SJÄLVSTÄNDIGA ARBETEN I MATEMATIK

MATEMATISKA INSTITUTIONEN, STOCKHOLMS UNIVERSITET

Advancing Dimensionality Reduction: The Potential of Spectral Density-Based Stopping Criteria

av

Alexander Todoran

2025 - No L12

MATEMATISKA INSTITUTIONEN, STOCKHOLMS UNIVERSITET, 106 91 STOCKHOLM

Advancing Dimensionality Reduction: The Potential of Spectral Density-Based Stopping Criteria

Alexander Todoran

Självständigt arbete i matematik 15 högskolepoäng, grundnivå

Handledare: Yishao Zhou

2025

Abstract

In the rapidly evolving field of data analytics, Principal Component Analysis (PCA) remains a fundamental technique for dimensionality reduction, particularly in high-dimensional datasets common in machine learning and signal processing. The performance of PCA is critically influenced by its stopping criterion, which determines the extent of dimensionality reduction. Building upon the work of Ubaru et al. (2018)—who introduced a randomized algorithm based on Krylov subspace methods and the Lanczos algorithm, employing an information criterion as the stopping rule—this study explores an alternative strategy: Spectral Density Estimation. We demonstrate that this approach yields lower approximation error compared to the original criterion, reduces the risk of overfitting, and improves computational efficiency by decreasing runtime. These results position Spectral Density Estimation as a promising stopping criterion and pave the way for the development of adaptable, domain-specific dimensionality reduction techniques in deep learning and related fields.

Sammanfattning

Principalkomponentanalys är en grundläggande metod för att reducera dimensionalitet i datamängder med många variabler. Resultatet påverkas starkt av vilket konvergenskriterium som används för att avgöra när extraktion av antalet faktorer är optimalt. I ett tidigare arbete av Ubaru med flera (2018) föreslogs en slumpmässig algoritm baserad på Krylov-delrum och Lanczos-metoden, där ett informationsmått styr konvergensen. I denna studie undersöker vi ett alternativt sätt: att använda skattningar av spektral täthet. Det leder till mindre fel, minskad risk för överanpassning och snabbare analys. Metoden framstår därmed som ett lovande konvergenskriterium och kan anpassas till olika tillämpningar inom djupinlärning och annan högdimensionell dataanalys.

Acknowledgments

I would like to express my deepest gratitude to my supervisor Prof. Yishao Zhou for her invaluable guidance, support, and mentorship throughout the course of my bachelor's thesis in mathematics. Her expertise and insights have been instrumental in the successful completion of this project.

Contents

1 Introduction 6

2 Variational Characterizations of Eigenvalues 7

2.1 Courant–Fischer Theorem 7

2.2 Weyl's Inequality 8

2.3 Cauchy's Interlacing Theorem 8

3 Theoretical Foundations of Principal Component Analysis (PCA) 9

3.1 Evolution of Krylov Subspace Methods: From Krylov to Lanczos 10

3.2 Krylov Subspace Methods and Its Application in PCA 15

3.3 Relationship between Lanczos Algorithm and The Conjugate Gradient Method 18

3.4 The Best Low-Rank Matrix Approximation (Eckart–Young–Mirsky Theorem) 22

4 The Ubaