tilde{\mathbf{R}}_{33})^T\end{bmatrix}^T.\end{aligned} \tag{4-21}$$

We use this representation to develop lower bounds on singular values and upper bounds on the error of the rank-k approximation of SOR-SVD.

Lemma 4.6 *Let $\mathbf{W}_1$ be defined as in* (4-20)*, and $\mathbf{\Omega}_1$ be full row rank. Then for $\hat{\mathbf{A}}_{SOR}$ defined in Algorithms 7 and 8, we have*

$$\sigma_k(\hat{\mathbf{A}}_{SOR}) \geq \frac{\sigma_k}{\sqrt{1+\|\mathbf{W}_1\|_2^2}}. \tag{4-22}$$

Proof. The proof is given in Appendix 8.2.2.

We now present the result.

Theorem 4.7 *Suppose that the matrix $\mathbf{A}$ has an SVD defined in* (2-1)*, $2 \leq k+p \leq \ell$, and $\hat{\mathbf{A}}_{SOR}$ is computed through the basic form of SOR-SVD (Algorithm 7). Assume furthermore that $\mathbf{\Omega}_1$ is full row rank, then for $j = 1, ..., k$, we have*

$$\sigma_j \geq \sigma_j(\hat{\mathbf{A}}_{SOR}) \geq \frac{\sigma_j}{\sqrt{1+\|\mathbf{\Omega}_2\|_2^2\|\mathbf{\Omega}_1{}^{\dagger}\|_2^2\left(\frac{\sigma_{\ell-p+1}}{\sigma_j}\right)^4}}, \tag{4-23}$$

and when the power method is used (Algorithm 8), we have

$$\sigma_j \geq \sigma_j(\hat{\mathbf{A}}_{SOR}) \geq \frac{\sigma_j}{\sqrt{1+\|\mathbf{\Omega}_2\|_2^2\|\mathbf{\Omega}_1{}^{\dagger}\|_2^2\left(\frac{\sigma_{\ell-p+1}}{\sigma_j}\right)^{4q+4}}}. \tag{4-24}$$

Proof. The proof is given in Appendix 8.2.3.

To derive upper bounds on the error of the low-rank approximation with respect to the spectral and Frobenius norms provided equations (4-10), (4-11), it suffices to bound the right-hand sides of the equations, i.e., $\|\mathbf{A}_k - \mathbf{Q}_1\mathbf{Q}_1^T\mathbf{A}_k\|_F$ and $\|\mathbf{A}_k - \mathbf{A}_k\mathbf{Q}_2\mathbf{Q}_2^T\|_F$. To this end, we begin by stating a proposition from [39].

Proposition 4.8 *(Halko et al. [39]) Suppose that for given matrices $\mathbf{N}_1$ and $\mathbf{N}_2$, $\mathcal{R}(\mathbf{N}_1) \subset \mathcal{R}(\mathbf{N}_2)$. Then for any matrix $\mathbf{A}$, it holds that*

$$\begin{aligned}&\|\mathbf{P}_{\mathbf{N}_1}\mathbf{A}\|_2 \leq \|\mathbf{P}_{\mathbf{N}_2}\mathbf{A}\|_2,\\ &\|(\mathbf{I}-\mathbf{P}_{\mathbf{N}_2})\mathbf{A}\|_2 \leq \|(\mathbf{I}-\mathbf{P}_{\mathbf{N}_1})\mathbf{A}\|_2,\end{aligned} \tag{4-25}$$

where $\mathbf{P}_{\mathbf{N}_1}$ *and* $\mathbf{P}_{\mathbf{N}_2}$ *are orthogonal projections onto* $\mathbf{N}_1$ *and* $\mathbf{N}_2$*, respectively.*

By combining Lemma 4.5 and Proposition 4.8, it follows that

$$\|\mathbf{A}_k(\mathbf{I}-\mathbf{Q}_2\mathbf{Q}_2^T)\|_F = \|\mathbf{A}_k(\mathbf{I}-\tilde{\mathbf{Q}}\tilde{\mathbf{Q}}^T)\|_F \leq \|\mathbf{A}_k(\mathbf{I}-\tilde{\mathbf{Q}}_1\tilde{\mathbf{Q}}_1^T)\|_F. \tag{4-26}$$

By the definition of $\tilde{\mathbf{Q}}_1$, equation (8-25), it follows

$$\mathbf{I}-\tilde{\mathbf{Q}}_1\tilde{\mathbf{Q}}_1^T = \mathbf{V}\begin{bmatrix} \mathbf{I}-\tilde{\mathbf{W}}^{-1} & \mathbf{0} & -\tilde{\mathbf{W}}^{-1}\mathbf{W}_1^T \\ \mathbf{0} & -\mathbf{I} & \mathbf{0} \\ -\mathbf{W}_1\tilde{\mathbf{W}}^{-1} & \mathbf{0} & \mathbf{I}-\mathbf{W}_1\tilde{\mathbf{W}}^{-1}\mathbf{W}_1^T \end{bmatrix}\mathbf{V}^T, \tag{4-27}$$

where $\mathbf{W}_1$ is defined in (4-20), and $\tilde{\mathbf{W}}^{-1}$ is defined as follows:

$$\tilde{\mathbf{R}}_{11}^{-1}\tilde{\mathbf{R}}_{11}^{-T} = (\tilde{\mathbf{R}}_{11}^T\tilde{\mathbf{R}}_{11})^{-1} = (1+\mathbf{W}_1^T\mathbf{W}_1)^{-1} = \tilde{\mathbf{W}}^{-1}.$$

We then have

$$\begin{aligned} \mathbf{A}_k(\mathbf{I}-\tilde{\mathbf{Q}}_1\tilde{\mathbf{Q}}_1^T) &= \mathbf{U}[\mathbf{\Sigma}_1 \quad \mathbf{0} \quad \mathbf{0}]\mathbf{V}^T(\mathbf{I}-\tilde{\mathbf{Q}}_1\tilde{\mathbf{Q}}_1^T) \\ &= \underbrace{\mathbf{U}[\mathbf{\Sigma}_1(\mathbf{I}-\tilde{\mathbf{W}}^{-1}) \quad \mathbf{0} \quad -\mathbf{\Sigma}_1\tilde{\mathbf{W}}^{-1}\mathbf{W}_1^T]\mathbf{V}^T}_{\mathbf{N}}. \end{aligned} \tag{4-28}$$

By the definition of the Frobenius norm, it follows that

$$\begin{aligned} \|\mathbf{A}_k(\mathbf{I}-\tilde{\mathbf{Q}}_1\tilde{\mathbf{Q}}_1^T)\|_F =& \sqrt{\operatorname{trace}(\mathbf{N}\mathbf{N}^T)} = \sqrt{\operatorname{trace}(\mathbf{\Sigma}_1\mathbf{W}_1^T(\mathbf{I}+\mathbf{W}_1\mathbf{W}_1^T)^{-1}\mathbf{W}_1\mathbf{\Sigma}_1^T)} \\ \leq& \sqrt{\operatorname{trace}([\|\mathbf{W}_1\mathbf{\Sigma}_1\|_2^2\mathbf{I}+\mathbf{\Sigma}_1^{-2}]^{-1})} \\ =& \|\mathbf{W}_1\mathbf{\Sigma}_1\|_2\sqrt{\operatorname{trace}(\mathbf{\Sigma}_1[\|\mathbf{W}_1\mathbf{\Sigma}_1\|_2^2\mathbf{I}+\mathbf{\Sigma}_1^{-2}]^{-1})\mathbf{\Sigma}_1^T)} \\ \leq& \frac{\sqrt{k}\|\mathbf{W}_1\mathbf{\Sigma}_1\|_2\sigma_1}{\sqrt{{\sigma_1}^2+\|\mathbf{W}_1\mathbf{\Sigma}_1\|_2^2}}. \end{aligned} \tag{4-29}$$

Plugging this into (4-26), we have

$$\|\mathbf{A}_k(\mathbf{I}-\mathbf{Q}_2\mathbf{Q}_2^T)\|_F \leq \frac{\sqrt{k}\|\mathbf{W}_1\mathbf{\Sigma}_1\|_2\sigma_1}{\sqrt{{\sigma_1}^2+\|\mathbf{W}_1\mathbf{\Sigma}_1\|_2^2}}. \tag{4-30}$$

To bound $\|\mathbf{A}_k-\mathbf{Q}_1\mathbf{Q}_1^T\mathbf{A}_k\|_F$, we need to perform the same procedure described for $\mathbf{T}_2$ for the matrix product $\mathbf{T}_1$ in equation (4-1). Thus, for $\mathbf{T}_1$ and by the SVD of $\mathbf{A}$ in (2-1), we have

$$\mathbf{T}_1 = \mathbf{A}\mathbf{\Omega} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T\mathbf{\Omega}. \tag{4-31}$$

By partitioning $\mathbf{\Sigma}$, we obtain

$$\mathbf{T}_1 = \mathbf{U}\begin{bmatrix} \mathbf{\Sigma}_1 & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{\Sigma}_2 & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{\Sigma}_3 \end{bmatrix}\begin{bmatrix} \mathbf{V}_1^T\mathbf{\Omega} \\ \\ \mathbf{V}_2^T\mathbf{\Omega} \end{bmatrix} = \mathbf{Q}_1\mathbf{R}_1. \tag{4-32}$$

Having (4-17), we now choose a matrix $\mathbf{X} \in \mathbb{R}^{\ell\times\ell}$, which orients the first k

columns of $\mathbf{T}_1\mathbf{X}$ in the directions of the k leading singular vectors in $\mathbf{U}$. Thus, we have

$$\mathbf{X} = \left[\boldsymbol{\Omega}_1^{\dagger}\begin{pmatrix}\boldsymbol{\Sigma}_1 & \mathbf{0}\\ \mathbf{0} & \boldsymbol{\Sigma}_2\end{pmatrix}^{-1}, \bar{\mathbf{X}}\right], \tag{4-33}$$

where the $\bar{\mathbf{X}} \in \mathbb{R}^{\ell\times p}$ is chosen such that $\mathbf{X} \in \mathbb{R}^{\ell\times\ell}$ is non-singular, and $\boldsymbol{\Omega}_1\bar{\mathbf{X}} = \mathbf{0}$. We now write

$$\mathbf{T}_1\mathbf{X} = \mathbf{A}\boldsymbol{\Omega}\mathbf{X} = \mathbf{U}\begin{bmatrix}\begin{pmatrix}\boldsymbol{\Sigma}_1 & \mathbf{0}\\ \mathbf{0} & \boldsymbol{\Sigma}_2\end{pmatrix}\boldsymbol{\Omega}_1\\ \boldsymbol{\Sigma}_3\boldsymbol{\Omega}_2\end{bmatrix}\mathbf{X} = \mathbf{U}\begin{bmatrix}\mathbf{I} & \mathbf{0} & \mathbf{0}\\ \mathbf{0} & \mathbf{I} & \mathbf{0}\\ \mathbf{D}_1 & \mathbf{D}_2 & \mathbf{D}_3\end{bmatrix}, \tag{4-34}$$

where $\mathbf{D}_1 = \boldsymbol{\Sigma}_3\boldsymbol{\Omega}_2\boldsymbol{\Omega}_1^{\dagger}\boldsymbol{\Sigma}_1^{-1} \in \mathbb{R}^{n-\ell+p\times k}$, $\mathbf{D}_2 = \boldsymbol{\Sigma}_3\boldsymbol{\Omega}_2\boldsymbol{\Omega}_1^{\dagger}\boldsymbol{\Sigma}_2^{-1} \in \mathbb{R}^{n-\ell+p\times\ell-p-k}$, and $\mathbf{D}_3 = \boldsymbol{\Sigma}_3\boldsymbol{\Omega}_2\bar{\mathbf{X}} \in \mathbb{R}^{n-\ell+p\times p}$. We then write

$$\mathbf{U}\begin{bmatrix}\mathbf{I} & \mathbf{0} & \mathbf{0}\\ \mathbf{0} & \mathbf{I} & \mathbf{0}\\ \mathbf{D}_1 & \mathbf{D}_2 & \mathbf{D}_3\end{bmatrix} = \bar{\mathbf{Q}}\bar{\mathbf{R}} = [\bar{\mathbf{Q}}_1 \quad \bar{\mathbf{Q}}_2 \quad \bar{\mathbf{Q}}_3]\begin{bmatrix}\bar{\mathbf{R}}_{11} & \bar{\mathbf{R}}_{12} & \bar{\mathbf{R}}_{13}\\ \mathbf{0} & \bar{\mathbf{R}}_{22} & \bar{\mathbf{R}}_{23}\\ \mathbf{0} & \mathbf{0} & \bar{\mathbf{R}}_{33}\end{bmatrix}. \tag{4-35}$$

and as a result, we have

$$\mathbf{U}\begin{bmatrix}\mathbf{I}\\ \mathbf{0}\\ \mathbf{D}_1\end{bmatrix} = \bar{\mathbf{Q}}_1\bar{\mathbf{R}}_{11}. \tag{4-36}$$

By Lemma 4.5, we have

$$\mathbf{Q}_1\mathbf{Q}_1^T = \bar{\mathbf{Q}}\bar{\mathbf{Q}}^T. \tag{4-37}$$

where $\mathbf{Q}_1$ and $\bar{\mathbf{Q}}$ are the Q-factors of the QR decompositions of $\mathbf{T}_1$ (4-32) and $\mathbf{T}_1\mathbf{X}$ (4-35), respectively. By combining Lemma 4.5 and Proposition 4.8, it follows that

$$\|(\mathbf{I} - \mathbf{Q}_1\mathbf{Q}_1^T)\mathbf{A}_k\|_F = \|(\mathbf{I} - \bar{\mathbf{Q}}\bar{\mathbf{Q}}^T)\mathbf{A}_k\|_F \leq \|(\mathbf{I} - \bar{\mathbf{Q}}_1\bar{\mathbf{Q}}_1^T)\mathbf{A}_k\|_F. \tag{4-38}$$

By the definition of $\bar{\mathbf{Q}}_1$, equation (4-36), it follows

$$\mathbf{I} - \bar{\mathbf{Q}}_1\bar{\mathbf{Q}}_1^T = \mathbf{U}\begin{bmatrix}\mathbf{I} - \bar{\mathbf{D}}^{-1} & \mathbf{0} & -\bar{\mathbf{D}}^{-1}\mathbf{D}_1^T\\ \mathbf{0} & -\mathbf{I} & \mathbf{0}\\ -\mathbf{D}_1\bar{\mathbf{D}}^{-1} & \mathbf{0} & \mathbf{I} - \mathbf{D}_1\bar{\mathbf{D}}^{-1}\mathbf{D}_1^T\end{bmatrix}\mathbf{U}^T, \tag{4-39}$$

where $\mathbf{D}_1$ is defined in (4-34), and $\bar{\mathbf{D}}^{-1}$ is defined as follows:

$$\bar{\mathbf{R}}_{11}^{-1}\bar{\mathbf{R}}_{11}^{-T} = (\bar{\mathbf{R}}_{11}^T\bar{\mathbf{R}}_{11})^{-1} = (1 + \mathbf{D}_1^T\mathbf{D}_1)^{-1} = \bar{\mathbf{D}}^{-1}.$$

We then write

$$(\mathbf{I} - \bar{\mathbf{Q}}_1\bar{\mathbf{Q}}_1^T)\mathbf{A}_k = (\mathbf{I} - \bar{\mathbf{Q}}_1\bar{\mathbf{Q}}_1^T)\mathbf{U}\begin{bmatrix}\boldsymbol{\Sigma}_1\\ \mathbf{0}\\ \mathbf{0}\end{bmatrix}\mathbf{V}^T = \underbrace{\mathbf{U}\begin{bmatrix}(\mathbf{I} - \bar{\mathbf{D}}^{-1})\boldsymbol{\Sigma}_1\\ \mathbf{0}\\ -\mathbf{D}_1\bar{\mathbf{D}}^{-1}\boldsymbol{\Sigma}_1\end{bmatrix}}_{\mathbf{H}}\mathbf{V}^T. \tag{4-40}$$

By the definition of the Frobenius norm, it follows that

$$\begin{aligned}\|(\mathbf{I}-\bar{\mathbf{Q}}_1\bar{\mathbf{Q}}_1^T)\mathbf{A}_k\|_F =&\sqrt{\text{trace}(\mathbf{H}^T\mathbf{H})} = \sqrt{\text{trace}(\mathbf{\Sigma}_1\mathbf{D}_1^T(\mathbf{I}+\mathbf{D}_1\mathbf{D}_1^T)^{-1}\mathbf{D}_1\mathbf{\Sigma}_1^T)}\\ \leq&\sqrt{\text{trace}([\|\mathbf{D}_1\mathbf{\Sigma}_1\|_2^2\mathbf{I}+\mathbf{\Sigma}_1^{-2}]^{-1})}\\ =&\|\mathbf{D}_1\mathbf{\Sigma}_1\|_2\sqrt{\text{trace}(\mathbf{\Sigma}_1[\|\mathbf{D}_1\mathbf{\Sigma}_1\|_2^2\mathbf{I}+\mathbf{\Sigma}_1^{-2}]^{-1})\mathbf{\Sigma}_1^T)}\\ \leq&\frac{\sqrt{k}\|\mathbf{D}_1\mathbf{\Sigma}_1\|_2\sigma_1}{\sqrt{{\sigma_1}^2+\|\mathbf{D}_1\mathbf{\Sigma}_1\|_2^2}}.\end{aligned} \tag{4-41}$$

By plugging this into (4-38), we obtain

$$\|(\mathbf{I}-\mathbf{Q}_1\mathbf{Q}_1^T)\mathbf{A}_k\|_F \leq\frac{\sqrt{k}\|\mathbf{D}_1\mathbf{\Sigma}_1\|_2\sigma_1}{\sqrt{{\sigma_1}^2+\|\mathbf{D}_1\mathbf{\Sigma}_1\|_2^2}}. \tag{4-42}$$

We now present the result.

Theorem 4.9 *With the notation of Theorem 4.7, and $\varrho = 2, F$, the approximation error for Algorithm 7 must satisfy*

$$\|\mathbf{A}-\hat{\mathbf{A}}_{SOR}\|_\varrho \leq \|\mathbf{A}_0\|_\varrho + \sqrt{\frac{\alpha^2\|\mathbf{\Omega}_2\|_2^2\|{\mathbf{\Omega}_1}^\dagger\|_2^2}{1+\beta^2\|\mathbf{\Omega}_2\|_2^2\|{\mathbf{\Omega}_1}^\dagger\|_2^2}} + \sqrt{\frac{\eta^2\|\mathbf{\Omega}_2\|_2^2\|{\mathbf{\Omega}_1}^\dagger\|_2^2}{1+\tau^2\|\mathbf{\Omega}_2\|_2^2\|{\mathbf{\Omega}_1}^\dagger\|_2^2}}, \tag{4-43}$$

where $\alpha = \sqrt{k}\frac{\sigma_{\ell-p+1}^2}{\sigma_k}$, $\beta = \frac{\sigma_{\ell-p+1}^2}{\sigma_1\sigma_k}$, $\eta = \sqrt{k}\sigma_{\ell-p+1}$, and $\tau = \frac{\sigma_{\ell-p+1}}{\sigma_1}$. When the power method is used (Algorithm 8), $\alpha = \sqrt{k}\frac{\sigma_{\ell-p+1}^2}{\sigma_k}\left(\frac{\sigma_{\ell-p+1}}{\sigma_k}\right)^{2q}$, $\beta = \frac{\sigma_{\ell-p+1}^2}{\sigma_1\sigma_k}\left(\frac{\sigma_{\ell-p+1}}{\sigma_k}\right)^{2q}$, $\eta = \frac{\sigma_k}{\sigma_{\ell-p+1}}\alpha$, and $\tau = \frac{1}{\sigma_{\ell-p+1}}\beta$.

Proof. The proof is given in Appendix 8.2.4.

Theorem 4.7 shows that the accuracy of singular values depends strongly on the ratio $\frac{\sigma_{\ell-p+1}}{\sigma_j}$ for $j = 1, ..., k$, whereas by Theorem 4.9, the accuracy of the low-rank approximation depends on $\frac{\sigma_{\ell-p+1}}{\sigma_k}$. The power method decreases the extra factors in the error bounds by driving down the aforesaid ratios exponentially fast. Thus, by increasing the number of iterations q, we make the extra factors as close to zero as we wish. However, this increases the cost of the algorithm.

4.2.2 Average-Case Error Bounds

In this section, we provide an average-case error analysis for the SOR-SVD algorithm, which, in contrast to the argument in Section 4.2.1, depends on distributional assumptions on the random matrix $\mathbf{\Omega}$. To be precise, $\mathbf{\Omega}$ has a standard Gaussian distribution which is invariant under all orthogonal transformations. This means that for any matrix $\mathbf{V}$ with orthonormal columns, the product $\mathbf{V}^T\mathbf{\Omega}$ has the same standard Gaussian distribution. This allows us to take advantage of the vast literature on properties of Gaussian matrices.

We begin by stating a few propositions that are used later on.

Proposition 4.10 *(Gu [36]) Let $\mathbf{G} \in \mathbb{R}^{m\times n}$ be a Gaussian matrix. For any $\alpha > 0$, we have*

$$\mathbb{E}\left(\frac{1}{\sqrt{1+\alpha^2\|\mathbf{G}\|_2^2}}\right) \geq \frac{1}{\sqrt{1+\alpha^2\nu^2}}, \tag{4-44}$$

where $\nu = \sqrt{m}+\sqrt{n}+7$.

Proposition 4.11 *With the notation of Proposition 4.10 and, furthermore, for any $\beta > 0$, we have*

$$\mathbb{E}\left(\sqrt{\frac{\alpha^2\|\mathbf{G}\|_2^2}{1+\beta^2\|\mathbf{G}\|_2^2}}\right) \leq \sqrt{\frac{\alpha^2\nu^2}{1+\beta^2\nu^2}}. \tag{4-45}$$

where ν is defined in (4-44).

Proof. The proof is given in Appendix 8.2.5.

Proposition 4.12 *(Gu [36]) Let $\mathbf{G} \in \mathbb{R}^{\ell-p\times\ell}$ be a Gaussian matrix. Then rank$(\mathbf{G}) = \ell - p$ with probability 1. For $p \geq 2$ and any $\alpha > 0$*

$$\mathbb{E}\left(\frac{1}{\sqrt{1+\alpha^2\|\mathbf{G}^\dagger\|_2^2}}\right) \geq \frac{1}{\sqrt{1+\alpha^2\nu^2}}, \tag{4-46}$$

where $\nu = \dfrac{4e\sqrt{\ell}}{p+1}$.

Proposition 4.13 *With the notation of Proposition 4.12 and, furthermore, for any $\beta > 0$, we have*

$$\mathbb{E}\left(\sqrt{\frac{\alpha^2\|\mathbf{G}^\dagger\|_2^2}{1+\beta^2\|\mathbf{G}^\dagger\|_2^2}}\right) \leq \sqrt{\frac{\alpha^2\nu^2}{1+\beta^2\nu^2}}. \tag{4-47}$$

where ν is defined in (4-46).

Proof. The proof is given in Appendix 8.2.6.

We now present the result.

Theorem 4.14 *With the notation of Theorem 4.7 and $\gamma_j = \frac{\sigma_{\ell-p+1}}{\sigma_j}$, for $j = 1, ..., k$, for Algorithm 7, we have*

$$\mathbb{E}(\sigma_j(\hat{\mathbf{A}}_{SOR})) \geq \frac{\sigma_j}{\sqrt{1+\nu^2\gamma_j^4}}, \tag{4-48}$$

and when the power method is used (Algorithm 8), we have

$$\mathbb{E}(\sigma_j(\hat{\mathbf{A}}_{SOR})) \geq \frac{\sigma_j}{\sqrt{1+\nu^2\gamma_j^{4q+4}}}, \tag{4-49}$$

where $\nu = \nu_1\nu_2$, $\nu_1 = \sqrt{n-\ell+p}+\sqrt{\ell}+7$, and $\nu_2 = \frac{4e\sqrt{\ell}}{p+1}$.

Proof. The proof is given in Appendix 8.2.7.

We now present a theorem that establishes average error bounds on the error of the low-rank approximation.

Theorem 4.15 *With the notation of Theorem 4.7, and $\varrho = 2, F$, for Algorithm 7, we have*

$$\mathbb{E}\|\mathbf{A} - \hat{\mathbf{A}}_{SOR}\|_\varrho \leq \ \|\mathbf{A}_0\|_\varrho + (1+\gamma_k)\sqrt{k}\nu\sigma_{\ell-p+1}, \tag{4-50}$$

and when the power method is used (Algorithm 8), we have

$$\mathbb{E}\|\mathbf{A} - \hat{\mathbf{A}}_{SOR}\|_\varrho \leq \ \|\mathbf{A}_0\|_\varrho + (1+\gamma_k)\sqrt{k}\nu\sigma_{\ell-p+1}\gamma_k^{2q}, \tag{4-51}$$

where γ_k and ν are defined in Theorem 4.14.

Proof. The proof is given in Appendix 8.2.8.

Remark 4.16 *The expectation bounds set forth in Theorem 4.14 and 4.15 describe the typical behavior of SOR-SVD due to measure concentration effects. Approximation error tail bounds can be developed by making use of the methods from [36, Section 5.3].*

4.2.3 Computational Complexity

To decompose the matrix $\mathbf{A}$, SOR-SVD of Algorithm 7 incurs the following costs: Step 1 costs $O(n\ell)$, Step 2 costs $O(mn\ell)$, Step 3 costs $O(mn\ell)$, Step 4 costs $O(m\ell^2 + n\ell^2)$, Step 5 costs $O(mn\ell + m\ell^2)$ (if the matrix $\mathbf{M}$ is approximated by $\mathbf{M}_{\text{approx}}$ of equation (4-8) in this step, the cost would be $O(m\ell^2 + n\ell^2 + \ell^3)$), Step 6 costs $O(\ell^3)$, Step 7 (forming the left and right approximate bases) costs $O(m\ell k + n\ell k)$. The dominant cost of Step 1 through Step 7 occurs when multiplying $\mathbf{A}$ and $\mathbf{A}^T$ with the corresponding matrices. Thus

$$C_{\text{SOR-SVD}} = O(mn\ell). \tag{4-52}$$

The sample size parameter ℓ is typically close to the minimal rank k. The cost in (4-52) results from the dense matrix $\mathbf{A}$ considered. If $\mathbf{A}$ is sparse, the arithmetic cost is proportional to the number s of non-zero entries of $\mathbf{A}$: $O(s\ell)$.

Algorithm 7 requires either three or two passes (when $\mathbf{M}$ is approximated by $\mathbf{M}_{\text{approx}}$) through data to factor $\mathbf{A}$. When the power method is incorporated (Algorithm 8), SOR-SVD requires either $(2q+3)$ or $(2q+2)$ passes (when $\mathbf{M}$ is approximated by $\mathbf{M}_{\text{approx}}$) over the data with arithmetic costs of $(2q+3)C_{\text{SOR-SVD}}$ or $(2q+2)C_{\text{SOR-SVD}}$, respectively.

Except for matrix-matrix multiplications, which are easily parallelizable, SOR-SVD performs two QR decompositions on matrices of size $m \times \ell$ and $n \times \ell$, whereas R-SVD performs one QR decomposition on an $m \times \ell$ matrix. Recently, Demmel et al. [21] developed communication-avoiding sequential and parallel QR decomposition algorithms that perform the computations with optimal communication costs. Thus, this step of both algorithms can be implemented efficiently. In addition, SOR-SVD performs an SVD on an $\ell \times \ell$ matrix, $\mathbf{M}$ (or $\mathbf{M}_{\text{approx}}$) in Algorithms 7 and 8, whereas the R-SVD performs an SVD on a $n \times \ell$ matrix, $\mathbf{B}$ in Algorithm 2. Standard techniques to compute an SVD, however, are challenging for parallelization [18, 61]. Given a large input matrix $\mathbf{A}$ for which a rank-k approximation to be computed, where $k \leq \ell \ll \min\{m, n\}$, $\mathbf{M}$ (or $\mathbf{M}_{\text{approx}}$) would be much smaller than $\mathbf{B}$. Considering the size of $\mathbf{M}$ (or $\mathbf{M}_{\text{approx}}$) and, further, having known that current advanced computers have hardware switches that are controlled in software [23], the SVD of the $\ell \times \ell$ matrix can be computed either *in-core* on a sequential processor or with minimum communication cost on parallel processors. Thus, this step of SOR-SVD can be executed efficiently. This significantly reduces the computational time of SOR-SVD, an advantage over R-SVD.

4.3 Numerical Experiments

In this section, we evaluate the performance of the SOR-SVD algorithm through numerical tests. Our goal is to experimentally investigate the behavior of the SOR-SVD algorithm in different scenarios.

In Section 4.3.1, we consider two classes of synthetic matrices to experimentally verify that the SOR-SVD algorithm provides highly accurate singular values and low-rank approximations. We compare the performance of SOR-SVD with those of the SVD, R-SVD (Algorithm 2), TSR-SVD (Algorithm 3), and SFRA (Algorithm 4).

In Section 4.3.2, we experimentally investigate the tightness of the low-rank approximation error bounds for the spectral and Frobenius norm provided in Theorem 4.9.

In Section 4.3.3, we apply the SOR-SVD to recovering a low-rank plus a sparse matrix through experiments on (i) synthetically generated data with various dimensions, numerical rank and gross errors, and (ii) real-time data in applications to background subtraction in surveillance video, and shadow and specularity removal from face images.

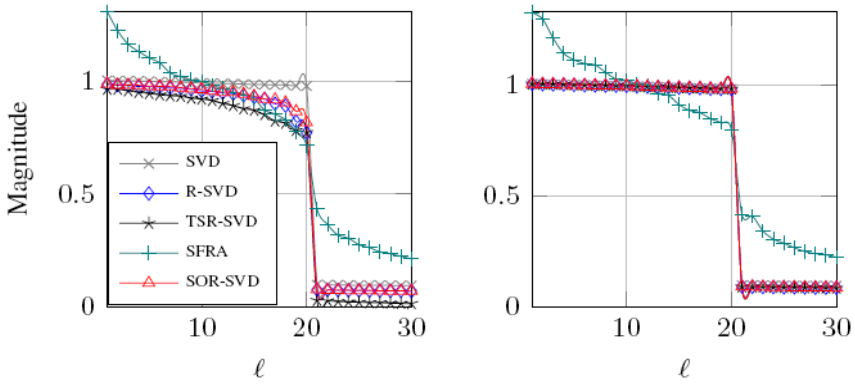


Figure 4.1: Comparison of singular values for the noisy low-rank matrix. No power method, ($q = 0$) (left), and $q = 2$ (right).

4.3.1 Synthetic Matrices

We first describe two types of data matrices that we use in our tests to assess the behavior of SOR-SVD. For the sake of simplicity, we focus on square matrices.

- Matrix 1: A Noisy Low-Rank Matrix. This matrix is generated as described in Section 3.3.1 (first example). Here we set gap = 0.1.
- Matrix 2: A Matrix with Rapidly Decaying Singular Values. Matrix $\mathbf{A} \in \mathbb{R}^{1000\times1000}$ is generated as $\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$, where $\mathbf{U}$ and $\mathbf{V}$ are random orthonormal matrices, $\mathbf{\Sigma} = \text{diag}(1^{-1}, 2^{-1}, ..., n^{-1})$. We set the rank $k = 10$.

To study the behavior of SOR-SVD and, further, to provide a fair comparison, we factor the matrix $\mathbf{A}$ using the SVD, R-SVD, TSR-SVD, SOR-SVD, and SFRA algorithms. R-SVD, TSR-SVD, and SOR-SVD all require a predetermined sample size parameter ℓ to compute an approximation of $\mathbf{A}$. Thus, we arbitrarily set $\ell = 38$ for the first test matrix, and $\ell = 18$ for the second test matrix. These randomized algorithms require the same number of passes through $\mathbf{A}$, either two or $2q + 2$ when the power iteration is used to perform the factorization. SFRA, however, requires sketch size parameters (p_1, p_2) to compute an approximation. We thus set $p_1 = 2k+1$ and $p_2 = 2p_1+1$ for both classes of matrices studied, based on the recommendations provided in the work [86, Section 4.5]. We then compare the singular values and low-rank approximations computed by the algorithms mentioned.

Figures 4.1 and 4.2 display the singular values computed by each algorithm for the two matrices. SOR-SVD and SFRA use a truncated SVD on the corresponding reduced-size matrices $\mathbf{M}$ and $\mathbf{X}$, respectively. However, we

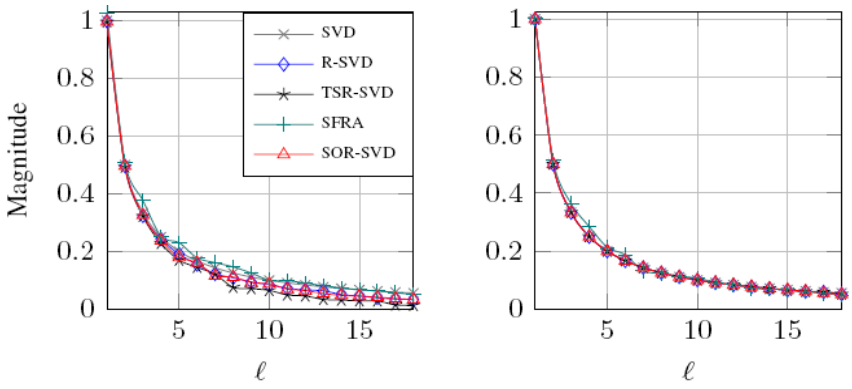


Figure 4.2: Comparison of singular values for the matrix with polynomially decaying singular values. No power method, ($q = 0$) (left), and $q = 2$ (right).

show the results for a full SVD, i.e., a full SVD performed on $\mathbf{M}$ and $\mathbf{X}$, for the purpose of comparison.

From the figures, we make the following observations:

- For the first matrix (noisy low-rank), SFRA shows the poorest performance among the algorithms studied: it considerably overestimates some of the leading as well as trailing singular values of the matrix.
- For the noisy low-rank matrix, when the power method is not used ($q = 0$), the SOR-SVD approximations of some singular values are good estimates of the true ones, while for the others the estimates are relatively poor. In this case, SOR-SVD outperforms TSR-SVD, while having a similar performance to R-SVD. When $q = 2$, the quality of estimated singular values shows a substantial improvement, and no loss of accuracy is seen in the approximations by SOR-SVD, compared to the optimal SVD.
- For the second matrix (polynomially decaying spectrum), when $q = 0$, TSR-SVD has the poorest performance among the randomized algorithm considered, while R-SVD, SOR-SVD, and SFRA show similar performances. When $q = 2$, which corresponds to incorporating the power method, SOR-SVD matches the performance of the optimal SVD. In this case, R-SVD and TSR-SVD approximate the singular values of the matrix with very high accuracy.

To compare the quality of low-rank approximations, we first construct a rank-k approximation $\hat{\mathbf{A}}_{\text{out}}$ to $\mathbf{A}$ by each algorithm. For R-SVD, TSR-SVD, and SOR-SVD, we compute the approximations by varying the sample size parameter ℓ, while the rank is fixed. For SFRA, however, we compute the rank-k approximation by varying the sketch size parameters (p_1, p_2) through

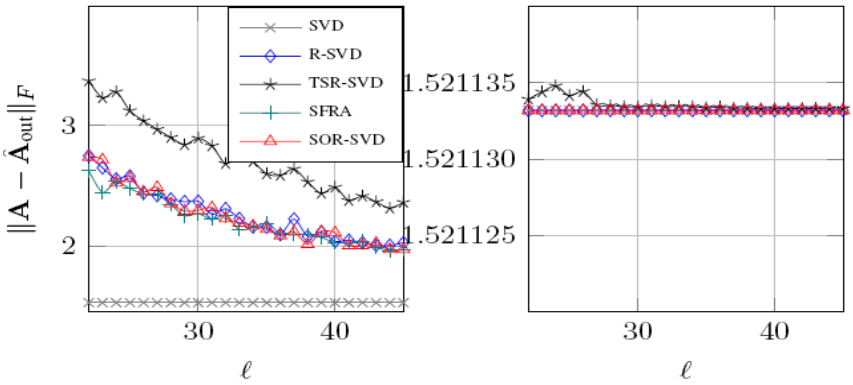


Figure 4.3: Comparison of the Frobenius norm approximation error for the noisy low-rank matrix. No power method, ($q = 0$) (left), and $q = 2$ (right).

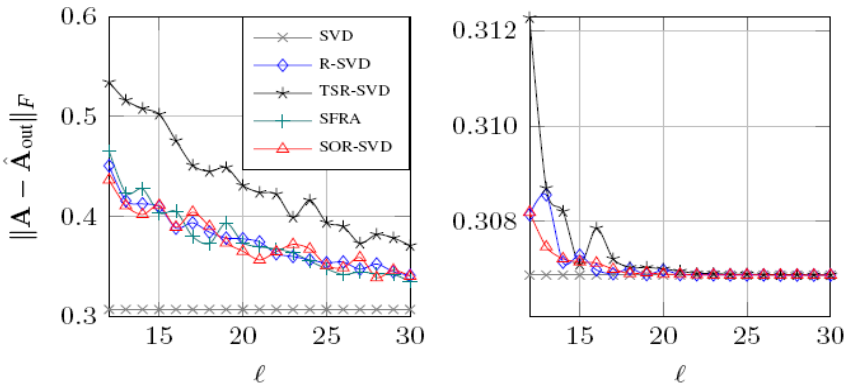


Figure 4.4: Comparison of the Frobenius norm approximation error for the matrix with polynomially decaying singular values. No power method, ($q = 0$) (left), and $q = 2$ (right).

increasing the target rank. We then calculate the Frobenius-norm error:

$$e_k = \|\mathbf{A} - \hat{\mathbf{A}}_{\text{out}}\|_F. \tag{4-53}$$

Figures 4.3 and 4.4 display the results. Since SFRA does not employ the power iteration, we have discarded its results from the second graphs of the figures in order to clearly demonstrate the results for other algorithms. We make the following observations:

- When no power method is employed by the randomized algorithms, for both test matrices TSR-SVD shows the worst performance among the algorithms, while R-SVD, SOR-SVD, and SFRA show similar behavior.
- When the power method is incorporated, for both test matrices SFRA has the poorest performance among the algorithms (we have discarded its results from the graphs since they are far away from those of the

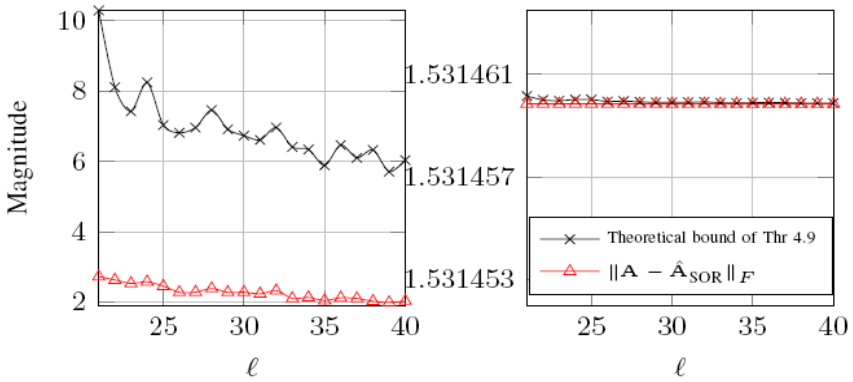


Figure 4.5: Comparison of the Frobenius norm error of SOR-SVD with the theoretical bound (Theorem 4.9). No power method, ($q = 0$) (left), and $q = 2$ (right).

optimal SVD). In this case, SOR-SVD approximates the matrices with almost no loss of accuracy compared to the SVD.

4.3.2 Empirical Evaluation of SOR-SVD Error Bounds

The theoretical error bounds for SOR-SVD are given in Theorem 4.9. To evaluate the tightness of the bounds provided by Theorem 4.9, we form an input matrix according to Matrix 1. With the rank $k = 20$ fixed, we increase the sample size parameter ℓ, considering the assumption $2 \leq p \leq \ell - k$. A comparison between the theoretical bounds and what are achieved in practice is shown in Figures 4.5 and 4.6; Figure 4.5 compares the Frobenius norm error with the corresponding theoretical bound, and Figure 4.6 compares the spectral norm error with the corresponding theoretical bound.

The effect of the power scheme can be easily seen from the figures; when $q = 2$, the theoretical bounds given in Theorem 4.9 closely track the error in the low-rank approximation of Algorithm 8. We conclude that for the noisy low-rank matrix, the theoretical error bounds are empirically sharp.

4.3.3 Robust PCA Using SOR-SVD

We now apply SOR-SVD to solve the robust PCA problem [9,14,93]; see Section 2.6 of Chapter 2 for a detailed description of robust PCA. We retain the original objective function proposed in [9,14,59,93], and apply SOR-SVD as a surrogate to the SVD to solve the optimization problem (2-26). We adopt the continuation technique [59,85], which increases μ in each iteration. The

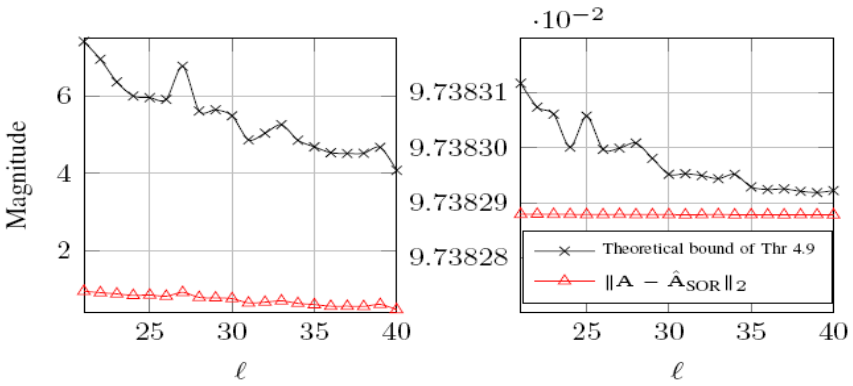


Figure 4.6: Comparison of the spectral norm error of SOR-SVD with the theoretical bound (Theorem 4.9). No power method, ($q = 0$) (left), and $q = 2$ (right).

pseudocode of the proposed method, called ALM-SOR-SVD hereafter, is given in Table 4.1.

Table 4.1: Pseudo-code for RPCA solved by the ALM-SOR-SVD method.

Input: Matrix $\mathbf{X}, \lambda, \mu_0, \bar{\mu}, \rho, \mathbf{Y}_0, \mathbf{S}_0, k = 0$;
Output: Low-rank plus sparse matrix
1: **while** the algorithm does not converge **do**
2: $(\mathbf{U}, \mathbf{\Sigma}, \mathbf{V}) = \text{sor-svd}(\mathbf{X} - \mathbf{S}_k + \mu_k^{-1}\mathbf{Y}_k)$;
3: $\mathbf{L}_{k+1} = \mathbf{U}\mathcal{S}_{\mu_k^{-1}}(\mathbf{\Sigma})\mathbf{V}^T$;
4: $\mathbf{S}_{k+1} = \mathcal{S}_{\lambda\mu_k^{-1}}(\mathbf{X} - \mathbf{L}_{k+1} + \mu_k^{-1}\mathbf{Y})$;
5: $\mathbf{Y}_{k+1} = \mathbf{Y}_k + \mu_k(\mathbf{X} - \mathbf{L}_{k+1} - \mathbf{S}_{k+1})$;
6: $\mu_{k+1} \leftarrow \max(\rho\mu_k, \bar{\mu})$;
7: **end while**
8: **return** $\mathbf{L}^*$ and $\mathbf{S}^*$

In Table 4.1 $\mathcal{S}_\varepsilon(x) = \text{sgn}(x)\max(|x| - \varepsilon, 0)$ is a soft-thresholding operator [38], and λ, μ_0, $\bar{\mu}$, ρ, $\mathbf{Y}_0$, and $\mathbf{S}_0$ are initial values.

In the next subsections, we verify the efficiency and efficacy of the ALM-SOR-SVD to solve the RPCA problem on randomly generated data, as well as real-time data. We compare the experimental results obtained with those of applying the partial SVD (by using PROPACK package) [57]. The PROPACK function provides an efficient algorithm, suitable for approximating large low-rank matrices, which computes a specified number of largest singular values and corresponding singular vectors of a matrix by making use of the Lanczos bidiagonalization algorithm with partial reorthogonalization (BPRO). We run the experiments in MATLAB on a desktop PC with a 3 GHz intel Core i5-4430 processor and 8 GB of memory.

4.3.3.1 Synthetic Data Recovery

We generate rank-k $\mathbf{X} = \mathbf{L} + \mathbf{S}$ as a sum of a low-rank matrix $\mathbf{L} \in \mathbb{R}^{n\times n}$ and a sparse matrix $\mathbf{S} \in \mathbb{R}^{n\times n}$. The matrix $\mathbf{L}$ is generated by a matrix multiplication $\mathbf{L} = \mathbf{U}\mathbf{V}^T$, where $\mathbf{U}$, $\mathbf{V} \in \mathbb{R}^{n\times k}$ have standard Gaussian distributed entries. The matrix $\mathbf{S}$ has s non-zero entries independently drawn from the set {-50, 50}. We apply the ALM-SOR-SVD method and the ALM method using the partial SVD, hereafter ALM-PSVD, on $\mathbf{X}$ to recover $\mathbf{L}$ and $\mathbf{S}$.

Table 4.2 summarizes the numerical results where $\text{rank}(\mathbf{L}) = 0.05\times n$ and $s = \|\mathbf{S}\|_0 = 0.05\times n^2$, and Table 4.3 presents the results for a more challenging scenario where $\text{rank}(\mathbf{L}) = 0.05 \times n$ and $s = \|\mathbf{S}\|_0 = 0.1 \times n^2$. In the Tables, *Time* denotes the computational time in seconds, *Iter.* denotes the number of iterations, and ξ denotes the relative error defined as $\|\mathbf{X} - \hat{\mathbf{L}} - \hat{\mathbf{S}}\|_F/\|\mathbf{X}\|_F$, where $(\hat{\mathbf{L}}, \hat{\mathbf{S}})$ is the pair of solution of either algorithm. For the simulations, the initial values suggested in [59] are adopted, and the algorithms stop when the following condition holds:

$$\frac{\|\mathbf{X} - \hat{\mathbf{L}} - \hat{\mathbf{S}}\|_F}{\|\mathbf{X}\|_F} < 10^{-7}. \tag{4-54}$$

Since SOR-SVD and the truncated SVD require a predetermined rank ℓ to perform the decomposition, we set $\ell = 2r$, a random start, and $q = 1$ for SOR-SVD.

The results in Tables 4.2 and 4.3 lead us to make several conclusions on the ALM-SOR-SVD method:

- It successfully detects the numerical rank k in all cases.
- It provides the exact recovery of the sparse matrix $\mathbf{S}$ from $\mathbf{X}$, with the same number of iterations compared to the ALM-PSVD method.
- In terms of runtime, it outperforms the ALM-PSVD method with speedups of up to 47 times.

4.3.3.2 Background Subtraction in Surveillance Video

In this experiment, we use four different real-time videos introduced in [58]. The first video sequence has 200 grayscale frames with dimensions 176×144 in each frame, and has been taken in a hall of an airport. We stack each frame as a column of the data matrix $\mathbf{X} \in \mathbb{R}^{25344\times 200}$. The second video has 200 grayscale frames with dimensions 256×320 in each frame, and has been taken in a shopping mall. Thus $\mathbf{X} \in \mathbb{R}^{81920\times 200}$. These two videos have relatively

Table 4.2: Comparison of the ALM-SOR-SVD and ALM-PSVD methods for synthetic data recovery for the case $r(\mathbf{L}) = 0.05 \times n$ and $s = 0.05 \times n^2$.

n	$r(\mathbf{L})$	$\|\mathbf{S}\|_0$	Methods	$r(\hat{\mathbf{L}})$	$\|\hat{\mathbf{S}}\|_0$	Time	Iter.	ξ
500	25	12.5e3	ALM-SOR-SVD	25	12.5e3	0.1	17	6.3e-8
			ALM-PSVD	25	12.5e3	2.2	17	6.6e-8
1000	50	5e4	ALM-SOR-SVD	50	5e4	0.7	17	5.3e-8
			ALM-PSVD	50	5e4	22	17	5.4e-8
2000	100	2e5	ALM-SOR-SVD	100	2e5	4.6	17	5.0e-8
			ALM-PSVD	100	2e5	167	17	5.1e-8
3000	150	45e4	ALM-SOR-SVD	150	45e4	11.8	17	4.9e-8
			ALM-PSVD	150	45e4	563	17	4.8e-8

Table 4.3: Comparison of the ALM-SOR-SVD and ALM-PSVD methods for synthetic data recovery for the case $r(\mathbf{L}) = 0.05 \times n$ and $s = 0.1 \times n^2$.

n	$r(\mathbf{L})$	$\|\mathbf{S}\|_0$	Methods	$r(\hat{\mathbf{L}})$	$\|\hat{\mathbf{S}}\|_0$	Time	Iter.	ξ
500	25	25e3	ALM-SOR-SVD	25	25e3	0.2	20	4.9e-8
			ALM-PSVD	25	25e3	2.5	20	3.2e-8
1000	50	1e5	ALM-SOR-SVD	50	1e5	0.8	19	8.3e-8
			ALM-PSVD	50	1e5	25.9	19	8.1e-8
2000	100	4e5	ALM-SOR-SVD	100	4e5	5.3	19	7.4e-8
			ALM-PSVD	100	4e5	189	19	6.8e-8
3000	150	9e5	ALM-SOR-SVD	150	9e5	13.6	19	7.6e-8
			ALM-PSVD	150	9e5	609	19	7.2e-8

static background. The third video has 200 grayscale frames with dimensions 130×160 in each frame, i.e., $\mathbf{X} \in \mathbb{R}^{20800 \times 200}$, and has been taken from an escalator at an airport. The background of this video changes due to the moving escalator. The fourth video has 250 grayscale frames with dimensions 128×160 in each frame taken in an office. Thus $\mathbf{X} \in \mathbb{R}^{20480 \times 250}$. In this video, the illumination changes drastically, making it very challenging to analyze.

The predetermined rank ℓ, assigned to both algorithms, is obtained by invoking the bound in (3-16). We assign the minimum value of k satisfying (3-16) to ℓ, i.e., $\ell = k$, and set $q = 1$ for SOR-SVD.

Figure 4.7: Background subtraction in surveillance video. Images in columns 1 and 4 are frames of the video sequence of an airport and a shopping mall, respectively. Images in columns 2 and 5 are recovered backgrounds $\hat{\mathbf{L}}$, and columns 3 and 6 correspond to foregrounds $\hat{\mathbf{S}}$ by the ALM-SOR-SVD method.

Figure 4.8: Background subtraction in surveillance video. Images in columns 1 and 4 are frames of the video sequence of an escalator and an office, respectively. Images in columns 2 and 5 are recovered backgrounds $\hat{\mathbf{L}}$, and columns 3 and 6 correspond to foregrounds $\hat{\mathbf{S}}$ by the ALM-SOR-SVD method.

Some sample frames of the videos with corresponding recovered backgrounds and foregrounds are shown in Figures 4.7 and 4.8. We only show the results of the ALM-SOR-SVD method since the results returned by the ALM-PSVD method are visually identical. It is evident that the proposed method can successfully recover the low-rank and sparse components of the data matrix in all scenarios.

Table 4.4 summarizes the numerical results. In all cases, the ALM-SOR-SVD method outperforms the ALM-PSVD method in terms of runtime, while having the same number of iterations.

4.3.3.3 Shadow and Specularity Removal From Face Images

In our experiment, we use face images taken from the Yale B face database [34]. Each image has dimensions 192×168 with a total of 64 illuminations. The images are stacked as columns of the data matrix $\mathbf{X} \in$

Table 4.4: Comparison of the ALM-PSVD and ALM-SOR-SVD methods for real-time data recovery.

Dataset	ALM-PSVD			ALM-SOR-SVD		
	Time	Iter.	ξ	Time	Iter.	ξ
Airport hall 25344×200	14.2	36	6.0e-8	5.7	36	6.6e-8
Shopping mall 81920×200	44.2	36	6.9e-8	19.1	36	6.8e-8
Escalator 20800×200	11.3	36	7.5e-8	4.6	36	6.5e-8
Lobby 20480×250	10.9	37	5.8e-8	6.6	37	5.7e-8
Yale B03 32256×64	6.0	36	9.2e-8	2.5	36	7.5e-8
Yale B08 32256×64	7.2	36	9.8e-8	2.6	36	7.6e-8

$\mathbb{R}^{32256\times64}$. The recovered images are shown in Figure 4.9, and the numerical results are presented in Table 4.4.

We conclude that the ALM-SOR-SVD method successfully recovers the low-rank and sparse matrices from the dataset with speedups of up to 2.7 times, compared to the ALM-PSVD method with the same number of iterations.

Figure 4.9: Removing shadows and specularities from face images. Images in columns 1 and 4 are face images under different illuminations. Images in columns 2 and 5 are are recovered images after removing shadows and specularities by the ALM-SOR-SVD method, and images in columns 3 and 6 correspond to the removed shadows and specularities.

5
Compressed Randomized UTV Decompositions

This chapter presents a rank-revealing decomposition algorithm powered by the randomized sampling schemes termed compressed randomized UTV (CoR-UTV) decomposition, which computes a low-rank approximation of a given matrix.

For a large and dense matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$ with numerical rank k, CoR-UTV computes a low-rank approximation $\hat{\mathbf{A}}_{\mathrm{CoR}}$ of $\mathbf{A}$ such as

$$\hat{\mathbf{A}}_{\mathrm{CoR}} = \mathbf{U}\mathbf{T}\mathbf{V}^T, \tag{5-1}$$

where $\mathbf{U}$ and $\mathbf{V}$ have orthonormal columns, and $\mathbf{T}$ is triangular (either upper or lower, whichever is preferred). Built on SOR-SVD, the difference between the two algorithms is that in CoR-UTV a QR algorithm with column pivoting (QRCP) is used on the compressed matrix rather than an SVD. This allows CoR-UTV (i) to be more computationally efficient and, (ii) to employ communication-avoiding QRCP algorithm [20, 26] to deliver the factorization, making it even more suitable on modern computational platforms than SOR-SVD. CoR-UTV only requires a few passes through data, for an $\mathbf{A}$ stored externally, and runs in $O(mnk)$ flops. The rank-revealing property of CoR-UTV is proved, and upper bounds on the error of the low-rank approximation are given. CoR-UTV is applied to treat an image reconstruction problem, as well as the robust PCA problem in applications of background subtraction in surveillance video and shadow and specularity removal from face images.

5.1
The CoR-UTV Algorithm

The CoR-UTV decomposition constructs a low-rank approximation of an input matrix in the form of (5-1). We focus on a matrix $\mathbf{A}$ with $m \geq n$, where CoR-UTV produces a middle matrix $\mathbf{T}$ that is upper triangular, i.e., URV decomposition [79]. The modifications required for a corresponding CoR-UTV algorithm for the other case where $m < n$, i.e., ULV decomposition [81], that produces a lower triangular middle matrix $\mathbf{T}$ is straightforward. For a theoretical comparison of the URV and ULV decompositions see [31, 40, 79, 81, 82] and the references therein. We also present a version of CoR-UTV with

power iteration, which improves the performance of the algorithm at an extra computational cost.

Given a matrix $\mathbf{A} \in \mathbb{R}^{m\times n}$, where $m \geq n$, with numerical rank k and an integer $k \leq \ell < n$, CoR-UTV is computed by taking the following seven steps:

1. Generate a standard Gaussian matrix $\mathbf{\Psi} \in \mathbb{R}^{n\times \ell}$,

2. Compute the matrix product:

$$\mathbf{C}_1 = \mathbf{A}\mathbf{\Psi}, \tag{5-2}$$

   The matrix $\mathbf{C}_1 \in \mathbb{R}^{m\times \ell}$ is a projection onto the subspace spanned by columns of $\mathbf{A}$.

3. Compute the matrix product:

$$\mathbf{C}_2 = \mathbf{A}^T\mathbf{C}_1, \tag{5-3}$$

   The matrix $\mathbf{C}_2 \in \mathbb{R}^{n\times \ell}$ is a projection onto the subspace spanned by rows of $\mathbf{A}$.

4. Compute QR decompositions of $\mathbf{C}_1$ and $\mathbf{C}_2$:

$$\mathbf{C}_1 = \mathbf{Q}_1\mathbf{R}_1, \quad \text{and} \quad \mathbf{C}_2 = \mathbf{Q}_2\mathbf{R}_2, \tag{5-4}$$

   The matrices $\mathbf{Q}_1$ and $\mathbf{Q}_2$ are approximate bases for $\mathcal{R}(\mathbf{A})$ and $\mathcal{R}(\mathbf{A}^T)$, respectively.

5. Compute the matrix product:

$$\mathbf{D} = \mathbf{Q}_1^T\mathbf{A}\mathbf{Q}_2, \tag{5-5}$$

   The matrix $\mathbf{D} \in \mathbb{R}^{\ell\times\ell}$ is formed by compression of $\mathbf{A}$ via left and right multiplications by orthonormal bases.

6. Compute a QR decomposition with column pivoting (QRCP) of $\mathbf{D}$:

$$\mathbf{D} = \widetilde{\mathbf{Q}}\widetilde{\mathbf{R}}\widetilde{\mathbf{P}}^T. \tag{5-6}$$

7. Form the CoR-UTV-based low-rank approximation of $\mathbf{A}$:

$$\hat{\mathbf{A}}_{\text{CoR}} = \mathbf{U}\mathbf{T}\mathbf{V}^T, \tag{5-7}$$

   where $\mathbf{U} = \mathbf{Q}_1\widetilde{\mathbf{Q}} \in \mathbb{R}^{m\times\ell}$ and $\mathbf{V} = \mathbf{Q}_2\widetilde{\mathbf{P}} \in \mathbb{R}^{n\times\ell}$ construct approximations to the first ℓ left and right singular vectors of $\mathbf{A}$, respectively, and $\mathbf{T} = \widetilde{\mathbf{R}} \in \mathbb{R}^{\ell\times\ell}$ is upper triangular with diagonals approximating the first ℓ singular values of $\mathbf{A}$.

Algorithm 9 Compressed Randomized UTV (CoR-UTV)

Input: Matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$, integers k and ℓ.
Output: A rank-ℓ approximation.

1: Draw a standard Gaussian matrix $\mathbf{\Psi} \in \mathbb{R}^{n \times \ell}$;
2: Form $\mathbf{C}_1 = \mathbf{A}\mathbf{\Psi}$;
3: Form $\mathbf{C}_2 = \mathbf{A}^T\mathbf{C}_1$;
4: Compute QR decompositions $\mathbf{C}_1 = \mathbf{Q}_1\mathbf{R}_1$ and $\mathbf{C}_2 = \mathbf{Q}_2\mathbf{R}_2$;
5: Form $\mathbf{D} = \mathbf{Q}_1^T\mathbf{A}\mathbf{Q}_2$;
6: Compute the QRCP $\mathbf{D} = \widetilde{\mathbf{Q}}\widetilde{\mathbf{R}}\widetilde{\mathbf{P}}^T$;
7: Form the CoR-UTV-based low-rank approximation of $\mathbf{A}$: $\hat{\mathbf{A}}_{\text{CoR}} = \mathbf{U}\mathbf{T}\mathbf{V}^T$; $\mathbf{U} = \mathbf{Q}_1\widetilde{\mathbf{Q}}, \mathbf{T} = \widetilde{\mathbf{R}}, \mathbf{V} = \mathbf{Q}_2\widetilde{\mathbf{P}}^T$.

The CoR-UTV algorithm is presented in Algorithm 9.

CoR-UTV requires three passes over the data, for a matrix stored externally, but it can be altered to revisit the data only once. To this end, the compressed matrix $\mathbf{D}$ (5-5), is computed by making use of available matrices as follows:

$$\mathbf{D}\mathbf{Q}_2^T\mathbf{\Psi} = \mathbf{Q}_1^T\mathbf{A}\mathbf{Q}_2\mathbf{Q}_2^T\mathbf{\Psi}. \tag{5-8}$$

where both sides of currently unknown $\mathbf{D}$ are postmultiplied by $\mathbf{Q}_2^T\mathbf{\Psi}$. Having defined $\mathbf{A} \approx \mathbf{A}\mathbf{Q}_2\mathbf{Q}_2^T$ and $\mathbf{C}_1 = \mathbf{A}\mathbf{\Psi}$, an approximation to $\mathbf{D}$ is obtained by

$$\mathbf{D}_{\text{approx}} = \mathbf{Q}_1^T\mathbf{C}_1(\mathbf{Q}_2^T\mathbf{\Psi})^\dagger. \tag{5-9}$$

The key differences between CoR-UTV and TSR-SVD (Algorithm 3) are as follows:

- CoR-UTV uses a sketch of the input matrix in order to project it onto its row space, i.e., equation (5-3). This (i) significantly improves the quality of the approximate basis $\mathbf{Q}_2$, and as a result, the approximate right singular subspace of $\mathbf{A}$, compared to that of TSR-SVD, which uses a random matrix for the projection, and (ii) allows (5-9) to provide a highly accurate approximation to (5-5).
- CoR-UTV applies a column-pivoted QR decomposition to $\mathbf{D}$, i.e., equation (5-6), whereas TSR-SVD uses an SVD to factor the compressed matrix. This, as explained later on, reduces the computational costs of the algorithm compared to TSR-SVD.

The key difference between CoR-UTV and SOR-SVD [54], however, lies in the computation of the compressed matrix $\mathbf{D}$; SOR-SVD applies a truncated SVD and, as result, gives a rank-k approximation to $\mathbf{A}$, while CoR-UTV employs a column-pivoted QR decomposition and returns a rank-ℓ approximation. Nevertheless, the SVD is computationally more expensive than the column-pivoted QR and, moreover, standard techniques to computing it

are challenging to parallelize [18, 36, 39]. While recently developed column-pivoted QR algorithms use randomization, which can factor a matrix with optimal/minimum communication cost [26, 62]. This can substantially reduce the computational costs of decomposing the compressed matrix, compared to the SVD, when it does not fit into fast memory.

CoR-UTV may be sufficiently accurate for matrices whose singular values display some decay, however, in applications where the data matrix has a slowly decaying singular values, it may produce poor singular vectors and singular values compared to those of the SVD. Thus, we incorporate q steps of a power iteration [39, 71] to improve the accuracy of the algorithm in these circumstances. Given the matrix $\mathbf{A}$, and integers $k \leq \ell < n$ and q, the resulting algorithm is described in Algorithm 10. Notice that to compute CoR-UTV when the power method is employed, a non-updated $\mathbf{C}_2$ must be used to form $\mathbf{D}_{\text{approx}}$.

Algorithm 10 CoR-UTV with Power Method

Input: Matrix $\mathbf{A} \in \mathbb{R}^{m\times n}$, integers k, ℓ and q.
Output: A rank-ℓ approximation.

1: Draw a standard Gaussian matrix $\mathbf{C}_2 \in \mathbb{R}^{n\times\ell}$;
2: **for** $i = 1$: $q+1$ **do**
3: Form $\mathbf{C}_1 = \mathbf{A}\mathbf{C}_2$;
4: Form $\mathbf{C}_2 = \mathbf{A}^T\mathbf{C}_1$;
5: **end for**
6: Compute QR decompositions $\mathbf{C}_1 = \mathbf{Q}_1\mathbf{R}_1$, $\mathbf{C}_2 = \mathbf{Q}_2\mathbf{R}_2$;
7: Form $\mathbf{D} = \mathbf{Q}_1^T\mathbf{A}\mathbf{Q}_2$ or $\mathbf{D}_{\text{approx}} = \mathbf{Q}_1^T\mathbf{C}_1(\mathbf{Q}_2^T\mathbf{C}_2)^\dagger$;
8: Compute a QRCP $\mathbf{D} = \widetilde{\mathbf{Q}}\widetilde{\mathbf{R}}\widetilde{\mathbf{P}}^T$ or $\mathbf{D}_{\text{approx}} = \widetilde{\mathbf{Q}}\widetilde{\mathbf{R}}\widetilde{\mathbf{P}}^T$;
9: Form the CoR-UTV-based low-rank approximation of $\mathbf{A}$: $\hat{\mathbf{A}}_{\text{CoR}} = \mathbf{U}\mathbf{T}\mathbf{V}^T$; $\mathbf{U} = \mathbf{Q}_1\widetilde{\mathbf{Q}}$, $\mathbf{T} = \widetilde{\mathbf{R}}$,$\mathbf{V} = \mathbf{Q}_2\widetilde{\mathbf{P}}^T$.

5.2 Analysis of CoR-UTV Decompositions

In this section, we provide an analysis of the performance of CoR-UTV, Algorithms 9 and 10. In particular, we discuss the rank-revealing property of the algorithm, and provide upper bounds for the error of the low-rank approximation.

We extensively borrow material from the analysis of SOR-SVD [54] since the two algorithms, CoR-UTV and SOR-SVD, have a few steps similar. However, the key difference is that these randomized algorithms employ different deterministic decompositional methods in order to factor the input matrix. We discuss that CoR-UTV is computationally cheaper and, moreover, can exploit advanced computer architectures better than SOR-SVD.

5.2.1 Rank-Revealing Property

To prove that CoR-UTV is rank-revealing, it is required to show that (i) the $\mathbf{T}$ factor of the decomposition reveals the rank of $\mathbf{A}$, and (ii) the trailing off-diagonal block of $\mathbf{T}$ is small. Furthermore, the relation between the Gaussian random matrix used and the $\mathbf{T}$ factor must be expressed. To be more precise, the quality of the k-th approximated singular value is to be expressed in terms of properties of the Gaussian matrix.

The $\mathbf{T}$ factor of CoR-UTV is, in fact, the $\mathbf{R}$ factor of a numerically stable deterministic QRCP of $\mathbf{D}$ (5-6), where $\mathbf{D}$ is a compressed version of $\mathbf{A}$. We now write (5-6) as:

$$\mathbf{D}\widetilde{\mathbf{P}} = \widetilde{\mathbf{Q}}\widetilde{\mathbf{R}} = \widetilde{\mathbf{Q}} \begin{bmatrix} \widetilde{\mathbf{R}}_{11} & \widetilde{\mathbf{R}}_{11} \\ \mathbf{0} & \widetilde{\mathbf{R}}_{22} \end{bmatrix}. \tag{5-10}$$

Thus, we guarantee that $\widetilde{\mathbf{R}}_{11} \in \mathbb{R}^{k\times k}$ reveals the rank of $\mathbf{D}$, i.e., $\sigma_{\min}(\widetilde{\mathbf{R}}_{11}) \leq \sigma_k(\mathbf{D})$, and $\|\widetilde{\mathbf{R}}_{22}\|_2 \leq \sigma_{k+1}(\mathbf{D})$. See [11,37,40] for more details on QRCP.

Next, we need to show how the singular values of $\mathbf{D}$ are related to those of $\mathbf{A}$. We establish this relation by stating a theorem from [83].

Theorem 5.1 *(Thompson [83]) Let the matrix* $\mathbf{A}$ *have an SVD as defined in* (2-1)*, and* $\mathbf{D} \in \mathbb{R}^{\ell\times\ell}$ *be any submatrix of* $\mathbf{A}$*. Then for* $j = 1, ..., \ell$*, we have*

$$\sigma_{j+1} \leq \sigma_j(\mathbf{D}) \leq \sigma_j. \tag{5-11}$$

To prove (5-11), it only suffices to allow $\mathbf{D}$ be $\mathbf{D} = \mathbf{Q}_1^T\mathbf{A}\mathbf{Q}_2$, where $\mathbf{Q}_1 \in \mathbb{R}^{m\times\ell}$ and $\mathbf{Q}_2 \in \mathbb{R}^{n\times\ell}$ are orthonormal matrices.

Thus, we will have

$$\begin{aligned} &\sigma_{\min}(\widetilde{\mathbf{R}}_{11}) \leq \sigma_k(\mathbf{D}) \leq \sigma_k, \\ &\|\widetilde{\mathbf{R}}_{22}\|_2 \leq \sigma_{k+1}(\mathbf{D}) \leq \sigma_{k+1}. \end{aligned} \tag{5-12}$$

Now, we furnish the relation of $\sigma_k(\mathbf{D})$ and the standard Gaussian matrix $\mathbf{\Psi}$. To this end, first, suppose that the sample size parameter ℓ satisfies

$$2 \leq p + k \leq \ell \tag{5-13}$$

where p is called an oversampling parameter [36, 39]. Since $\mathbf{\Psi}$ has interaction with the right singular vectors $\mathbf{V}$ of $\mathbf{A}$, i.e., equation (5-2), we have

$$\widetilde{\mathbf{\Psi}} = \mathbf{V}_{\mathrm{A}}^T\mathbf{\Psi} = [\widetilde{\mathbf{\Psi}}_1^T \quad \widetilde{\mathbf{\Psi}}_2^T]^T \tag{5-14}$$

where $\widetilde{\mathbf{\Psi}}_1$ and $\widetilde{\mathbf{\Psi}}_2$ have $\ell - p$ and $n - \ell + p$ rows, respectively. The following theorem, taken from [54], bounds $\sigma_k(\mathbf{D})$.

Theorem 5.2 *Suppose that the matrix* $\mathbf{A}$ *has an SVD defined in* (2-1), $2 \leq p+k \leq \ell$, *and the matrix* $\mathbf{D}$ *is formed through step 1 to step 5 of Algorithm 9. Moreover, assume that* $\widetilde{\mathbf{\Psi}}_1$ *is full row rank, then we have*

$$\sigma_k \geq \sigma_k(\mathbf{D}) \geq \frac{\sigma_k}{\sqrt{1 + \|\widetilde{\mathbf{\Psi}}_2\|_2^2 \|\widetilde{\mathbf{\Psi}}_1^\dagger\|_2^2 \left(\frac{\sigma_{\ell-p+1}}{\sigma_k}\right)^4}}, \tag{5-15}$$

and when the matrix $\mathbf{D}$ *is formed through step 1 to step 7 of Algorithm 10, i.e., the power method is used, we have*

$$\sigma_k \geq \sigma_k(\mathbf{D}) \geq \frac{\sigma_k}{\sqrt{1 + \|\widetilde{\mathbf{\Psi}}_2\|_2^2 \|\widetilde{\mathbf{\Psi}}_1^\dagger\|_2^2 \left(\frac{\sigma_{\ell-p+1}}{\sigma_k}\right)^{4q+4}}}. \tag{5-16}$$

Finally, since the random matrix $\mathbf{\Psi}$ has the standard Gaussian distribution, the average-case lower bound on the k-th singular value of CoR-UTV is given in the following theorem, taken from [54].

Theorem 5.3 *With the notation of Theorem 5.2, and* $\gamma_k = \frac{\sigma_{\ell-p+1}}{\sigma_k}$, *for Algorithm 9, we have*

$$\mathbb{E}(\sigma_k(\mathbf{D})) \geq \frac{\sigma_k}{\sqrt{1 + \nu^2 \gamma_k^4}}, \tag{5-17}$$

and when the power method is used, Algorithm 10, we have

$$\mathbb{E}(\sigma_k(\mathbf{D})) \geq \frac{\sigma_k}{\sqrt{1 + \nu^2 \gamma_k^{4q+4}}}, \tag{5-18}$$

where $\nu = \nu_1 \nu_2$, $\nu_1 = \sqrt{n - \ell + p} + \sqrt{\ell} + 7$, *and* $\nu_2 = \frac{4e\sqrt{\ell}}{p+1}$.

This completes the discussion on the rank-revealing property of the CoR-UTV algorithm.

5.2.2 Low-Rank Approximation

CoR-UTV efficiently constructs an accurate low-rank approximation to an input matrix $\mathbf{A}$. We provide theoretical guarantees on the accuracy of these approximations in terms of the Frobenius and spectral norm. To this end, we first state a theorem from [54].

Theorem 5.4 *Let the matrix* $\mathbf{A}$ *have an SVD as defined in* (2-1), *and* $\mathbf{Q}_1 \in \mathbb{R}^{m \times \ell}$ *and* $\mathbf{Q}_2 \in \mathbb{R}^{n \times \ell}$ *be matrices with orthonormal columns constructed by means of CoR-UTV, where* $1 \leq k \leq \ell$. *Let, furthermore,* $\mathbf{D}_k$ *be the best rank-k of* $\mathbf{D} = \mathbf{Q}_1^T \mathbf{A} \mathbf{Q}_2$. *Then, we have*

$$\|\mathbf{A} - \mathbf{Q}_1\mathbf{D}_k\mathbf{Q}_2^T\|_F \leq \|\mathbf{A}_0\|_F + \|\mathbf{A}_k - \mathbf{Q}_1\mathbf{Q}_1^T\mathbf{A}_k\|_F + \|\mathbf{A}_k - \mathbf{A}_k\mathbf{Q}_2\mathbf{Q}_2^T\|_F, \tag{5-19}$$

and

$$\|\mathbf{A} - \mathbf{Q}_1\mathbf{D}_k\mathbf{Q}_2^T\|_2 \leq \|\mathbf{A}_0\|_2 + \|\mathbf{A}_k - \mathbf{Q}_1\mathbf{Q}_1^T\mathbf{A}_k\|_F + \|\mathbf{A}_k - \mathbf{A}_k\mathbf{Q}_2\mathbf{Q}_2^T\|_F. \tag{5-20}$$

Now, we rewrite the CoR-UTV-based low-rank approximation, equation (5-7), as follows:

$$\hat{\mathbf{A}}_{\text{CoR}} = \mathbf{Q}_1\mathbf{D}\mathbf{Q}_2^T. \tag{5-21}$$

This perfectly makes sense since the column-pivoted QR decomposition, which factors $\mathbf{D}$, is a numerically stable deterministic method [35]. Thus, for $\xi = 2, F$, it follows that

$$\|\mathbf{A} - \hat{\mathbf{A}}_{\text{CoR}}\|_\xi \leq \|\mathbf{A} - \mathbf{Q}_1\mathbf{D}_k\mathbf{Q}_2^T\|_\xi. \tag{5-22}$$

This relation holds because $\mathbf{D}_k$ is the rank-k approximation of $\mathbf{D}$.

Theorem 5.5 *With the notation of Theorem 5.2, for $\xi = 2, F$, we have*

$$\|\mathbf{A} - \hat{\mathbf{A}}_{CoR}\|_\xi \leq \|\mathbf{A}_0\|_\xi + \|\mathbf{A}_k - \mathbf{Q}_1\mathbf{Q}_1^T\mathbf{A}_k\|_F + \|\mathbf{A}_k - \mathbf{A}_k\mathbf{Q}_2\mathbf{Q}_2^T\|_F. \tag{5-23}$$

Having stated the connection between CoR-UTV and SOR-SVD, we can obtain upper bounds for the CoR-UTV-based low-rank approximation.

Theorem 5.6 *Let the matrix $\mathbf{A}$ have an SVD as defined in* (2-1)*, $2 \leq p+k \leq \ell$, and $\hat{\mathbf{A}}_{CoR}$ is computed through the basic version of CoR-UTV, Algorithm 9. Furthermore, assume that $\widetilde{\mathbf{\Psi}}_1$ is full row rank. Then, for $\xi = 2, F$, we have*

$$\|\mathbf{A} - \hat{\mathbf{A}}_{CoR}\|_\xi \leq \|\mathbf{A}_0\|_\xi + \sqrt{\frac{\alpha^2\|\widetilde{\mathbf{\Psi}}_2\|_2^2\|\widetilde{\mathbf{\Psi}}_1^\dagger\|_2^2}{1+\beta^2\|\widetilde{\mathbf{\Psi}}_2\|_2^2\|\widetilde{\mathbf{\Psi}}_1^\dagger\|_2^2}} + \sqrt{\frac{\eta^2\|\widetilde{\mathbf{\Psi}}_2\|_2^2\|\widetilde{\mathbf{\Psi}}_1^\dagger\|_2^2}{1+\tau^2\|\widetilde{\mathbf{\Psi}}_2\|_2^2\|\widetilde{\mathbf{\Psi}}_1^\dagger\|_2^2}}, \tag{5-24}$$

where $\alpha = \sqrt{k}\frac{\sigma_{\ell-p+1}^2}{\sigma_k}$, $\beta = \frac{\sigma_{\ell-p+1}^2}{\sigma_1\sigma_k}$, $\eta = \sqrt{k}\sigma_{\ell-p+1}$ and $\tau = \frac{\sigma_{\ell-p+1}}{\sigma_1}$.

When the power iteration is used, Algorithm 10, $\alpha = \sqrt{k}\frac{\sigma_{\ell-p+1}^2}{\sigma_k}\left(\frac{\sigma_{\ell-p+1}}{\sigma_k}\right)^{2q}$, $\beta = \frac{\sigma_{\ell-p+1}^2}{\sigma_1\sigma_k}\left(\frac{\sigma_{\ell-p+1}}{\sigma_k}\right)^{2q}$, $\eta = \dfrac{\sigma_k}{\sigma_{\ell-p+1}}\alpha$ and $\tau = \dfrac{1}{\sigma_{\ell-p+1}}\beta$.

Theorem 5.6 implies that, at least, the bounds for SOR-SVD hold for CoR-UTV. For a detailed error analysis of the SOR-SVD algorithm, see [54].

The random matrix $\mathbf{\Psi}$ has the standard Gaussian distribution, we thus present a theorem that establishes average error bounds on the CoR-UTV-based low-rank approximation.

Theorem 5.7 *With the notation of Theorem 5.6, and $\gamma_k = \frac{\sigma_{\ell-p+1}}{\sigma_k}$, for the basic version of CoR-UTV, Algorithm 9, we have*

$$\mathbb{E}\|\mathbf{A} - \hat{\mathbf{A}}_{CoR}\|_\xi \leq \ \|\mathbf{A}_0\|_\xi + (1+\gamma_k)\sqrt{k}\nu\sigma_{\ell-p+1}, \tag{5-25}$$

and when the power method is used, Alg. 10, we have

$$\mathbb{E}\|\mathbf{A} - \hat{\mathbf{A}}_{CoR}\|_\xi \leq \ \|\mathbf{A}_0\|_\xi + (1+\gamma_k)\sqrt{k}\nu\sigma_{\ell-p+1}\gamma_k^{2q}, \tag{5-26}$$

where ν is defined in Theorem 5.3.

5.2.3 Computational Complexity

CoR-UTV in order to decompose the matrix $\mathbf{A}$, where $\mathbf{A}$ is stored out-of-core, only requires either three passes or $2q+3$ passes (when the power method is employed) over data with the following operation count:

$$C_{\text{CoR-UTV}} \sim (2q+3)\ell C_{\text{mult}} + 6\ell^2 + 2n\ell(\ell+1) + \frac{8}{3}\ell^3, \tag{5-27}$$

where C_{mult} is the cost of a matrix-vector multiplication with $\mathbf{A}$ or $\mathbf{A}^T$, and the cost of the QR and QRCP, for instance, for $\mathbf{A}$, are $2mn^2$ and $\frac{8}{3}mn^2$ flops, respectively [35, 40]. For the case where the compressed matrix $\mathbf{D}$ is approximated by $\mathbf{D}_{\text{approx}}$, CoR-UTV requires either two or $2q+2$ passes (when the power method is used) over data, and the flop count satisfies

$$C_{\text{CoR-UTV}} \sim (2q+2)\ell C_{\text{mult}} + 6\ell^2 + 2n\ell(2\ell+1) + \frac{17}{3}\ell^3. \tag{5-28}$$

The CoR-UTV algorithm, except for matrix-matrix multiplications, which are readily parallelizable, performs two QR decompositions on matrices of size $m \times \ell$ and $n \times \ell$, and one QRCP on an $\ell \times \ell$ matrix. TSR-SVD and SOR-SVD, however, perform an SVD on the $\ell \times \ell$ matrix, which is more expensive than a QRCP. Furthermore, recently developed column-pivoted QR algorithms, based on randomization, can perform the factorization with optimal/minimum communication costs [20, 21, 26, 62], while standard techniques to compute an SVD are challenging for parallelization [18, 36, 39]. As a result, for very large matrices to be factored on advanced computational platforms, where the compressed $\ell \times \ell$ matrix does not fit into fast memory, the execution time to computing CoR-UTV can be substantially less than those of TSR-SVD and SOR-SVD. See [20, 21] for a comprehensive discussion on communication cost.

5.2.4 Robust PCA Using CoR-UTV

We now apply CoR-UTV to solve the robust PCA problem [9,14,93]; We retain the original objective function proposed in [9,14,59,93], and apply CoR-

UTV as a surrogate to the SVD to solve the optimization problem (2-26). We adopt the continuation technique [59, 85], which increases μ in each iteration. The pseudocode of the proposed method, called `ALM-CoRUTV` hereafter, is given in Table 5.1.

Table 5.1: Pseudo-code of robust PCA solved by `ALM-CoRUTV`.

Input: Matrix $\mathbf{M}, \lambda, \mu_0, \bar{\mu}, \rho, \mathbf{Y}_0, \mathbf{S}_0, j = 0$;
Output: Low-rank plus sparse matrix
1: **while** the algorithm does not converge **do**
2: Compute $\mathbf{L}_{j+1} = \mathcal{C}_{\mu_j^{-1}}(\mathbf{M} - \mathbf{S}_j + \mu_j^{-1}\mathbf{Y}_j)$;
3: Compute $\mathbf{S}_{j+1} = \mathcal{S}_{\lambda\mu_j^{-1}}(\mathbf{M} - \mathbf{L}_{j+1} + \mu_j^{-1}\mathbf{Y})$;
4: Compute $\mathbf{Y}_{j+1} = \mathbf{Y}_j + \mu_j(\mathbf{M} - \mathbf{L}_{j+1} - \mathbf{S}_{j+1})$;
5: Update $\mu_{j+1} = \max(\rho\mu_j, \bar{\mu})$;
6: **end while**
7: **return** $\mathbf{L}^*$ and $\mathbf{S}^*$

In Table 5.1, for any matrix $\mathbf{B}$ having a CoR-UTV decomposition described in Section 5.1, $\mathcal{C}_\delta(\mathbf{B})$ refers to a CoR-UTV thresholding operator defined as:

$$\mathcal{C}_\delta(\mathbf{B}) = \mathbf{U}(:, 1:r)\mathbf{T}(1:r, :)\mathbf{V}^T, \tag{5-29}$$

where r is the number of diagonals of $\mathbf{T}$ greater than δ, and λ, μ_0, $\bar{\mu}$, ρ, $\mathbf{Y}_0$, and $\mathbf{S}_0$ are initial values. The main operation of the `ALM-CoRUTV` algorithm is computing CoR-UTV in each iteration, which is efficient in terms of flops, $O(mnk)$, and can be computed with minimum communication costs; see subsection 5.2.3. In subsection 5.3.4, we experimentally verify that `ALM-CoRUTV` converges to the exact optimal solution.

5.3 Numerical Experiments

In this section, we present the results of some numerical experiments conducted to evaluate the performance of the CoR-UTV algorithm for approximating a low-rank input matrix. We show that CoR-UTV provides highly accurate singular values and low-rank approximations, and compare our algorithm against several other algorithms from the literature. We furthermore employ CoR-UTV for solving the robust PCA problem.

5.3.1 Comparison of Rank-revealing Property and Singular Values

We first show that CoR-UTV (i) is a rank revealer, i.e., the gap in the singular values of the input matrix is revealed, and (ii) provides highly accurate

singular values that with remarkable fidelity track singular values of the matrix. For the sake of simplicity, we focus on square matrices, and experiments are implemented in MATLAB.

We construct two types of matrices of order 10^3:

- Matrix 1 (Noisy Low-Rank). This matrix is formed by a linear superposition of two matrices $\mathbf{A} = \mathbf{A}_1 + \mathbf{E}$. $\mathbf{A}_1 = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$, where $\mathbf{U}$ and $\mathbf{V}$ are random orthonormal matrices, $\mathbf{\Sigma}$ is a diagonal matrix containing the singular values σ_is that decrease linearly from 1 to 10^{-9}, and $\sigma_j = 0$ for $j = k+1, ..., 10^3$. The matrix $\mathbf{E}$ is a Gaussian matrix normalized to have ℓ_2-norm gap $\times\ \sigma_k$. In MATLAB notation, `smax = 1; smin = 1e-9; s = linspace(smax,smin,n); s(k+1:n) = 0; G = randn(n); G = G/norm(G); A = orth(rand(n)) * diag(s) * orth(rand(n)) + gap * s(k) * G.` We set the numerical rank $k = 20$, and consider two cases:
  - gap = 0.01; `NoisyLowRank-I`
  - gap = 0.1; `NoisyLowRank-II`
- Matrix 2 (Fast Decay). This matrix is formed in a similar way as $\mathbf{A}_1$ of Matrix 1, but now the diagonals of $\mathbf{\Sigma}$ take a different form such that $\sigma_j = 1$ for $j = 1, ..., k$, and $\sigma_j = (j - k + 1)^{-2}$ for $j = k+1, ..., 10^3$ [86]. We set the numerical rank $k = 10$.

We compare the quality of singular values computed by our method, described in Section 5.1, against that of alternative rank-revealing methods such as the SVD, QR with column pivoting (QRCP), UTV, and TSR-SVD.

For CoR-UTV and TSR-SVD, we set the sample size parameter $\ell = 2k$, chosen randomly. Both algorithms require the same number of passes over $\mathbf{A}$, either two or $2q + 2$ when the power method is used, to perform the factorization. To compute a UTV decomposition, we implement the `lurv` function from [32].

The results are shown in Figures 5.1−5.3. We make the following observations:

- For all matrices (`NoisyLowRank-I`, `NoisyLowRank-II`, Matrix 2), CoR-UTV strongly reveals the numerical rank k, as do the SVD, UTV and TSR-SVD, while QRCP weakly reveals the rank of `NoisyLowRank-I` and Matrix 2, and only suggests the gap in the singular values of `NoisyLowRank-II`.

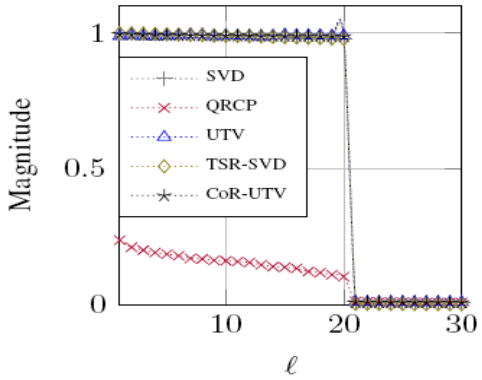


Figure 5.1: Comparison of singular values for `NoisyLowRank-I`. The power method is not used, $q = 0$.

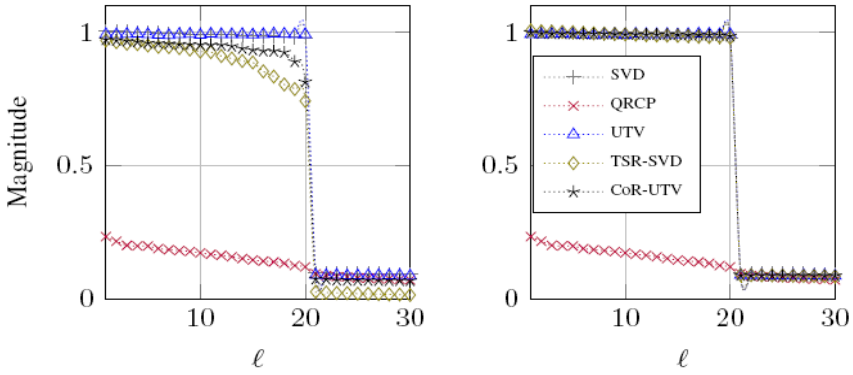


Figure 5.2: Comparison of singular values for `NoisyLowRank-II`. Left: No power method, $q = 0$. Right: $q = 2$.

– CoR-UTV, without making use of the power iteration scheme, i.e., $q = 0$, provides an excellent approximation to singular values for `NoisyLowRank-I` and Matrix 2. For `NoisyLowRank-II`, CoR-UTV outperforms TSR-SVD when $q = 0$, in approximating both leading and trailing singular values, and it only requires two steps of the power iteration to deliver singular values as accurate as the optimal SVD. The QRCP algorithm, however, gives a fuzzy approximation to singular values of the matrices considered.

5.3.2 Comparison of Low-Rank Approximation

1. Rank-ℓ Approximation. Since CoR-UTV computes a rank-ℓ approximation of a given matrix, we first investigate how accurate this approximation is. To this end, we compute a rank-ℓ approximation $\hat{\mathbf{A}}_{\text{CoR}}$ for `NoisyLowRank-I`, `NoisyLowRank-II`, and Matrix 2 using Algorithms 9 and 10 for each sample

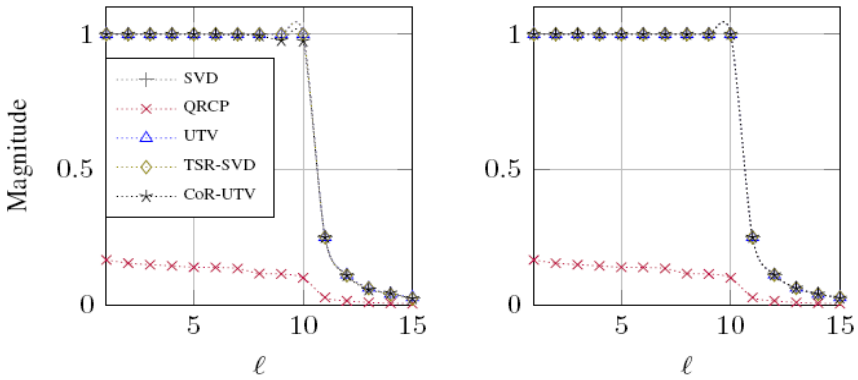


Figure 5.3: Comparison of singular values for Matrix 2. Left: No power method, $q = 0$. Right: $q = 2$.

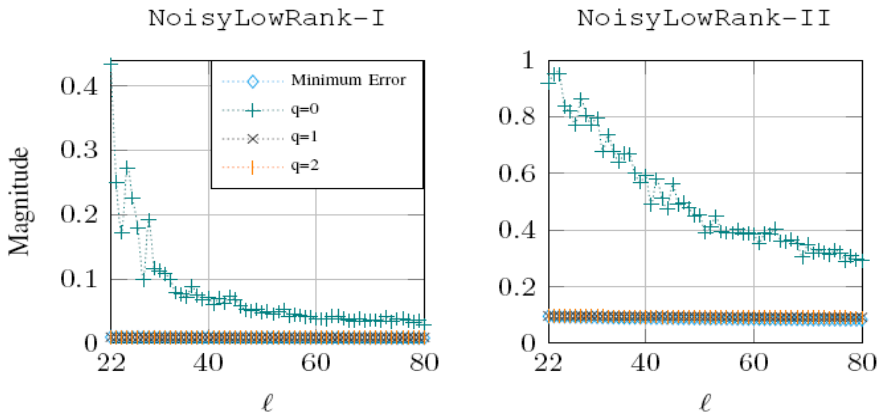


Figure 5.4: Comparison of low-rank approximation errors of the SVD and CoR-UTV for Matrix 1.

size parameter ℓ, and calculate the approximation error as:

$$e_\ell = \|\mathbf{A} - \hat{\mathbf{A}}_{\text{CoR}}\|_2. \tag{5-30}$$

We compare the approximation errors (5-30) against those produced by the rank-ℓ approximations using the SVD, i.e., minimal error $\sigma_{\ell+1}$. The results are shown in Figures 5.4 and 5.5.

Judging from the figures, (i) when $q = 0$, which corresponds to the basic version of CoR-UTV (Algorithm 9), the approximation is rather poor. (ii) The error incurred by Alg. 9 produces an upper bound for the minimal error. (iii) With only one step of the power iteration $q = 1$, Alg. 10, the accuracy of the approximation substantially improves, resulting in an approximation as accurate as the optimal SVD for all three matrices.

2. Rank-k Approximation. We now compare the low-rank approximations constructed by our method against those of the SVD, QRCP, TSR-SVD, and SOR-SVD. We have excluded the UTV algorithm because it has, by

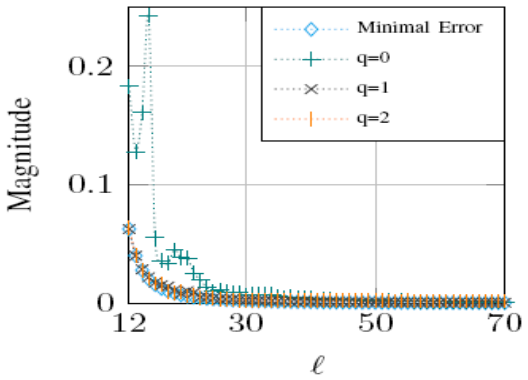


Figure 5.5: Comparison of low-rank approximation errors of the SVD and CoR-UTV for Matrix 2.

far, the worst performance among the algorithms discussed. This allows us to display the behavior of other algorithms clearly in the graphs. To make a fair comparison, we construct a rank-k approximation $\hat{\mathbf{A}}_{\text{out}}$ to $\mathbf{A}$ by each algorithm, and calculate the error:

$$e_k = \|\mathbf{A} - \hat{\mathbf{A}}_{\text{out}}\|_\xi, \tag{5-31}$$

where $\xi = F$ for the Frobenius-norm error, and $\xi = 2$ for the spectral-norm error. For the randomized algorithms, TSR-SVD, SOR-SVD, CoR-UTV, we construct a rank-k approximation by varying the sample size parameter ℓ, since, as shown, this parameter colors the quality of approximations. The rank-k approximation by TSR-SVD is constructed by selecting the first k approximate singular vectors and corresponding singular values. SOR-SVD constructs a rank-k approximation of an input matrix, and the rank-k approximation by CoR-UTV is computed as follows:

$$\hat{\mathbf{A}}_{\text{CoR}-k} = \mathbf{U}(:, 1:k)\mathbf{T}(1:k, :)\mathbf{V}^T. \tag{5-32}$$

For the randomized algorithms, we run the experiment with zero step of the power method ($q = 0$), and $q = 2$. The results are shown in Figures 5.6-5.8. (The results for the spectral-norm error are shown in the appendix). We make two observations: (1) For matrices `NoisyLowRank-I` and `NoisyLowRank-II`, when $q = 0$, as the number of samples increases the performance of CoR-UTV exceeds that of QRCP, becoming close to optimal performance of the SVD. In this case, CoR-UTV and SOR-SVD show similar performances, exceeding that of TSR-SVD. (2) When $q = 2$, the errors resulting from CoR-UTV show no loss of accuracy compared to the optimal SVD. In this case, QRCP has the poorest performance for all examples.

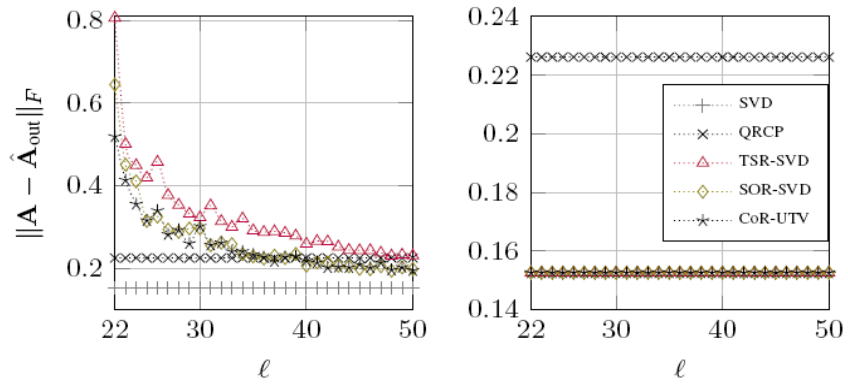


Figure 5.6: Comparison of the Frobenius-norm error for `NoisyLowRank-I`. Left: No power method, $q = 0$. Right: $q = 2$.

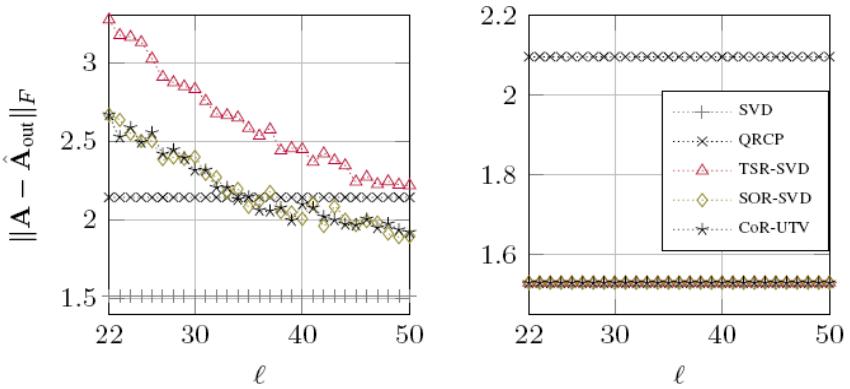


Figure 5.7: Comparison of the Frobenius-norm error for `NoisyLowRank-II`. Left: No power method, $q = 0$. Right: $q = 2$.

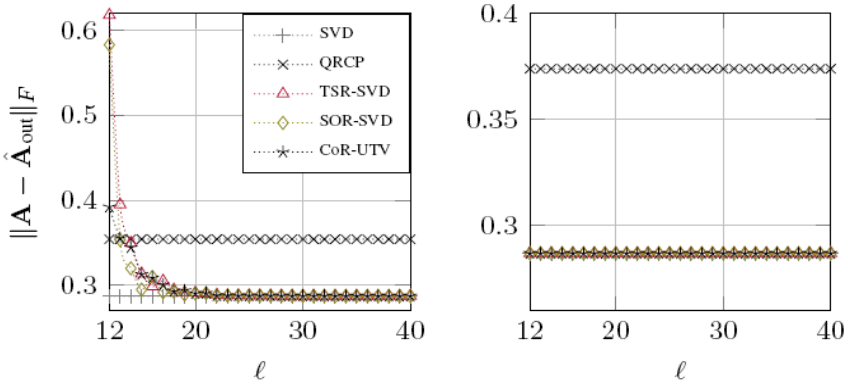


Figure 5.8: Comparison of the Frobenius-norm error for Matrix 2. Left: No power method, $q = 0$. Right: $q = 2$.

5.3.3 Image Reconstruction

We assess the quality of low-rank approximation by reconstructing a gray-scale image of a differential gear of size 1280×804, taken from [26], using CoR-UTV, truncated QRCP, and the truncated SVD by using PROPACK package [57]. This set of experiments were run in MATLAB on a desktop PC with a 3 GHz intel Core i5-4430 processor and 8 GB of memory.

The results are shown in Figures 5.9 and 5.10; Figure 5.9a displays the Frobenius-norm approximation error against the corresponding approximation rank, where the error is calculated as:

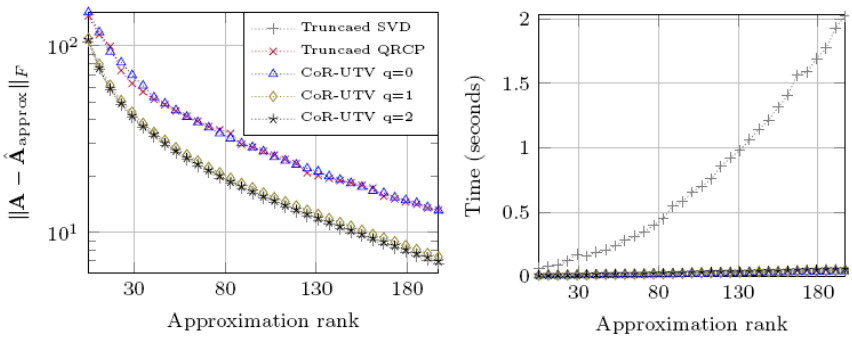
$$e_{\text{approx}} = \|\mathbf{A} - \hat{\mathbf{A}}_{\text{approx}}\|_F, \tag{5-33}$$

where $\hat{\mathbf{A}}_{\text{approx}}$ is the approximation computed by each algorithm, and Figure 5.9 shows the reconstructed images of the differential gear with $rank = 20$ and $rank = 90$ using the algorithms mentioned.

Judging from the figures, for the rank-20 approximation, truncated QRCP and CoR-UTV without power iteration technique produce the poorest reconstruction qualities. CoR-UTV with one step of power iteration produces a better result. Truncated SVD and CoR-UTV with two steps of power iteration, however, appear to have reconstructed images that are visually identical. For the rank-90 approximation, with a careful scrutiny, fine defects appear in reconstructions by truncated QRCP and CoR-UTV with $q = 0$, while reconstructed images by truncated SVD, CoR-UTV with $q = 1$ and $q = 2$ are visually indistinguishable from the original.

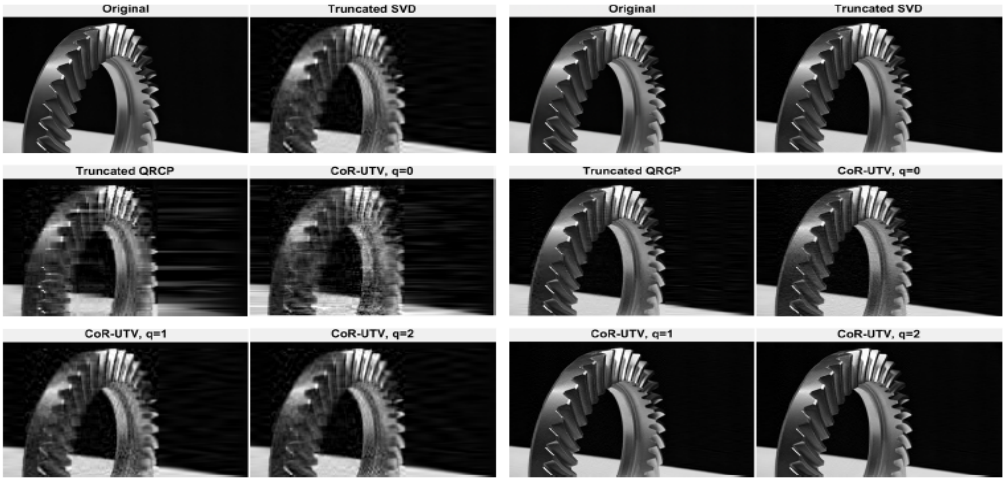
Figure 5.9b compares the runtime of CoR-UTV and the truncated SVD against the corresponding approximation rank for the reconstruction scenario. We have excluded truncated QRCP because there is no optimized LAPACK function for QRCP with a specified rank. Figure 5.9b illustrates how the execution time for truncated SVD substantially grows as the approximation rank increases. While, even two steps of the power method hardly adds to the execution time of more efficient CoR-UTV. This, taken together with reconstructions of Figure 5.10, shows that CoR-UTV produces comparable results with truncated SVD at a much lower cost. We expect, however, that since CoR-UTV can be performed using operations with optimal communication cost, on current and future advanced computers CoR-UTV to be faster, where communication cost is a major bottleneck on the performance of an algorithm.

In order to illustrate the computational time of CoR-UTV and TSR-SVD, randomized algorithms that produce a rank-ℓ approximation, we provide another figure, Figure 5.11, which compares the runtime of algorithms against



(a) Approximation error (b) Computational time

Figure 5.9: (a) Errors incurred by the algorithms considered in reconstructing the differential gear image. (b) Computational time in seconds for different algorithms.



(a) Reconstruction with $rank = 20$ (b) Reconstruction with $rank = 90$

Figure 5.10: Low-rank image reconstruction.

the corresponding approximation. As can be seen CoR-UTV shows a better performance as the approximation rank increases, and due to employing QRCP rather than an SVD, to approximate very large matrices, we expect CoR-UTV to be faster on advanced computers.

5.3.4 Robust PCA

In this subsection, we experimentally investigate the efficiency and efficacy of `ALM-CoRUTV`, described in Table 5.1, in recovering the low-rank and sparse components of synthetic and real data. We compare the results obtained with those of the efficiently implemented inexact ALM method by [59], called `InexactALM` hereafter.

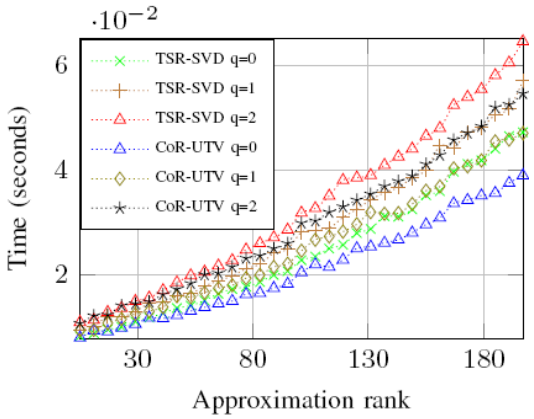


Figure 5.11: Runtime comparison of TSR-SVD and CoR-UTV in reconstructing the differential gear image.

5.3.4.1 Recovery of Synthetic Matrix

We generate rank-k $\mathbf{X} = \mathbf{L} + \mathbf{S}$ as described in Section 4.3.3.1. The only difference is that the non-zero entries of $\mathbf{S}$ are drawn from the set {-80, 80} instead of {-50, 50}. We apply the `ALM-CoRUTV` and `InexactALM` algorithms to $\mathbf{X}$ to recover $\mathbf{L}$ and $\mathbf{S}$. The numerical results are summarized in Tables 5.2 and 5.3. Both algorithms are terminated when the following stopping condition holds:

$$\frac{\|\mathbf{X} - \mathbf{L}^{out} - \mathbf{S}^{out}\|_F}{\|\mathbf{X}\|_F} < 10^{-5}, \tag{5-34}$$

where $(\mathbf{L}^{out}, \mathbf{S}^{out})$ is the pair of output of either algorithm.

Table 5.2: Comparison of the `ALM-CoRUTV` and `ALM-CoRUTV` methods for synthetic data recovery for the case $r(\mathbf{L}) = 0.05 \times n$ and $s = 0.05 \times n^2$.

n	$r(\mathbf{L})$	$\|\mathbf{S}\|_0$	Methods	$r(\hat{\mathbf{L}})$	$\|\hat{\mathbf{S}}\|_0$	Time	Iter.	ξ
1000	50	5e4	`ALM-CoRUTV`	50	5e4	0.6	12	9.6e-6
			`InexactALM`	50	5e4	4.1	12	2.1e-6
2000	100	2e5	`ALM-CoRUTV`	100	2e5	3.7	12	8.3e-6
			`InexactALM`	100	2e5	27.4	12	2.7e-6
3000	150	45e5	`ALM-CoRUTV`	150	45e5	9.4	12	8.7e-6
			`InexactALM`	150	45e5	75.6	12	3.1e-6
4000	200	8e5	`ALM-CoRUTV`	200	8e5	20	12	8.1e-6
			`InexactALM`	200	8e5	173.3	12	3.5e-6

Table 5.3: Comparison of the `ALM-CoRUTV` and `InexactALM` methods for synthetic data recovery for the case $r(\mathbf{L}) = 0.05 \times n$ and $s = 0.1 \times n^2$.

n	$r(\mathbf{L})$	$\|\mathbf{S}\|_0$	Methods	$r(\hat{\mathbf{L}})$	$\|\hat{\mathbf{S}}\|_0$	Time	Iter.	ξ
1000	50	1e5	`ALM-CoRUTV`	50	1e5	0.7	14	9.1e-6
			`InexactALM`	50	1e5	4.5	13	4.4e-6
2000	100	4e5	`ALM-CoRUTV`	100	4e5	4.1	14	8.9e-6
			`InexactALM`	100	4e5	29.2	13	5.5e-6
3000	150	9e5	`ALM-CoRUTV`	150	9e5	10.9	14	9.3e-6
			`InexactALM`	150	9e5	83.9	13	6.8e-6
4000	200	16e5	`ALM-CoRUTV`	200	16e5	23.2	14	9.5e-6
			`InexactALM`	200	16e5	189.4	13	7.8e-6

CoR-UTV requires a prespecified rank ℓ to perform the factorization. Thus, we set $\ell = 2k$, as a random start, and $q = 1$ (one step of a power iteration). Judging from the results in Tables 5.2 and 5.3, we make several observations on `ALM-CoRUTV`:

- It successfully detects the exact numerical rank k of the input matrix in all cases.
- It provides the exact optimal solution, having the same number of iterations for the first test case, while it requires one more iteration for the second challenging test case, compared to `InexactALM`.
- It outperforms `InexactALM` in terms of runtime, with speedups of up to 8.6 times.

In summary, `ALM-CoRUTV` exactly recovers the low-rank and sparse matrices from a grossly corrupted matrix at a much lower cost compared to `InexactALM`. However, we expect `ALM-CoRUTV` to be faster on multicore and accelerator-based computers, since CoR-UTV can be computed with minimum communication cost.

5.3.4.2 Background Modeling in Surveillance Video

We apply `ALM-CoRUTV` to two surveillance videos introduced in [58] (The first and third video described in Section 4.3.3.2). In order for CoR-UTV, used in `ALM-CoRUTV`, to approximate the low-rank component of real data, we determine the prespecified rank ℓ by making use of the bound in (3-16). We

set $\ell = k + p$, where k is the minimum value satisfying (3-16), and $p = 2$ is an oversampling parameter. Again, we set $q = 1$ for CoR-UTV.

Some frames of the surveillance videos with recovered foregrounds and backgrounds are displayed in Figure 5.12. We only show the results of `ALM-CoRUTV` since those produced by `InexactALM` are visually identical. As can be seen the proposed `ALM-CoRUTV` can successfully recover the low-rank and sparse components of the videos. Table 5.4 presents the numerical results. In both examples, `ALM-CoRUTV` outperforms `InexactALM` in terms of runtime.

Figure 5.12: Background modeling. Images in columns 1 and 4 are frames of the surveillance video of an airport and a escalator, respectively. Images in columns 2 and 5 are recovered backgrounds $\mathbf{L}^*$, and columns 3 and 6 correspond to foregrounds $\mathbf{S}^*$ by `ALM-CoRUTV`.

Table 5.4: Numerical results for real matrix recovery.

Dataset	`InexactALM`			`ALM-CoRUTV`		
	Time	Iter.	ξ	Time	Iter.	ξ
Airport hall 25344×200	15.4	28	7.4e-6	5.1	28	7.1e-6
Escalator 20800×200	11.9	28	8.5e-6	4.2	28	6.2e-6
Yale B01 32256×64	4.2	26	7.6e-6	2.1	26	8.6e-6
Yale B02 32256×64	4.2	26	6.5e-6	2.1	26	8.3e-6

5.3.4.3 Shadow and Specularity Removal From Face Images

In this experiment, we use face images taken from the Yale B face database [34]. The recovered images are displayed in Figure 5.13. From this figure, we observe that the shadows and specularities have been effectively

Figure 5.13: Removing shadows and specularities from face images. Images in columns 1 and 4 are face images under different illuminations. Images in columns 2 and 5 are are recovered images after removing shadows and specularities by the ALM-SOR-SVD method, and images in columns 3 and 6 correspond to the removed shadows and specularities.

extracted in the sparse components by `ALM-CoRUTV`. Table 5.4 summarizes the numerical results.

We conclude that `ALM-CoRUTV` can successfully recover the face images under different illuminations from the dataset studied two times faster than `InexactALM`.

6
Randomized Subspace Methods

This chapter develops two novel subspace methods using randomization collectively called randomized subspace methods (RSMs) to detect anomalies in Internet Protocol (IP) networks [52]. We first describe the PCA-based subspace method and its application for network anomaly detection. Next, we present our work.

The PCA-based subspace method, discussed in Section 6.2, focuses on the link traffic covariance matrix to detect network anomalies; first, the covariance matrix is formed. Second, its SVD is computed. Third, the normal and anomalous subspaces of the traffic data are separated using a specific number of singular vectors. Finally, a statistical test, the Q-statistic, is applied to diagnose anomalies in the anomalous subspace. The randomized subspace methods, described in Section 6.3, on the other hand, do not form the covariance matrix of the link traffic; using randomized sampling techniques, they obtain approximate bases for the range of the traffic data and, accordingly separate the normal and anomalous subspaces of the data by using a specific number of these bases. The variances captured by the bases are computed, and the Q-statistic follows to detect anomalies in the anomalous subspaces.

6.1
Data Model

Based on the structure of a network and the flow of data obtained by network tomography [90], we can model the link traffic as a function of the origin-destination (OD) flow traffic and the network-specific routing. Specifically, the relationship between the link traffic $\mathbf{Y} \in \mathbb{R}^{m \times t}$ and OD flow traffic $\mathbf{X} \in \mathbb{R}^{n \times t}$, for a network with m links and n OD flows may be written as

$$\mathbf{Y} = \mathbf{R}\mathbf{X}, \tag{6-1}$$

where t is the number of snapshots and $\mathbf{R} \in \mathbb{R}^{m \times n}$ is a routing matrix. The entries of $\mathbf{R}$, i.e., $\mathbf{R}_{i,j}$, are assigned a value equal to one ($\mathbf{R}_{i,j} = 1$) if the OD flow j traverses link i, and are assigned a value equal to zero otherwise.

The network traffic model that takes into account the traffic anomalies and the measurement noise over the links can be expressed by

$$\mathbf{Y} = \mathbf{R}(\mathbf{X} + \mathbf{A}) + \mathbf{V}, \tag{6-2}$$

where $\mathbf{R} \in \mathbb{R}^{m \times n}$ is a fixed routing matrix, $\mathbf{X} \in \mathbb{R}^{n \times t}$ is the clean traffic matrix, $\mathbf{A} \in \mathbb{R}^{n \times t}$ is the matrix with traffic anomalies and $\mathbf{V} \in \mathbb{R}^{m \times t}$ denotes the link measurement noise samples.

6.2 PCA-Based Subspace Method for Anomaly Detection

Subspace methods are powerful tools to decompose a given data matrix $\mathbf{Y}$ into two components such as $\mathbf{Y} = \hat{\mathbf{Y}} + \tilde{\mathbf{Y}}$, where, in signal processing terms, $\hat{\mathbf{Y}}$ and $\tilde{\mathbf{Y}}$ are referred to as the signal subspace and noise subspace, respectively [42, 75]. They have many applications in IP networks anomaly detection [56], statistical process control and multidimensional fault identification [27, 46], face recognition [70], and system identification [55].

The seminal paper by Lakhina et al. [56] was the first that used PCA-based subspace method to detect traffic anomalies in IP networks. Given a matrix of link traffic data $\mathbf{Y} \in \mathbb{R}^{m \times t}$, where $m \leq t$, the approach performs a normal-plus-anomalous matrix decomposition such that $\mathbf{Y} = \hat{\mathbf{Y}} + \tilde{\mathbf{Y}}$, where$\hat{\mathbf{Y}}$ is the modeled traffic and $\tilde{\mathbf{Y}}$ is the anomalous or residual traffic. Then, it seeks anomalies in the anomalous subspace $\tilde{\mathbf{Y}}$. The modeled traffic represented by $\hat{\mathbf{Y}}$ is the projection of $\mathbf{Y}$ onto the normal subspace $\mathcal{S}$, and the residual traffic modeled by $\tilde{\mathbf{Y}}$ is the projection of $\mathbf{Y}$ onto the anomalous subspace $\tilde{\mathcal{S}}$, both by making use of a selected number of principal components of $\mathbf{Y}$. To be specific, the modeled traffic is obtained by

$$\hat{\mathbf{Y}} = \mathbf{P}\mathbf{P}^T\mathbf{Y} = \hat{\mathbf{C}}\mathbf{Y}, \tag{6-3}$$

and the residual traffic is obtained by

$$\tilde{\mathbf{Y}} = (\mathbf{I} - \mathbf{P}\mathbf{P}^T)\mathbf{Y} = \tilde{\mathbf{C}}\mathbf{Y}, \tag{6-4}$$

where $\mathbf{P} = [\mathbf{w}_1, \mathbf{w}_2, ..., \mathbf{w}_r]$ is formed by the first r singular vectors $\mathbf{W}$ of the covariance of the centered traffic data $\hat{\mathbf{\Sigma}} = \frac{1}{t-1}(\mathbf{Y} - \mu)(\mathbf{Y} - \mu)^T$, where μ contains the mean of $\mathbf{Y}$, and $\hat{\mathbf{\Sigma}} = \mathbf{W}\mathbf{\Lambda}\mathbf{W}^T$ is a singular value decomposition.

Typically, a traffic anomaly results in a large change to the residual traffic $\tilde{\mathbf{Y}}$ [56]. To detect abnormal changes in $\tilde{\mathbf{Y}}$, a statistic referred to as the Q-statistic [47] is applied by computing the squared prediction error (SPE) of the residual traffic:

$$\text{SPE} = \|\tilde{\mathbf{Y}}\|_2^2 = \|\tilde{\mathbf{C}}\mathbf{Y}\|_2^2. \tag{6-5}$$

The network traffic is considered to be normal if

$$\text{SPE} \leq Q_\beta, \tag{6-6}$$

where Q_β is a threshold for the SPE defined as

$$Q_\beta = \theta_1 \Big[\frac{c_\beta \sqrt{2\theta_2 h_0^2}}{\theta_1} + 1 + \frac{\theta_2 h_0 (h_0 - 1)}{\theta_1^2} \Big]^{\frac{1}{h_0}}, \tag{6-7}$$

where

$$h_0 = 1 - \frac{2\theta_1 \theta_3}{3\theta_2^2}, \tag{6-8}$$

and

$$\theta_i = \sum_{j=r+1}^{m} \lambda_j^i, \text{for } i = 1, 2, 3, \tag{6-9}$$

with λ_j denoting the j-th singular value of $\hat{\mathbf{\Sigma}}$ and c_β is the $1 - \beta$ percentile in a standard normal distribution.

The singular vectors of $\hat{\mathbf{\Sigma}}$ (or principal components of $\mathbf{Y}$) maximize the variance of the projected data. Thus, the j-th singular value of $\hat{\mathbf{\Sigma}}$ (or the variance captured by the j-th PC) can be expressed as $\lambda_j = \mathbb{V}\text{ar}\{(\mathbf{w}_j{}^T \mathbf{Y})^T\}$ [50]. Note that, each column in $\mathbf{Y}$, $Y_i \in \mathbb{R}^m$.

6.3 Randomized Subspace Methods for Anomaly Detection

Randomized subspace methods (RSMs), namely Randomized Basis for Anomaly Detection (RBAD) and Switched Subspace-Projected Basis for Anomaly Detection (SSPBAD) [52], separate normal and anomalous subspaces of a traffic matrix $\mathbf{Y}$ by using the randomized sampling scheme [19, 39], and seek traffic anomalies in the anomalous subspace. In contrast to the works in [6, 43, 56], the RSMs do not form the covariance matrix from the traffic data and consequently obviate the computation of the SVD for the subspace separation. In RSMs first, a set of orthonormal basis $\mathbf{Q} \in \mathbb{R}^{m \times m}$ whose range approximates the range of $\mathbf{Y}$ is constructed using randomization (See Tables 6.1 and 6.2). Next, the normal and anomalous components of $\mathbf{Y}$ are formed as follows:

$$\hat{\mathbf{Y}} = \mathbf{P}\mathbf{P}^T\mathbf{Y} = \hat{\mathbf{C}}\mathbf{Y}, \tag{6-10}$$

and

$$\tilde{\mathbf{Y}} = (\mathbf{I} - \mathbf{P}\mathbf{P}^T)\mathbf{Y} = \tilde{\mathbf{C}}\mathbf{Y}, \tag{6-11}$$

where $\mathbf{P} = [\mathbf{q}_1, \mathbf{q}_2, ..., \mathbf{q}_r]$ contains the first r columns of $\mathbf{Q}$. In words, the modeled traffic $\hat{\mathbf{Y}}$ is the projection of $\mathbf{Y}$ onto the normal subspace $\mathcal{S}_Q$ spanned by $\{\mathbf{q}_1, ..., \mathbf{q}_r\}$, and the residual traffic $\tilde{\mathbf{Y}}$ is the projection of $\mathbf{Y}$ onto the anomalous subspace $\tilde{\mathcal{S}}_Q$, a subspace orthogonal to $\mathcal{S}_Q$, i.e., $\tilde{\mathcal{S}}_Q = \mathcal{S}_Q^\perp$. In order to detect abnormal behavior in the anomalous component using the Q-statistic [47], the variances captured by the orthonormal basis must be known. The variances are computed as follows

$$\mathbf{\Lambda_Q} = \mathbb{V}\text{ar}\{(\mathbf{Q}^T\mathbf{Y})^T\}. \tag{6-12}$$

The Q-statistic then is used to diagnose traffic anomalies. In the proposed approaches, the orthonormal bases obtained and the variances captured serve as surrogates to the basis of principal components and the singular values (used in the Q-statistic), respectively, used in the PCA-based approach [27, 56].

6.3.1 Randomized Basis for Anomaly Detection (RBAD)

To separate normal and anomalous subspaces as described in (6-3) and (6-4), RBAD first computes the product $\mathbf{B} = \mathbf{Y}\mathbf{\Phi}$ through a random matrix $\mathbf{\Phi} \in \mathbb{R}^{t\times m}$, e.g., drawn from a standard Gaussian distribution. A QR factorization is then performed on $\mathbf{B}$ such as $\mathbf{QR} = \mathbf{B}$. Once the basis $\mathbf{Q}$ is obtained, the subspaces are separated as described in (6-10) and (6-11). Next, the variances captured by $\mathbf{Q}$ are calculated as described in (6-12). Finally, to detect abnormal behavior in the anomalous component, the Q-statistic is applied. The integer q corresponds to the number of steps of power iterations used to improve the accuracy of the basis [39, 71]. A pseudocode for RBAD is given in Table 6.1.

Table 6.1: Pseudocode for the proposed RBAD technique.

Input: Traffic matrix $\mathbf{Y} \in \mathbb{R}^{m\times t}$, integers r, q (e.g., $q = 1$ or $q = 2$).
Output: Anomalies in $\mathbf{Y}$.

1: Generate a random matrix $\mathbf{\Phi} \in \mathbb{R}^{t\times m}$;
2: Compute $\mathbf{B} = (\mathbf{Y}\mathbf{Y}^T)^q\mathbf{Y}\mathbf{\Phi}$;
3: Perform a QR factorization: $\mathbf{B} = \mathbf{QR}$;
4: Separate the subspaces with rank r: $\mathbf{Y} = \hat{\mathbf{Y}} + \tilde{\mathbf{Y}}$;
5: Compute the variances: $\mathbf{\Lambda_Q} = \mathbb{V}\mathrm{ar}\{(\mathbf{Q}^T\mathbf{Y})^T\}$;
6: Apply the Q-statistic to $\tilde{\mathbf{Y}}$: if SPE $> Q_\beta \rightarrow$ anomalies;
7: **return** Anomalies in $\mathbf{Y}$

6.3.2 Switched Subspace-Projected Basis for Anomaly Detection (SSPBAD)

The proposed SSPBAD technique first forms the product $\mathbf{B}_1 = \mathbf{Y}^T\mathbf{B}_2$ by means of a random matrix $\mathbf{B}_2 \in \mathbb{R}^{m\times m}$. Next, $\mathbf{T}_2$ is updated by $\mathbf{B}_1$ such that $\mathbf{B}_2 = \mathbf{Y}\mathbf{B}_1$. A QR factorization is then performed on $\mathbf{B}_2$ such as $\mathbf{QR} = \mathbf{B}_2$ to construct the orthonormal basis for the range of $\mathbf{Y}$. This orthonormal basis, a surrogate to the basis of principal components used in [27, 56], is employed to separate normal and anomalous subspaces as described in (6-10), (6-11). Subsequently, the variances captured by $\mathbf{Q}$ are computed as described in

(6-12). To detect traffic anomalies in the anomalous component, the Q-statistic follows.

To increase robustness of the algorithm for detecting anomalies, we employ different random matrices $\mathbf{B}_2$ as in [51], [17]. The random matrices generated include

- a matrix with i.i.d Gaussian entries i.e., $\mathcal{N}(0, 1)$,
- a matrix whose entries are i.i.d. random variables drawn from a Bernoulli distribution with probability 0.5,
- a Markov matrix whose entries are all nonnegative and the entries of each column add up to 1,
- a matrix whose entries are independently drawn from {-1, 1}.

Therefore, SSPBAD switches among different random matrices and chooses the best one in order to obtain the maximum number of traffic anomalies. A pseudocode for SSPBAD is given in Table 6.2.

Table 6.2: Pseudocode for the proposed SSPBAD technique.

Input: Traffic matrix $\mathbf{Y} \in \mathbb{R}^{m \times t}$ and integer r.
Output: Anomalies in $\mathbf{Y}$.
1: Generate four random matrices $\mathbf{B}_2 \in \mathbb{R}^{m \times m}$;
2: **for** each random matrix **do**
3: Compute $\mathbf{B}_1 = \mathbf{Y}^T \mathbf{B}_2$;
4: Update $\mathbf{B}_2 = \mathbf{Y}\mathbf{B}_1$;
5: Perform a QR factorization: $\mathbf{B}_2 = \mathbf{QR}$;
6: Separate the subspaces with rank r: $\mathbf{Y} = \hat{\mathbf{Y}} + \tilde{\mathbf{Y}}$;
7: Compute the variances: $\mathbf{\Lambda_Q} = \mathbb{V}\text{ar}\{(\mathbf{Q}^T\mathbf{Y})^T\}$;
8: Apply the Q-statistic to $\tilde{\mathbf{Y}}$: if SPE $> Q_\beta \rightarrow$ anomalies;
9: **end for**
10: Choose the random matrix with maximum number of anomalies;
11: **return** Anomalies in $\mathbf{Y}$

The computational cost for either RSMs is $O(tm^2)$ flops, the same order as the PCA-based subspace method. However, the operations of RSMs include matrix-matrix multiplications and a QR factorization, which can be organized to take advantage of the modern computers, an advantage over PCA-based subspace method.

6.4 Simulations

In this section, we verify the proposed methods on synthetically generated data and compare the results with those of PCA [56] and robust PCA [9, 14, 60, 93].

The data matrix $\mathbf{Y}$ is generated according to the model in (6-2) with dimensions $m = 120, n = 240, t = 640$. The low-rank matrix $\mathbf{X}$ is formed by a matrix multiplication $\mathbf{U}\mathbf{V}^T$, where $\mathbf{U} \in \mathbb{R}^{n\times r}$ and $\mathbf{V} \in \mathbb{R}^{t\times r}$ have Gaussian distributed entries $\mathcal{N}(0, 1/n)$ and $\mathcal{N}(0, 1/t)$, respectively and $r = 0.2 \times m$. The routing matrix $\mathbf{R}$ is generated by entries drawn from a Bernoulli distribution with probability 0.05. The sparse matrix of anomalies has $s = 0.001 \times mt$ non-zero entries drawn randomly from the set $\{-1, 1\}$, and the noise matrix $\mathbf{V}$ has independent and identically distributed (i.i.d) Gaussian entries with zero mean and variance σ^2. We set the confidence limit $1 - \beta = 99.5\%$ for the value of the Q-statistic for the approaches studied.

In Figure 6.1, we compare the variances captured by the orthonormal bases of the proposed approaches with those of the principal components since they play a crucial role in the statistical test (the Q-statistic) used to detect anomalies. As can be seen, returned variances by RBAD and SSPBAD are very close to those returned by the SVD.

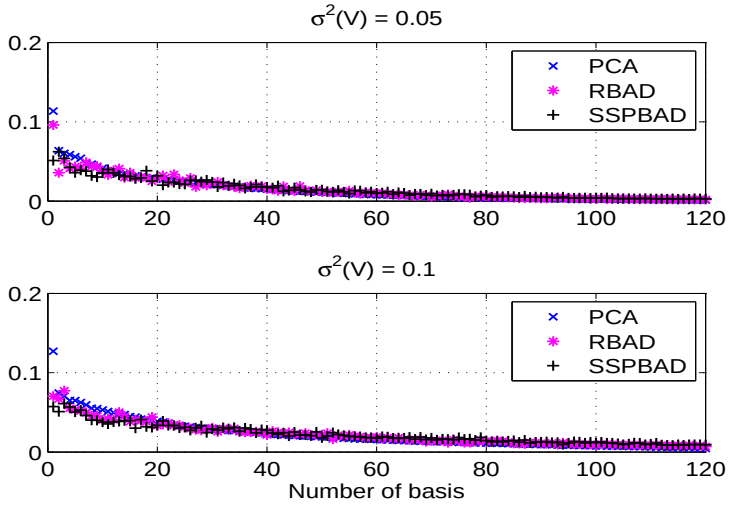


Figure 6.1: A comparison of variances for PCA, RBAD, SSPBAD.

Figure 6.2 compares the detection rate against the number of basis for different approaches. The detection rate combines false-alarm rate and detection probability into one measure and obviates the need for showing these two probabilities in one versus the other manner [96]. As can be seen, the proposed RBAD and SSPBAD outperform PCA when the measurement noise

has a higher variance, due to brittleness of PCA to grossly corrupted entries. Furthermore, robust PCA [9, 14, 60, 93] performs poorly. The reason is that, since we consider measurement noise $\mathbf{V}$ in our data model (6-2), by increasing the rank, these noise samples contaminate the matrix of outliers returned by RPCA and as a result, the abnormal patterns of the network (anomalies) cannot be recovered.

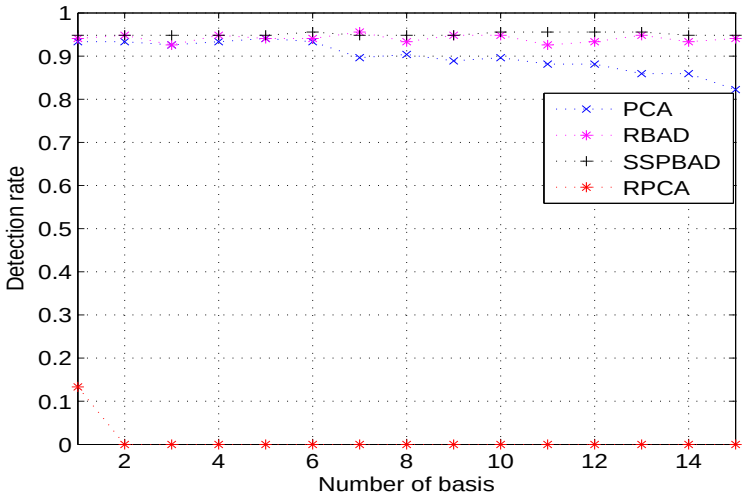


Figure 6.2: A comparison of detection rate for PCA, RBAD, SSPBAD and RPCA. Variance of the measurement noise $\sigma^2 = 0.1$

6.5 Concluding Remarks

In this chapter we have proposed RBAD and SSPBAD, two novel random subspace methods to detect traffic anomalies in IP networks. Both approaches form normal and anomalous subspaces of the traffic data through randomized orthonormal bases constructed for the range of the data. A statistical test is then applied and detects anomalies in the network traffic measurements. Simulations show that RBAD and SSPBAD outperform PCA and RPCA. Further advantages of RBAD and SSPBAD over PCA and RPCA are that their operations include only matrix-matrix multiplications and QR factorizations, which can be organized to exploit modern computers.

7
Conclusions and Future Work

This chapter summarizes the contributions of this dissertation, and presents some brief remarks on directions for extensions and future work.

7.1
Summary of Contributions

In this dissertation, we have presented three different algorithms based on randomization for low-rank matrix approximation. Recently, new formulations, methods and analyses for approximating an input matrix by one of lower rank, based on randomized sampling paradigm have been devised. Randomized methods offer advantages over their traditional counterparts: (i) they are computationally more efficient since they work on a reduced-size version of the input matrix rather than the matrix itself, and (ii) they can be organized to exploit modern computational platforms. Continued research on low-rank matrix approximation methods is fueled by numerous applications in science and engineering in which low-rank matrices arise. We have studied the proposed methods in image reconstruction and robust PCA problems, however they can be applied by researchers to other areas outside the scope of this work. This dissertation offers several novel contributions:

- *Randomized Rank-Revealing UZV Decomposition.* In Chapter 3 we presented RRR-UZVD, a randomized algorithm that approximates a given low-rank matrix. In addition to be computationally efficient, RRR-UZVD's operations only involve matrix-matrix multiplications and QR decompositions. This allows the algorithm to be highly parallelizable on modern architectures. We showed that RRR-UZVD is rank-revealing, and applied it to reconstruct a low-rank image as well as to solve the robust PCA problem.

- *Subspace-Orbit Randomized Singular Value Decomposition.* In Chapter 4 we introduced SOR-SVD, an efficient algorithm that provides a rank-k approximation to an input matrix. We provided a detailed theoretical analysis of SOR-SVD, i.e., lower bounds on the singular values and upper bounds on the error of the low-rank approximation. To best of our

knowledge, this is (unlike R-SVD) the first randomized SVD algorithm based on *two-sided projections* with a complete mathematical analysis. SOR-SVD can exploit modern computational platforms better by exposing higher levels of parallelism than R-SVD. We demonstrated the effectiveness of SOR-SVD through experiments on synthetic data as well as real data in computer vision applications of background/foreground separation in surveillance video and shadow and specularity removal from face images.

– *Compressed Randomized UTV Decompositions.* In Chapter 5 we proposed CoR-UTV, an efficient rank-revealing algorithm based on randomized sampling, which provides a rank-ℓ approximation to a given low-rank matrix. We provided theoretical analysis for CoR-UTV, and studied the algorithm in image reconstruction and low-rank-plus-sparse matrix decomposition applications. CoR-UTV's operations involve matrix-matrix multiplications and QR and QRCP algorithms. This enables the algorithm to readily take advantage of advanced computational platforms.

– *Randomized Subspace Methods.* In chapter 6 we proposed RSMs, namely RBAD and SSPBAD, to diagnose anomalies in IP networks. The advantages of the proposed methods over the PCA-based method are that (i) they do not form the covariance of the link traffic data and, as a result, avoid computing an SVD, (ii) they show better performance, and (iii) they can be organized to take advantage of advanced computational environments.

7.2 Future Directions

This research can be extended in a variety of ways, and new algorithms can be developed along the line of this work.

With regard to SOR-SVD and CoR-UTV, our analysis is specialized to the case where the test matrix is standard Gaussian. However, there are matrices drawn from another distributions such as subsampled randomized Hadamard transform (SRHT) that employing them may lead to potential benefits in terms of arithmetic and communication costs as well as the quality of error bounds.

Implementing RRR-UZVD, SOR-SVD, and CoR-UTV on multicore and accelerator-based computers can be a subject of another work where a detailed study of communication costs is carried out.

The RRR-UZVD, SOR-SVD, and CoR-UTV all require at least two passes through data to factor a given matrix. However, in some applications it is only possible to make one pass over the data. Thus, redesigning these algorithms (or developing new algorithms based on these algorithms) in order to visit the data only once is desired.

With respect to RBAD and SSPBAD, (i) a detailed analysis of these methods can be developed considering the distributional assumption of the test matrices, which is standard Gaussian. (ii) Analogous to the argument given for SOR-SVD and CoR-UTV, test matrices with other distributions can be examined in these methods.

8
Appendix

8.1
Proofs for Chapter 3

Proof of Theorem 3.1.

To prove the first bound (3-10), let $\mathbf{W} \in \mathbb{R}^{m\times m}$ and $\mathbf{D} \in \mathbb{R}^{n\times n}$ be two matrices with orthonormal columns. Obviously, $\mathbf{A}$ and $\mathbf{WAD}$ have the same singular values [83]. Let $\mathbf{B}$ be a submatrix of $\mathbf{WAD}$:

$$\mathbf{B} = \mathbf{W}_1\mathbf{A}\mathbf{D}_1, \tag{8-1}$$

where $\mathbf{W}_1 \in \mathbb{R}^{k\times m}$ contains the first k rows of $\mathbf{W}$, and $\mathbf{D}_1 \in \mathbb{R}^{n\times k}$ contains the first k columns of $\mathbf{D}$, with singular values $\beta_1 \geq \beta_2 \geq ... \geq \beta_k$. Thus

$$\mathbf{B}\mathbf{B}^T = \mathbf{W}_1\mathbf{M}\mathbf{M}^T\mathbf{W}_1^T, \tag{8-2}$$

where $\mathbf{M} = \mathbf{A}\mathbf{D}_1$ is an $m\times k$ matrix. Hence $\mathbf{B}\mathbf{B}^T$ is a $k\times k$ principal submatrix of the $m \times m$ Hermitian matrix $\mathbf{W}\mathbf{M}\mathbf{M}^T\mathbf{W}^T$. Let $\alpha_1 \geq \alpha_2 \geq ... \geq \alpha_k$ be the singular values of $\mathbf{M}$. Therefore the eigenvalues of $\mathbf{M}\mathbf{M}^T$ are given as follows

$$\alpha_1^2 \geq \alpha_2^2 \geq ... \geq \alpha_k^2 \geq \alpha_{k+1}^2 = ... = \alpha_m^2 = 0. \tag{8-3}$$

From the well-known formulas relating the eigenvalues of a Hermitian matrix with the eigenvalues of a principal submatrix [92], we get

$$\alpha_1^2 \geq \beta_1^2 \geq ... \geq \alpha_k^2 \geq \beta_k^2. \tag{8-4}$$

Now we form $\mathbf{M}^T\mathbf{M}$ whose nonzero eigenvalues coincide with those of $\mathbf{M}\mathbf{M}^T$:

$$\mathbf{M}^T\mathbf{M} = \mathbf{D}_1^T\mathbf{A}^T\mathbf{A}\mathbf{D}_1. \tag{8-5}$$

Hence $\mathbf{M}^T\mathbf{M}$ is a $k \times k$ principal submatrix of the $n \times n$ Hermitian matrix $\mathbf{D}^T\mathbf{A}^T\mathbf{A}\mathbf{D}$. Thus

$$\sigma_1^2 \geq \alpha_1^2 \geq ... \geq \sigma_k^2 \geq \alpha_k^2, \tag{8-6}$$

and consequently

$$\sigma_1^2 \geq \alpha_1^2 \geq \beta_1^2 \geq ... \geq \sigma_k^2 \geq \alpha_k^2 \geq \beta_k^2. \tag{8-7}$$

By substituting $\mathbf{W}_1$ and $\mathbf{D}_1$ in (8-1), with $\mathbf{U}_1^T$ and $\mathbf{V}_1$ in (3-9), respectively,

then we have

$$\sigma_k(\mathbf{Z}) \leq \sigma_k. \tag{8-8}$$

To prove the second bound (3-11), let the matrix $\hat{\mathbf{A}}_{\text{UZV}}$ have an SVD such as $\hat{\mathbf{A}}_{\text{UZV}} = \hat{\mathbf{U}}\hat{\mathbf{\Sigma}}\hat{\mathbf{V}}^T$, where $\hat{\mathbf{U}}$ and $\hat{\mathbf{V}}$ have orthonormal columns, and $\hat{\mathbf{\Sigma}}$ contains the singular values $\hat{\sigma}_i$s. We write $\hat{\mathbf{A}}_{\text{UZV}} = \hat{\mathbf{U}}_k\hat{\mathbf{\Sigma}}_k\hat{\mathbf{V}}_k^T + \hat{\mathbf{U}}_0\hat{\mathbf{\Sigma}}_0\hat{\mathbf{V}}_0^T$, and by the argument in the proof of (3-10), we have

$$\sigma_{k+1} = \|\mathbf{A}^T\mathbf{U}_0\|_2 \geq \hat{\sigma}_{k+1} = \|\hat{\mathbf{A}}_{\text{UZV}}^T\hat{\mathbf{U}}_0\|_2. \tag{8-9}$$

We also have

$$\hat{\sigma}_{k+1} = \|\hat{\mathbf{A}}_{\text{UZV}}^T\hat{\mathbf{U}}_0\|_2 \leq \|\hat{\mathbf{A}}_{\text{UZV}}^T\mathbf{U}_2\|_2. \tag{8-10}$$

This relation holds since $\mathbf{U}_2$ is an approximation to $\hat{\mathbf{U}}_0$. In practice, a combination of the power method and column permutation techniques proposed results in a $\mathbf{U}_2$ that spans the null space of $\hat{\mathbf{A}}$. As a result

$$\hat{\sigma}_{k+1} \approx \|\hat{\mathbf{A}}\text{UZV}^T\mathbf{U}_2\|_2. \tag{8-11}$$

By substituting $\hat{\mathbf{A}}_{\text{UZV}}$ (3-9) into (8-11), it follows

$$\begin{aligned}\hat{\sigma}_{k+1} &\approx \| \begin{bmatrix}\mathbf{V}_1 & \mathbf{V}_2\end{bmatrix} \begin{bmatrix}\mathbf{Z}_k^T & \mathbf{H}^T \\ \mathbf{G}^T & \mathbf{E}^T\end{bmatrix} \begin{bmatrix}\mathbf{U}_1^T \\ \mathbf{U}_2^T\end{bmatrix} \mathbf{U}_2\|_2 \\ &\approx \|[\mathbf{H} \quad \mathbf{E}]\|_2.\end{aligned} \tag{8-12}$$

To prove the third bound (3-12), with a similar argument, we have

$$\sigma_{k+1} = \|\mathbf{A}\mathbf{V}_0\|_2 \geq \hat{\sigma}_{k+1} = \|\hat{\mathbf{A}}_{\text{UZV}}\hat{\mathbf{V}}_0\|_2. \tag{8-13}$$

And accordingly

$$\hat{\sigma}_{k+1} \approx \|\hat{\mathbf{A}}_{\text{UZV}}\mathbf{V}_2\|_2. \tag{8-14}$$

By substituting $\hat{\mathbf{A}}_{\text{UZV}}$ (3-9) into (8-14), it follows that

$$\begin{aligned}\hat{\sigma}_{k+1} &\approx \| \begin{bmatrix}\mathbf{U}_1 & \mathbf{U}_2\end{bmatrix} \begin{bmatrix}\mathbf{Z}_k & \mathbf{G} \\ \mathbf{H} & \mathbf{E}\end{bmatrix} \begin{bmatrix}\mathbf{V}_1^T \\ \mathbf{V}_2^T\end{bmatrix} \mathbf{V}_2\|_2 \\ &\approx \|[\mathbf{G}^T \quad \mathbf{E}^T]^T\|_2.\end{aligned} \tag{8-15}$$

This completes the proof.

8.2 Proofs for Chapter 4

8.2.1 Proof of Theorem 4.2

We first rewrite the term in the left-hand side of (4-9):

$$\begin{aligned}\|\mathbf{A}-\mathbf{Q}_1\mathbf{M}\mathbf{Q}_2^T\|_F^2 &= \|\mathbf{A}-\mathbf{Q}_1\mathbf{Q}_1^T\mathbf{A}\mathbf{Q}_2\mathbf{Q}_2^T+\mathbf{Q}_1\mathbf{Q}_1^T\mathbf{A}\mathbf{Q}_2\mathbf{Q}_2^T-\mathbf{Q}_1\mathbf{M}\mathbf{Q}_2^T\|_F^2\\ &= \|\mathbf{A}-\mathbf{Q}_1\mathbf{Q}_1^T\mathbf{A}\mathbf{Q}_2\mathbf{Q}_2^T\|_F^2+\|\mathbf{Q}_1^T\mathbf{A}\mathbf{Q}_2-\mathbf{M}\|_F^2.\end{aligned} \tag{8-16}$$

By Theorem 2.1, the result in (4-9) immediately follows.

According to Theorem 2.1, $\mathbf{A}_k$ is the best low-rank approximation to $\mathbf{A}$, whereas according to (4-9) (Theorem 4.2), $\mathbf{Q}_1\mathbf{M}_k\mathbf{Q}_2^T$ is the best restricted (within a subspace) low-rank approximation to $\mathbf{A}$ with respect to the Frobenius norm. This leads to the following result

$$\begin{aligned}\|\mathbf{A}-\mathbf{A}_k\|_F \leq& \|\mathbf{A}-\mathbf{Q}_1\mathbf{M}_k\mathbf{Q}_2^T\|_F\\ \leq& \|\mathbf{A}-\mathbf{Q}_1\mathbf{Q}_1^T\mathbf{A}_k\mathbf{Q}_2\mathbf{Q}_2^T\|_F.\end{aligned} \tag{8-17}$$

The second relation holds because $\mathbf{Q}_1\mathbf{Q}_1^T\mathbf{A}_k\mathbf{Q}_2\mathbf{Q}_2^T$ is an undistinguished restricted Frobenius norm approximation to $\mathbf{A}$.

To prove (4-10), we calculate

$$\begin{aligned}&\|\mathbf{A}-\mathbf{Q}_1\mathbf{Q}_1^T\mathbf{A}_k\mathbf{Q}_2\mathbf{Q}_2^T\|_F^2\\ &= \mathrm{trace}\Big((\mathbf{A}-\mathbf{Q}_1\mathbf{Q}_1^T\mathbf{A}_k\mathbf{Q}_2\mathbf{Q}_2^T)^T(\mathbf{A}-\mathbf{Q}_1\mathbf{Q}_1^T\mathbf{A}_k\mathbf{Q}_2\mathbf{Q}_2^T)\Big)\\ &= \mathrm{trace}\Big((\mathbf{A}-\mathbf{A}_k\mathbf{Q}_2\mathbf{Q}_2^T+\mathbf{A}_k\mathbf{Q}_2\mathbf{Q}_2^T-\mathbf{Q}_1\mathbf{Q}_1^T\mathbf{A}_k\mathbf{Q}_2\mathbf{Q}_2^T)^T\\ &\quad\times(\mathbf{A}-\mathbf{A}_k\mathbf{Q}_2\mathbf{Q}_2^T+\mathbf{A}_k\mathbf{Q}_2\mathbf{Q}_2^T-\mathbf{Q}_1\mathbf{Q}_1^T\mathbf{A}_k\mathbf{Q}_2\mathbf{Q}_2^T)\Big)\\ &= \|\mathbf{A}-\mathbf{A}_k\mathbf{Q}_2\mathbf{Q}_2^T\|_F^2+\|\mathbf{A}_k\mathbf{Q}_2\mathbf{Q}_2^T-\mathbf{Q}_1\mathbf{Q}_1^T\mathbf{A}_k\mathbf{Q}_2\mathbf{Q}_2^T\|_F^2+\\ &\quad 2\mathrm{trace}\Big((\mathbf{A}-\mathbf{A}_k\mathbf{Q}_2\mathbf{Q}_2^T)^T(\mathbf{A}_k\mathbf{Q}_2\mathbf{Q}_2^T-\mathbf{Q}_1\mathbf{Q}_1^T\mathbf{A}_k\mathbf{Q}_2\mathbf{Q}_2^T)\Big)\\ &= \|\mathbf{A}-\mathbf{A}_k\mathbf{Q}_2\mathbf{Q}_2^T\|_F^2+\|\mathbf{A}_k-\mathbf{Q}_1\mathbf{Q}_1^T\mathbf{A}_k\|_F^2\\ &\quad+2\mathrm{trace}\Big((\mathbf{I}-\mathbf{Q}_2\mathbf{Q}_2^T)\underbrace{\mathbf{A}_k\mathbf{Q}_2\mathbf{Q}_2^T(\mathbf{A}-\mathbf{A}_k\mathbf{Q}_2\mathbf{Q}_2^T)^T}_{0}\Big).\end{aligned} \tag{8-18}$$

Combining the last relation in (8-18) with equation (8-17) gives

$$\|\mathbf{A}-\mathbf{Q}_1\mathbf{M}_k\mathbf{Q}_2^T\|_F \leq \|\mathbf{A}-\mathbf{A}_k\mathbf{Q}_2\mathbf{Q}_2^T\|_F+\|\mathbf{A}_k-\mathbf{Q}_1\mathbf{Q}_1^T\mathbf{A}_k\|_F. \tag{8-19}$$

Writing $\mathbf{A}=\mathbf{A}_k+\mathbf{A}_0$, and applying the triangle inequality gives equation (4-10).

To prove (4-11), we observe that

$$\begin{aligned}\|\mathbf{A}-\mathbf{Q}_1\mathbf{M}\mathbf{Q}_2^T\|_2 \leq& \|\mathbf{A}-\mathbf{Q}_1\mathbf{M}_k\mathbf{Q}_2^T\|_2\\ \leq& \|\mathbf{A}-\mathbf{Q}_1\mathbf{Q}_1^T\mathbf{A}_k\mathbf{Q}_2\mathbf{Q}_2^T\|_2\\ =& \|\mathbf{A}_0+\mathbf{A}_k-\mathbf{Q}_1\mathbf{Q}_1^T\mathbf{A}_k\mathbf{Q}_2\mathbf{Q}_2^T\|_2\\ \leq& \|\mathbf{A}_0\|_2+\|\mathbf{A}_k-\mathbf{Q}_1\mathbf{Q}_1^T\mathbf{A}_k\mathbf{Q}_2\mathbf{Q}_2^T\|_2.\end{aligned} \tag{8-20}$$

We write the second term of the last equation as:

$$\begin{aligned}
&\|\mathbf{A}_k - \mathbf{Q}_1\mathbf{Q}_1^T\mathbf{A}_k\mathbf{Q}_2\mathbf{Q}_2^T\|_2 \\
&= \|\mathbf{A}_k - \mathbf{Q}_1\mathbf{Q}_1^T\mathbf{A}_k + \mathbf{Q}_1\mathbf{Q}_1^T\mathbf{A}_k - \mathbf{Q}_1\mathbf{Q}_1^T\mathbf{A}_k\mathbf{Q}_2\mathbf{Q}_2^T\|_2 \\
&\leq \|\mathbf{A}_k - \mathbf{Q}_1\mathbf{Q}_1^T\mathbf{A}_k\|_2 + \|\mathbf{Q}_1\mathbf{Q}_1^T(\mathbf{A}_k - \mathbf{A}_k\mathbf{Q}_2\mathbf{Q}_2^T)\|_2 \\
&\leq \|\mathbf{A}_k - \mathbf{Q}_1\mathbf{Q}_1^T\mathbf{A}_k\|_F + \|\mathbf{A}_k - \mathbf{A}_k\mathbf{Q}_2\mathbf{Q}_2^T\|_F.
\end{aligned} \tag{8-21}$$

Plugging this into equation (8-20) yields (4-11).

The last relation in (8-21) holds due to the unitary invariance property of the ℓ_2-norm and, furthermore, to the relation that for any matrix $\mathbf{\Pi}$, $\|\mathbf{\Pi}\|_2 \leq \|\mathbf{\Pi}\|_F$.

8.2.2 Proof of Lemma 4.6

According to Lemma 4.5, we have

$$\mathbf{Q}_2\mathbf{Q}_2^T = \tilde{\mathbf{Q}}\tilde{\mathbf{Q}}^T. \tag{8-22}$$

Thus

$$\mathbf{Q}_1\mathbf{Q}_1^T\mathbf{A}\mathbf{Q}_2\mathbf{Q}_2^T = \mathbf{Q}_1\mathbf{Q}_1^T\mathbf{A}\tilde{\mathbf{Q}}\tilde{\mathbf{Q}}^T. \tag{8-23}$$

We now write

$$\mathbf{A}\tilde{\mathbf{Q}}\tilde{\mathbf{Q}}^T = \mathbf{A}(\tilde{\mathbf{Q}}_1 \quad \tilde{\mathbf{Q}}_2 \quad \tilde{\mathbf{Q}}_3)\tilde{\mathbf{Q}}^T = \mathbf{U}\underbrace{\begin{bmatrix} \mathbf{\Sigma}_1 & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{\Sigma}_2 & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{\Sigma}_3 \end{bmatrix}}_{\mathbf{S}} \mathbf{V}^T[\tilde{\mathbf{Q}}_1 \quad \tilde{\mathbf{Q}}_2 \quad \tilde{\mathbf{Q}}_3]\,\tilde{\mathbf{Q}}. \tag{8-24}$$

We now write $\mathbf{S}$ as:

$$\mathbf{S} = \left(\begin{array}{c|c} (\mathbf{\Sigma}_1 \quad \mathbf{0} \quad \mathbf{0})\mathbf{V}^T\tilde{\mathbf{Q}}_1 & (\mathbf{\Sigma}_1 \quad \mathbf{0} \quad \mathbf{0})\mathbf{V}^T(\tilde{\mathbf{Q}}_2 \;\; \tilde{\mathbf{Q}}_3) \\ \hline \begin{pmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{\Sigma}_1 & \mathbf{0} \\ \mathbf{0} & \mathbf{\Sigma}_2 \end{pmatrix}^T \mathbf{V}^T\tilde{\mathbf{Q}}_1 & \begin{pmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{\Sigma}_1 & \mathbf{0} \\ \mathbf{0} & \mathbf{\Sigma}_2 \end{pmatrix}^T \mathbf{V}^T(\tilde{\mathbf{Q}}_2 \;\; \tilde{\mathbf{Q}}_3) \end{array}\right)$$

We observe that the matrix $(\mathbf{\Sigma}_1 \quad \mathbf{0} \quad \mathbf{0})\mathbf{V}^T\tilde{\mathbf{Q}}_1$ is a submatrix of $\mathbf{Q}_1\mathbf{Q}_1^T\mathbf{A}\mathbf{Q}_2\mathbf{Q}_2^T$. By relations in equation (4-21), we have

$$\mathbf{V}\begin{bmatrix} \mathbf{I} \\ \mathbf{0} \\ \mathbf{W}_1 \end{bmatrix} = \tilde{\mathbf{Q}}_1\tilde{\mathbf{R}}_{11}. \tag{8-25}$$

and accordingly

$$\|\tilde{\mathbf{R}}_{11}\|_2 = \sqrt{1 + \|\mathbf{W}_1\|_2^2}. \tag{8-26}$$

By substituting $\tilde{\mathbf{Q}}_1$ of (8-25) into $(\mathbf{\Sigma}_1 \quad \mathbf{0} \quad \mathbf{0})\mathbf{V}^T\tilde{\mathbf{Q}}_1$,

$$(\mathbf{\Sigma}_1 \quad \mathbf{0} \quad \mathbf{0})\mathbf{V}^T\tilde{\mathbf{Q}}_1 = (\mathbf{\Sigma}_1 \quad \mathbf{0} \quad \mathbf{0})\mathbf{V}^T\mathbf{V}\begin{bmatrix}\mathbf{I}\\ \mathbf{0}\\ \mathbf{W}_1\end{bmatrix}\tilde{\mathbf{R}}_{11}^{-1} = \mathbf{\Sigma}_1\tilde{\mathbf{R}}_{11}^{-1}, \tag{8-27}$$

and by Remark 4.4 it follows that

$$\sigma_j(\hat{\mathbf{A}}_{\text{SOR}}) = \sigma_j(\mathbf{Q}_1\mathbf{Q}_1^T\mathbf{A}\mathbf{Q}_2\mathbf{Q}_2^T) \geq \sigma_j(\mathbf{\Sigma}_1\tilde{\mathbf{R}}_{11}^{-1}). \tag{8-28}$$

On the other hand, we also have

$$\sigma_j = \sigma_j(\mathbf{\Sigma}_1\tilde{\mathbf{R}}_{11}^{-1}\tilde{\mathbf{R}}_{11}) \leq \sigma_j(\mathbf{\Sigma}_1\tilde{\mathbf{R}}_{11}^{-1})\|\tilde{\mathbf{R}}_{11}\|_2. \tag{8-29}$$

Plugging the last relation into (8-28) and using (8-26), we obtain (4-22).

8.2.3 Proof of Theorem 4.7

To prove (4-23), according to the definition of $\mathbf{W}_1$ in equation (4-20), we obtain

$$\|\mathbf{W}_1\|_2 \leq \left(\frac{\sigma_{\ell-p+1}}{\sigma_k}\right)^2 \|\mathbf{\Omega}_2\|_2\|\mathbf{\Omega}_1^\dagger\|_2. \tag{8-30}$$

Taking this together with equation (4-22), yields the result in (4-23) for $j = k$.

To prove (4-24), we observe that when the power method is used, the SOR-SVD computation starts off by setting $\mathbf{T}_2 = \mathbf{\Omega}$, and $\mathbf{T}_2$ is updated such that

$$\mathbf{T}_2 = (\mathbf{A}^T\mathbf{A})^q\mathbf{A}^T\mathbf{T}_1 = (\mathbf{A}^T\mathbf{A})^q\mathbf{A}^T\mathbf{A}\mathbf{\Omega}. \tag{8-31}$$

By writing the SVD of $\mathbf{A}$ (2-1),

$$\mathbf{T}_2 = \mathbf{V}\mathbf{\Sigma}^{2q+2}\mathbf{V}^T\mathbf{\Omega} = \mathbf{V}\begin{bmatrix}\mathbf{\Sigma}_1^{2q+2} & \mathbf{0} & \mathbf{0}\\ \mathbf{0} & \mathbf{\Sigma}_2^{2q+2} & \mathbf{0}\\ \mathbf{0} & \mathbf{0} & \mathbf{\Sigma}_3^{2q+2}\end{bmatrix}\begin{bmatrix}\mathbf{V}_1^T\mathbf{\Omega}\\ \\ \mathbf{V}_2^T\mathbf{\Omega}\end{bmatrix} = \mathbf{Q_2}\mathbf{R_2}. \tag{8-32}$$

Consequently, the matrix $\mathbf{X}$, defined in (4-19), is now defined as follows:

$$\mathbf{X} = \left[\mathbf{\Omega}_1^\dagger\begin{pmatrix}\mathbf{\Sigma}_1^{2q+2} & \mathbf{0}\\ \mathbf{0} & \mathbf{\Sigma}_2^{2q+2}\end{pmatrix}^{-1}, \tilde{\mathbf{X}}\right]. \tag{8-33}$$

Forming $\mathbf{T}_2\mathbf{X}$ yields:

$$\mathbf{W}_1 = \mathbf{\Sigma}_3^{2q+2}\mathbf{\Omega}_2\mathbf{\Omega}_1^\dagger\mathbf{\Sigma}_1^{-(2q+2)}. \tag{8-34}$$

Thus

$$\|\mathbf{W}_1\|_2 \leq \left(\frac{\sigma_{\ell-p+1}}{\sigma_k}\right)^{2q+2} \|\mathbf{\Omega}_2\|_2\|\mathbf{\Omega}_1^\dagger\|_2. \tag{8-35}$$

Taking this together with equation (4-22), yields the result for $j = k$.

To prove Theorem 4.7 for any $1 \leq j < k$, since by Remark 4.4 $\sigma_j(\hat{\mathbf{A}}_{\text{SOR}}) = \sigma_j(\mathbf{Q}_1\mathbf{Q}_1^T\mathbf{A}\mathbf{Q}_2\mathbf{Q}_2^T)$, it is only required to repeat all previous arguments for a rank j truncated SVD.

8.2.4 Proof of Theorem 4.9

First, by plugging (4-30) and (4-42) into (4-10) and (4-11), for $\varrho = 2, F$, we obtain

$$\|\mathbf{A} - \hat{\mathbf{A}}_{\text{SOR}}\|_\varrho \leq \|\mathbf{A}_0\|_\varrho + \frac{\sqrt{k}\|\mathbf{W}_1\boldsymbol{\Sigma}_1\|_2\sigma_1}{\sqrt{{\sigma_1}^2 + \|\mathbf{W}_1\boldsymbol{\Sigma}_1\|_2^2}} + \frac{\sqrt{k}\|\mathbf{D}_1\boldsymbol{\Sigma}_1\|_2\sigma_1}{\sqrt{{\sigma_1}^2 + \|\mathbf{D}_1\boldsymbol{\Sigma}_1\|_2^2}}. \quad (8\text{-}36)$$

When the basic form of SOR-SVD is implemented, we write $\mathbf{W}_1\boldsymbol{\Sigma}_1$ as

$$\mathbf{W}_1\boldsymbol{\Sigma}_1 = \boldsymbol{\Sigma}_3^2\boldsymbol{\Omega}_2\boldsymbol{\Omega}_1^\dagger\boldsymbol{\Sigma}_1^{-1}. \quad (8\text{-}37)$$

where $\mathbf{W}_1$ is defined in (4-20). Thus

$$\|\mathbf{W}_1\boldsymbol{\Sigma}_1\|_2 \leq \left(\frac{\sigma_{\ell-p+1}^2}{\sigma_k}\right)\|\boldsymbol{\Omega}_2\|_2\|\boldsymbol{\Omega}_1^\dagger\|_2. \quad (8\text{-}38)$$

For $\mathbf{D}_1\boldsymbol{\Sigma}_1$, we write

$$\mathbf{D}_1\boldsymbol{\Sigma}_1 = \boldsymbol{\Sigma}_3\boldsymbol{\Omega}_2\boldsymbol{\Omega}_1^\dagger. \quad (8\text{-}39)$$

where $\mathbf{D}_1$ is defined in (4-34). Thus

$$\|\mathbf{D}_1\boldsymbol{\Sigma}_1\|_2 \leq \sigma_{\ell-p+1}\|\boldsymbol{\Omega}_2\|_2\|\boldsymbol{\Omega}_1^\dagger\|_2. \quad (8\text{-}40)$$

Plugging (8-38) and (8-40) into (8-36), and dividing both the numerator and denominator by σ_1 gives the result in (4-43).

For the case when the power method is incorporated into the algorithm, by using (8-34) we write $\mathbf{W}_1\boldsymbol{\Sigma}_1$ as

$$\mathbf{W}_1\boldsymbol{\Sigma}_1 = \boldsymbol{\Sigma}_3^{2q+2}\boldsymbol{\Omega}_2\boldsymbol{\Omega}_1^\dagger\boldsymbol{\Sigma}_1^{-(2q+1)}. \quad (8\text{-}41)$$

Consequently, we have

$$\|\mathbf{W}_1\boldsymbol{\Sigma}_1\|_2 \leq \frac{\sigma_{\ell-p+1}^2}{\sigma_k}\left(\frac{\sigma_{\ell-p+1}}{\sigma_k}\right)^{2q}\|\boldsymbol{\Omega}_2\|_2\|\boldsymbol{\Omega}_1^\dagger\|_2. \quad (8\text{-}42)$$

To get $\mathbf{D}_1\boldsymbol{\Sigma}_1$, we first need to obtain $\mathbf{D}_1$, equation (4-34), for the case when the power method is employed. To this end, the procedure starts with substituting $\mathbf{T}_1$ in equation (4-31) with

$$\mathbf{T}_1 = (\mathbf{A}\mathbf{A}^T)^q\mathbf{A}\boldsymbol{\Omega}. \quad (8\text{-}43)$$

By writing the SVD of $\mathbf{A}$ (2-1), we have

$$\mathbf{T}_1 = \mathbf{U}\boldsymbol{\Sigma}^{2q+1}\mathbf{V}^T\boldsymbol{\Omega} = \mathbf{U}\begin{bmatrix}\boldsymbol{\Sigma}_1^{2q+1} & \mathbf{0} & \mathbf{0}\\ \mathbf{0} & \boldsymbol{\Sigma}_2^{2q+1} & \mathbf{0}\\ \mathbf{0} & \mathbf{0} & \boldsymbol{\Sigma}_3^{2q+1}\end{bmatrix}\begin{bmatrix}\mathbf{V}_1^T\boldsymbol{\Omega}\\ \\ \mathbf{V}_2^T\boldsymbol{\Omega}\end{bmatrix} = \mathbf{Q_1}\mathbf{R_1}. \quad (8\text{-}44)$$

and consequently $\mathbf{X}$ is defined as:

$$\mathbf{X} = \left[\boldsymbol{\Omega}_1^\dagger \begin{pmatrix} \boldsymbol{\Sigma}_1^{2q+1} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{\Sigma}_2^{2q+1} \end{pmatrix}^{-1}, \tilde{\mathbf{X}}\right]. \tag{8-45}$$

Forming $\mathbf{T}_1\mathbf{X}$ yields:

$$\mathbf{D}_1 = \boldsymbol{\Sigma}_3^{2q+2}\boldsymbol{\Omega}_2\boldsymbol{\Omega}_1^\dagger\boldsymbol{\Sigma}_1^{-(2q+1)}. \tag{8-46}$$

and we have

$$\mathbf{D}_1\boldsymbol{\Sigma}_1 = \boldsymbol{\Sigma}_3^{2q+1}\boldsymbol{\Omega}_2\boldsymbol{\Omega}_1^\dagger\boldsymbol{\Sigma}_1^{-2q}. \tag{8-47}$$

Accordingly, we have

$$\|\mathbf{D}_1\boldsymbol{\Sigma}_1\|_2 \leq \sigma_{\ell-p+1}\left(\frac{\sigma_{\ell-p+1}}{\sigma_k}\right)^{2q}\|\boldsymbol{\Omega}_2\|_2\|\boldsymbol{\Omega}_1^\dagger\|_2. \tag{8-48}$$

Substituting (8-42) and (8-48) into (8-36), and dividing both the numerator and denominator by σ_1 gives the result.

8.2.5 Proof of Proposition 4.11

To prove Proposition 4.11, we first present several key results that are used later on.

Proposition 8.1 *(Halko et al. [39]). For fixed matrices* $\mathbf{S}$ *and* $\mathbf{T}$*, and a standard Gaussian matrix* $\mathbf{G}$*, we have*

$$\mathbb{E}\|\mathbf{SGT}\|_2 \leq \|\mathbf{S}\|_2\|\mathbf{T}\|_F + \|\mathbf{S}\|_F\|\mathbf{T}\|_2.$$

Proposition 8.2 *(Halko et al. [39]). Let* h *be a real valued Lipschitz function on matrices:*

$$|h(\mathbf{X}) - h(\mathbf{X})| \leq L\|\mathbf{X} - \mathbf{Y}\|_F, \quad \textit{for all} \quad \mathbf{X}, \mathbf{Y}$$

where $L > 0$*. Draw a standard Gaussian matrix* $\mathbf{G}$*. Then*

$$\mathbb{P}\{h(\mathbf{G}) \geq \mathbb{E}h(\mathbf{G}) + Lu\} \leq e^{-u^2/2}.$$

Proposition 8.3 *(Halko et al. [39]). Let* $\mathbf{G} \in \mathbb{R}^{\ell-p\times p}$ *be a Gaussian matrix, where* $p \geq 0$ *and* $\ell - p \geq 2$*. Then for* $t \geq 1$*,*

$$\mathbb{P}\left\{\|\mathbf{G}^\dagger\|_2 \geq \frac{et\sqrt{\ell}}{p+1}\right\} \leq t^{-(p+1)}.$$

Proposition 8.4 *(Gu [36]). Let $g(\cdot)$ be a non-negative continuously differentiable function with $g(0) = 0$, and let $\mathbf{G}$ be a random matrix, then*

$$\mathbb{E}_g(\|\mathbf{G}\|_2) = \int_0^\infty g\prime(x)\mathbb{P}\{\|\mathbf{G}\|_2 \ \geq x\}dx.$$

We first define the following function

$$g(x) \triangleq \sqrt{\frac{\alpha^2 x^2}{1+\beta^2 x^2}},$$

whose derivative with respect to x is defined as

$$g\prime(x) = \frac{\alpha^2 x}{(1+\beta^2 x^2)^2\sqrt{\frac{\alpha^2 x^2}{1+\beta^2 x^2}}}.$$

where $\alpha, \beta > 0$.

Next, for the Gaussian matrix $\mathbf{G} \in \mathbb{R}^{m\times n}$, we define a function $h(\mathbf{G}) = \|\mathbf{G}\|_2$. By Proposition 8.1, it follows that

$$\mathbb{E}(h(\mathbf{G})) \leq \sqrt{m} + \sqrt{n} < \sqrt{m} + \sqrt{n} + 3 \triangleq \varepsilon.$$

By definition of $g(x)$, Proposition 8.4, and Proposition 8.2, for $x = u - \varepsilon$, we have

$$\mathbb{P}\{\|\mathbf{G}\|_2 \geq x\} \leq \mathrm{e}^{-u^2/2}.$$

We can rewrite equation (4-45) as follows

$$\begin{aligned}
&\mathbb{E}\left(\sqrt{\frac{\alpha^2\|\mathbf{G}\|_2^2}{1+\beta^2\|\mathbf{G}\|_2^2}}\right) = \mathbb{E}(g(\|\mathbf{G}\|_2)) = \int_0^\infty g\prime(x)\mathbb{P}\{\|\mathbf{G}\|_2 \geq x\}dx \\
&\leq \int_0^\varepsilon g\prime(x)dx + \int_\varepsilon^\infty g\prime(x)\mathbb{P}\{\|\mathbf{G}\|_2 \geq x\}dx \\
&\leq \sqrt{\frac{\alpha^2\varepsilon^2}{1+\beta^2\varepsilon^2}} + \int_\varepsilon^\infty \frac{\alpha^2 x}{(1+\beta^2x^2)^2\sqrt{\frac{\alpha^2x^2}{1+\beta^2x^2}}}\mathrm{e}^{(x-\varepsilon)^2/2}dx \\
&= \sqrt{\frac{\alpha^2\varepsilon^2}{1+\beta^2\varepsilon^2}} + \frac{\alpha^2}{(1+\beta^2\varepsilon^2)^2\sqrt{\frac{\alpha^2\varepsilon^2}{1+\beta^2\varepsilon^2}}} \underbrace{\int_0^\infty (u+\varepsilon)^{-u^2/2}du}_{\varepsilon\sqrt{\pi/2}+1}.
\end{aligned}$$

We now must find a $\nu > 0$ such that

$$\sqrt{\frac{\alpha^2\varepsilon^2}{1+\beta^2\varepsilon^2}} + \frac{\alpha^2(\varepsilon\sqrt{\pi/2}+1)}{(1+\beta^2\varepsilon^2)^2\sqrt{\frac{\alpha^2\varepsilon^2}{1+\beta^2\varepsilon^2}}} \leq \sqrt{\frac{\alpha^2\nu^2}{1+\beta^2\nu^2}},$$

which leads to

$$\nu^2-\varepsilon^2 \geq \frac{1+\beta^2\nu^2}{1+\beta^2\varepsilon^2}\times(\varepsilon\sqrt{\pi/2}+1)\times\left[\frac{\sqrt{\frac{\alpha^2\nu^2}{1+\beta^2\nu^2}}}{\sqrt{\frac{\alpha^2\varepsilon^2}{1+\beta^2\varepsilon^2}}}+1\right].$$

The right-hand side of the inequality approaches the maximum value as β approaches ∞. Thus ν must satisfy

$$\nu^2-\varepsilon^2 \geq \frac{\nu^2}{\varepsilon^2}(\varepsilon\sqrt{\pi/2}+1).$$

which results in

$$\nu \geq \frac{\varepsilon^2}{\sqrt{\varepsilon^2-(\varepsilon\sqrt{\pi/2}+1)}}.$$

The inequality is satisfied when $\nu=\sqrt{m}+\sqrt{n}+7=\varepsilon+4$.

8.2.6 Proof of Proposition 4.13

According to Proposition 8.3, for any $x>0$, we have

$$\mathbb{P}\Big\{\|\mathbf{G}^\dagger\|_2 \geq x\} \leq \left(\frac{p+1}{\mathrm{e}\sqrt{\ell}}x\right)^{-(p+1)}.$$

With similar arguments to those in the proof of Proposition 4.11, for a constant $C>0$ to be determined later on, we have

$$\begin{aligned}
&\mathbb{E}\left(\sqrt{\frac{\alpha^2\|\mathbf{G}^\dagger\|_2^2}{1+\beta^2\|\mathbf{G}^\dagger\|_2^2}}\right)=\mathbb{E}(g(\|\mathbf{G}^\dagger\|_2))=\int_0^\infty g\prime(x)\mathbb{P}\{\|\mathbf{G}^\dagger\|_2\geq x\}dx\\
&\leq \int_0^C g\prime(x)dx+\int_C^\infty g\prime(x)\mathbb{P}\{\|\mathbf{G}^\dagger\|_2\geq x\}dx\\
&\leq \sqrt{\frac{\alpha^2C^2}{1+\beta^2C^2}}+\int_C^\infty \frac{\alpha^2\left(\frac{p+1}{\mathrm{e}\sqrt{\ell}}x\right)^{-(p+1)}}{(1+\beta^2x^2)^2\sqrt{\frac{\alpha^2x^2}{1+\beta^2x^2}}}dx\\
&= \sqrt{\frac{\alpha^2C^2}{1+\beta^2C^2}}+\frac{\alpha^2C^2}{(p-1)(1+\beta^2C^2)^2\sqrt{\frac{\alpha^2C^2}{1+\beta^2C^2}}}\quad\underbrace{\frac{\int_C^\infty\left(\frac{p+1}{\mathrm{e}\sqrt{\ell}}x\right)^{-(p+1)}}{\left(\frac{p+1}{\mathrm{e}\sqrt{\ell}}C\right)^{-(p+1)}}}.
\end{aligned}$$

Likewise, we seek $\nu>0$ such that

$$\sqrt{\frac{\alpha^2C^2}{1+\beta^2C^2}}+\frac{\alpha^2C^2\left(\frac{p+1}{\mathrm{e}\sqrt{\ell}}C\right)^{-(p+1)}}{(p-1)(1+\beta^2C^2)^2\sqrt{\frac{\alpha^2C^2}{1+\beta^2C^2}}}\leq\sqrt{\frac{\alpha^2\nu^2}{1+\beta^2\nu^2}}.$$

The solution satisfies

$$\nu \geq \frac{C^2}{\sqrt{C^2 - \frac{C^2}{p-1}\left(\frac{p+1}{\mathrm{e}\sqrt{\ell}}C\right)^{-(p+1)}}}.$$

The value $\nu = \frac{4\mathrm{e}\sqrt{\ell}}{p+1}$ satisfies this inequality for $C = \left(\frac{\mathrm{e}\sqrt{\ell}}{p+1}\right)\left(\frac{2p}{p-1}\right)^{1/(p+1)}$.

8.2.7 Proof of Theorem 4.14

We only prove Theorem 4.14 for $j = k$, as is the case for Theorem 4.7. Theorem 4.14 can be proved for other values of $1 \leq j < k$ by referring to Theorem 4.14 for a rank j truncated SVD. Since $\mathbf{\Omega}_1$ and $\mathbf{\Omega}_2$ are independent from each other, to bound expectations we in turn take expectations over $\mathbf{\Omega}_2$ and $\mathbf{\Omega}_1$:

$$\begin{aligned}
\mathbb{E}(\sigma_k(\hat{\mathbf{A}}_{\mathrm{SOR}})) &= \mathbb{E}_{\mathbf{\Omega}_1}\Big(\mathbb{E}_{\mathbf{\Omega}_2}\Big[\sigma_k(\hat{\mathbf{A}}_{\mathrm{SOR}})\Big]\Big) \\
&= \mathbb{E}_{\mathbf{\Omega}_1}\left(\mathbb{E}_{\mathbf{\Omega}_2}\left[\frac{\sigma_j}{\sqrt{1 + \|\mathbf{\Omega}_2\|_2^2\|\mathbf{\Omega}_1{}^{\dagger}\|_2^2\left(\frac{\sigma_{\ell-p+1}}{\sigma_j}\right)^4}}\right]\right) \qquad (8\text{-}49) \\
&\geq \mathbb{E}_{\mathbf{\Omega}_1}\left(\frac{\sigma_k}{\sqrt{1 + \nu_1^2\|\mathbf{\Omega}_1{}^{\dagger}\|_2^2\gamma_j^4}}\right) \geq \frac{\sigma_k}{\sqrt{1 + \nu^2\gamma_k^4}}.
\end{aligned}$$

The second line follows from Theorem 4.7, equation (4-23), and the last line from Proposition 4.10, and Proposition 4.12. The result in (4-49), likewise, follows by substituting (4-24) into the second line of equation (8-49).

8.2.8 Proof of Theorem 4.15

Similar to the proof of Theorem 4.14, we first take expectations over $\mathbf{\Omega}_2$ and next over $\mathbf{\Omega}_1$. By invoking Theorem 4.9, Propositions 4.11, and Proposition 4.13, we have

$$
\begin{aligned}
&\mathbb{E}\|\mathbf{A}-\hat{\mathbf{A}}_{\mathrm{SOR}}\|_{\varrho} = \mathbb{E}_{\boldsymbol{\Omega}_1}\mathbb{E}_{\boldsymbol{\Omega}_2}\|\mathbf{A}-\hat{\mathbf{A}}_{\mathrm{SOR}}\|_{\varrho} \\
&= \|\mathbf{A}_0\|_{\varrho} + \mathbb{E}_{\boldsymbol{\Omega}_1}\mathbb{E}_{\boldsymbol{\Omega}_2}\left(\sqrt{\frac{\alpha^2\|\boldsymbol{\Omega}_2\|_2^2\|{\boldsymbol{\Omega}_1}^{\dagger}\|_2^2}{1+\beta^2\|\boldsymbol{\Omega}_2\|_2^2\|{\boldsymbol{\Omega}_1}^{\dagger}\|_2^2}} + \sqrt{\frac{\eta^2\|\boldsymbol{\Omega}_2\|_2^2\|{\boldsymbol{\Omega}_1}^{\dagger}\|_2^2}{1+\tau^2\|\boldsymbol{\Omega}_2\|_2^2\|{\boldsymbol{\Omega}_1}^{\dagger}\|_2^2}}\right) \\
&\leq \|\mathbf{A}_0\|_{\varrho} + \mathbb{E}_{\boldsymbol{\Omega}_1}\left(\sqrt{\frac{\alpha^2\nu_1^2\|{\boldsymbol{\Omega}_1}^{\dagger}\|_2^2}{1+\beta^2\nu_1^2\|{\boldsymbol{\Omega}_1}^{\dagger}\|_2^2}} + \sqrt{\frac{\eta^2\nu_1^2\|{\boldsymbol{\Omega}_1}^{\dagger}\|_2^2}{1+\tau^2\nu_1^2\|{\boldsymbol{\Omega}_1}^{\dagger}\|_2^2}}\right) \\
&\leq \|\mathbf{A}_0\|_{\varrho} + \sqrt{\frac{\alpha^2\nu_1^2\nu_2^2}{1+\beta^2\nu_1^2\nu_2^2}} + \sqrt{\frac{\eta^2\nu_1^2\nu_2^2}{1+\tau^2\nu_1^2\nu_2^2}} \leq \|\mathbf{A}_0\|_{\varrho} + (\alpha+\eta)\nu.
\end{aligned}
\tag{8-50}
$$

Plugging values of α and η defined in Theorem 4.9, and ν into the last inequality gives the results.

Bibliography

[1] D. Achlioptas, *Database-friendly random projections*, in PODS '01 Proceedings of the twentieth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems, 2001, pp. 274–281.

[2] D. Achlioptas and F. Mcsherry, *Fast computation of low-rank matrix approximations*, Journal of the ACM, 54 (2007).

[3] N. Ailon and B. Chazelle, *The Fast Johnson–Lindenstrauss Transform and Approximate Nearest Neighbors*, SIAM Journal on Computing, 39 (2009), pp. 302—-322.

[4] T. Blumensath and M. E. Davies, *Iterative Thresholding for Sparse Approximations*, Journal of Fourier Analysis and Applications, 14 (2008), pp. 629–654.

[5] T. Bouwmans and E. H. Zahzah, *Robust PCA via Principal Component Pursuit: A review for a comparative evaluation in video surveillance*, Computer Vision and Image Understanding, 122 (2014), pp. 22–34.

[6] D. Brauckhoff, K. Salamatian, and M. Martin, *Applying PCA for Traffic Anomaly Detection: Problems and Solutions*, in Proceedings of INFOCOM 2009, apr 2009, pp. 2866 – 2870.

[7] J. Cai and E. Candès, *A Singular Value Thresholding Algorithm for Matrix Completion*, SIAM Journal on Optimization, 20 (2008), pp. 1956—-1982.

[8] D. Calvetti, L. Reichel, and D. Sorensen, *An implicitly restarted Lanczos method for large symmetric eigenvalue problems*, ETNA, 2 (1994), pp. 1–21.

[9] E. J. Candès, X. Li, Y. Ma, and J. Wright, *Robust principal component analysis?*, Journal of the ACM, 58 (2011), pp. 1–37.

[10] E. J. Candès and T. Tao, *Decoding by linear programming*, IEEE Transactions on Information Theory, 51 (2005), pp. 4203–4215.

[11] T. F. Chan, *Rank revealing QR factorizations*, Linear Algebra and its Applications, 88–89 (1987), pp. 67–82.

[12] S. Chandrasekaran and I. Ipsen, *On rank-revealing QR factorizations*, SIAM. J. Matrix Anal. & Appl., 15 (1994), pp. 946—977.

[13] V. Chandrasekaran, P. Parrilo, and A. Willsky, *Latent variable graphical model selection via convex optimization*, The Annals of Statistics, 40 (2012), pp. 1935—-1967.

[14] V. Chandrasekaran, S. Sanghavi, P. a. Parrilo, and A. S. Willsky, *Rank-Sparsity Incoherence for Matrix Decomposition*, SIAM Journal on Optimization, 21 (2009), pp. 572–596.

[15] K. L. Clarkson and D. P. Woodruff, *Low-rank approximation and regression in input sparsity time*, J. ACM, 63 (2017), pp. 54:1–54:45.

[16] D. D. Gross, Y.-K. Liu, S. Flammia, S. Becker, and J. Eisert, *Quantum state tomography via compressed sensing*, Phys. Rev. Lett., 105 (2010), p. 150401.

[17] R. de Lamare and R. Sampaio-Neto, *Adaptive Reduced-Rank Processing Based on Joint and Iterative Interpolation, Decimation, and Filtering*, IEEE Transactions on Signal Processing, 57 (2009), pp. 2503–2514.

[18] J. Demmel, *Applied Numerical Linear Algebra*, SIAM, (1997).

[19] J. Demmel, I. Dumitriu, and O. Holtz, *Fast linear algebra is stable*, Numerische Mathematik, 108 (2007), pp. 59—91.

[20] J. Demmel, L. Grigori, M. Gu, and H. Xiang, *Communication Avoiding Rank Revealing QR Factorization with Column Pivoting*, SIAM J. Matrix Anal. & Appl., 36 (2015), pp. 55—-89.

[21] J. Demmel, L. Grigori, M. Hoemmen, and J. Langou, *Communication-optimal Parallel and Sequential QR and LU Factorizations*, SIAM J. Sci. Comput., 34 (2012), pp. A206—-A239.

[22] A. Deshpande and S. Vempala, *Adaptive Sampling and Fast Low-Rank Matrix Approximation*, Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, 4110, pp. 292–303.

[23] J. Dongarra, S. Tomov, P. Luszczek, J. Kurzak, M. Gates, I. Yamazaki, H. Anzt, A. Haidar, and A. Abdelfattah, *With*

Extreme Computing, the Rules Have Changed, Computing in Science and Engineering, 19 (2017), pp. 52–62.

[24] D. L. Donoho, *High-dimensional data analysis: The curses and blessings of dimensionality*, in AMS conference on Math Challenges of 21st Centuary, 2000.

[25] P. Drineas, R. Kannan, and M. Mahoney, *Fast Monte Carlo Algorithms for Matrices II: Computing a Low-Rank Approximation to a Matrix*, SIAM J. Compu., 36, pp. 158—-183.

[26] J. A. Duersch and M. Gu, *Randomized QR with Column Pivoting*, SIAM J. Sci. Comput., 39 (2017), pp. C263–C291.

[27] R. Dunia and S. J. Qin, *A Subspace Approach to Multidimensional Fault Identification and Reconstruction*, American Institute of Chemical Engineers (AIChE) Journal, 44 (1998), pp. 1813—-1831.

[28] C. Eckart and G. Young, *The approximation of one matrix by another of lower rank*, Psychometrika, 1 (1936), pp. 211–218.

[29] M. Fazel, H. Hindi, and S. Boyd, *A rank minimization heuristic with application to minimum order system approximation*, in Proceedings of the American Control Conference, vol. 6, 2001, pp. 4734—-4739.

[30] 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.

[31] R. D. Fierro and P. C. Hansen, *Low-rank revealing UTV decompositions*, Numerical Algorithms, 15 (1997), pp. 37—55.

[32] R. D. Fierro, P. C. Hansen, and H. P. S. K., *UTV Tools: Matlab templates for rank-revealing UTV decompositions*, Numerical Algorithms, 20 (1999), pp. 165—194.

[33] A. Frieze, R. Kannan, and S. Vempala, *Fast monte-carlo algorithms for finding low-rank approximations*, J. ACM, 51 (2004), pp. 1025–1041.

[34] A. S. Georghiades, P. N. Belhumeur, and D. J. Kriegman, *From few to many: illumination cone models for face recognition under variable lighting and pose*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 23 (2001), pp. 643–660.

[35] G. H. Golub and C. F. van Loan, *Matrix Computations*, 3rd ed., Johns Hopkins Univ. Press, Baltimore, MD, 1996.

[36] M. Gu, *Subspace Iteration Randomization and Singular Value Problems*, SIAM J. Sci. Comput., 37, pp. A1139–A1173.

[37] 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.

[38] E. Hale, W. Yin, and Y. Zhang, *Fixed-Point Continuation for ℓ_1-Minimization: Methodology and Convergence*, SIAM Journal on Optimization, 19 (2008), pp. 1107—-1130.

[39] 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.

[40] P. C. Hansen, *Rank-Deficient and Discrete Ill-Posed Problems: Numerical Aspects of Linear Inversion*, (SIAM, 1998).

[41] Y. P. Hong and C.-T. Pan, *Rank-Revealing QR Factorizations and the Singular Value Decomposition*, Math. of Compu., 58 (1992), pp. 213–232.

[42] H. Hotelling, *Relations between two sets of variates*, Biometrica, 28 (1936), pp. 321–377.

[43] L. Huang, X. Nguyen, M. Garofalakis, M. Jordan, A. D. Joseph, and N. Taft, *In-network pca and anomaly detection*, Tech. Report UCB/EECS-2007-10, University of California, Berkeley, Jan 2007.

[44] P. Indyk and R. Motwani, *Approximate nearest neighbors: towards removing the curse of dimensionality*, in STOC '98 Proceedings of the 30th annual ACM symp. on Theory of computing, 1998, pp. 604–613.

[45] J. Jack Dongarra, I. Foster, G. Fox, W. Gropp, K. Kennedy, L. Torczon, and A. White, *The Sourcebook of Parallel Computing*, Morgan Kaufmann, 1st ed., 2002.

[46] J. E. Jackson, *A User's Guide to Principal Components*, New York: Wiley, 1991.

[47] J. E. Jackson and G. S. Mudholkar, *Control procedures for residuals associated with principal component analysis*, Technometrics, 21 (1979), pp. 341–349.

[48] N. L. Johnson, S. Kotz, and N. Balakrishnan, *Continuous Univariate Distributions*, vol. 2, 2nd ed, New York, Wiley, 1995.

[49] W. Johnson and J. Lindenstrauss, *Extensions of Lipschitz mappings into a Hilbert space*, in Contemporary Mathematics, vol. 26, 1984, pp. 189—-206.

[50] I. T. Jolliffe, *Principal Component Analysis*, Springer, 2nd ed., 2002.

[51] M. F. Kaloorazi and R. C. de Lamare, *Switched-Randomized Robust PCA for Background and Foreground Separation in Video Surveillance*, in IEEE SAM 2016, Jul 2016.

[52] ——, *Anomaly Detection in IP Networks Based on Randomized Subspace Methods*, in ICASSP 2017, Mar 2017.

[53] ——, *Low-Rank and Sparse Matrix Recovery Based on a Randomized Rank-revealing Decomposition*, in 22nd Intl Conf. on DSP, UK, Aug 2017.

[54] ——, *Subspace-Orbit Randomized Decomposition for Low-rank Matrix Approximation*, Arxiv preprint arXiv:1804.00462v1, (2018).

[55] T. Katayama, *Subspace Methods for System Identification*, Springer-Verlag London, 2005.

[56] A. Lakhina, M. Crovella, and C. Diot, *Diagnosing Network-Wide Traffic Anomalies*, in proceedings of ACM SIGCOMM, aug 2004.

[57] R. M. Larsen, *Efficient algorithms for helioseismic inversion*, PhD Thesis, University of Aarhus, Denmark (1998).

[58] L. Li, W. Huang, I.-H. Gu, and Q. Tian, *Statistical Modeling of Complex Backgrounds for Foreground Object Detection*, IEEE Transactions on Image Processing, 13 (2004), pp. 1459–1472.

[59] Z. Lin, R. Liu, and Z. Su, *Linearized Alternating Direction Method with Adaptive Penalty for Low-Rank Representation*, in NIPS 2011, no. 1, 2011, pp. 1–9.

[60] M. Mardani, G. Mateos, and G. B. Giannakis, *Dynamic anomalography: Tracking network anomalies via sparsity and low rank*, IEEE Journal on Selected Topics in Signal Processing, 7 (2013), pp. 50–66.

[61] P.-G. Martinsson, G. Quintana-Orti, and N. Heavner, *randUTV: A blocked randomized algorithm for computing a rank-revealing UTV factorization*, Arxiv preprint arXiv:1703.00998, (2017).

[62] P.-G. Martinsson, G. Quintana Ortí, N. Heavner, and R. van de Geijn, *Householder QR Factorization With Randomization for Column Pivoting (HQRRP)*, SIAM J. Sci. Comput., 39 (2017), pp. C96–C115.

[63] J. Matousek, *Lectures on Discrete Geometry*, vol. 2, New York, Springer-Verlag, 2002.

[64] L. Mirsky, *Symmetric gauge functions and unitarily invariant norms*, Quarterly Journal of Mathematics, 11 (1960), pp. 50–59.

[65] B. K. Natarajan, *Sparse Approximate Solutions to Linear Systems*, SIAM Journal on Computing, 24 (1995), pp. 227—-234.

[66] J. Nelson and H. L. Nguyen, *Osnap: Faster numerical linear algebra algorithms via sparser subspace embeddings*, in Proc. of 54th Annual Symp. on FOCS '13, 2013, pp. 117–126.

[67] T.-H. Oh, Y. Matsushita, Y.-W. Tai, and I. S. Kweon, *Fast randomized singular value thresholding for low-rank optimization*, IEEE Trans. Pattern Anal. Mach. Intell., 40 (2017), pp. 376–391.

[68] M. Rahmani and G. Atia, *Coherence Pursuit: Fast, Simple, and Robust Principal Component Analysis*, IEEE Trans. Signal Process., 65 (2017), pp. 6260–6275.

[69] ——, *High Dimensional Low Rank Plus Sparse Matrix Decomposition*, IEEE Trans. Signal Process., 65 (2017), pp. 2004–2019.

[70] A. Rao and S. Noushathb, *Subspace methods for face recognition*, Computer Science Review, 4 (2010), pp. 1—-17.

[71] V. Rokhlin, A. Szlam, and M. Tygert, *A randomized algorithm for principal component analysis*, SIAM Journal on Matrix Analysis and Applications (SIMAX), 31 (2009), pp. 1100 – 1124.

[72] S. Roweis, *EM algorithms for PCA and SPCA*, in conference on advances in neural information processing systems 10, 1997, pp. 626–632.

[73] M. Rudelson and R. Vershynin, *Sampling from large matrices: An approach through geometric functional analysis*, J. ACM, 54 (2007).

[74] T. Sarlós, *Improved approximation algorithms for large matrices via random projections*, in 47th Ann. IEEE Symp. on Foundations of Computer Science. FOCS '06., vol. 1, Oct. 2006.

[75] R. O. Schmidt, *Multiple emitter location and signal parameter estimation*, IEEE Transactions on Antennas and Propagation, 34 (1986), pp. 276–280.

[76] M. Soltanolkotabi, E. Elhamifar, and E. J. Candès, *Robust subspace clustering*, Annals of Statistics, 42 (2014), pp. 669—-699.

[77] N. Srebro, *Learning with Matrix Factorizations*, PhD thesis, Massachusetts Institute of Technology, 2004.

[78] N. Srebro and T. Jaakkola, *Generalization error bounds for collaborative prediction with low-rank matrices*, in Advances In Neural Information Processing Systems 17, MIT Press, 2005, pp. 5–27.

[79] G. Stewart, *An updating algorithm for subspace tracking*, IEEE Transactions on Signal Processing, 40 (1992), pp. 1535–1541.

[80] G. W. Stewart, *The QLP Approximation to the Singular Value Decomposition*, SIAM J. Sci. Comput., 20, pp. 1336–1348.

[81] ——, *Updating a Rank-Revealing ULV Decomposition*, SIAM Journal on Matrix Analysis and Applications, 14 (1993), pp. 494—-499.

[82] ——, *Matrix Algorithms: Volume 1: Basic Decompositions*, (SIAM, Philadelphia, PA, 1998).

[83] R. Thompson, *Principal submatrices IX: Interlacing inequalities for singular values of submatrices*, Linear Algebra and its App., 5 (1972), pp. 1—-12.

[84] R. Tibshirani, *Regression shrinkage and selection via the lasso*, Journal of the Royal Statistical Society, Series B, 58 (1996), pp. 267–288.

[85] K. C. Toh and S. Yun, *An accelerated proximal gradient algorithm for nuclear norm regularized linear least squares problems*, Pacific Journal of Optimization, 6 (2010), pp. 615–640.

[86] J. Tropp, A. Yurtsever, M. Udell, and V. Cevher, *Practical Sketching Algorithms for Low-Rank Matrix Approximation*, SSIAM J. Matrix Anal. & Appl., 38 (2017), pp. 1454–1485.

[87] J. A. Tropp, *Improved analysis of the subsampled randomized Hadamard transform*, Adv. Adapt. Data Anal, 3 (2011), pp. 115–126.

[88] O. Troyanskaya, M. Cantor, G. Sherlock, P. Brown, T. Hastie, R. Tibshirani, David, D. Botstein, and R. B. Altman, *Missing value estimation methods for dna microarrays*, Bioinformatics, 17 (2001), pp. 520–525.

[89] L. Vandenberghe and S. Boyd, *Semidefinite Programming*, SIAM Review, 38 (1996), pp. 49—-95.

[90] Y. Vardi, *Network Tomography: Estimating Source-Destination Traffic Intensities from Link Data*, Journal of the American Statistical Association, (1996), pp. 365–377.

[91] B. Victor, K. Bowyer, and S. Sarkar, *An evaluation of face and ear biometrics*, in ICPR 2002, vol. 1, pp. 429–432.

[92] J. H. Wilkinson, *The Algebraic Eigenvalue Problem*, Oxford, (1965).

[93] J. Wright, Y. Peng, Y. Ma, A. Ganesh, and S. Rao, *Robust Principal Component Analysis: Exact Recovery of Corrupted Low-Rank Matrices*, in NIPS 2009, Dec 2009, pp. 2080–2088.

[94] J. Wright, A. Yang, A. Ganesh, S. Sastry, and Y. Ma, *Robust face recognition via sparse representation*, IEEE Trans. Pattern Anal. Mach. Intell., 31 (2009), pp. 210–227.

[95] X. Yuan and J. Yang, *Sparse and low-rank matrix decomposition via alternating direction methods*, Pacific Journal of Optimization, 9 (2009), pp. 167–180.

[96] Y. Zhang, Z. Ge, A. Greenberg, and M. Roughan, *Network Anomography*, in Proceedings of the 5th ACM SIGCOMM conference on Internet Measurement (IMC '05), oct 2005.

[97] T. Zhou and D. Tao, *Godec: Randomized low-rank & sparse matrix decomposition in noisy case*, in ICML, 2011.

[98] P. Zikopoulos, D. deRoos, K. Parasuraman, T. Deutsch, J. Giles, and D. Corrigan, *Harness the Power of Big Data The IBM Big Data Platform*, McGraw-Hill Education, 2012.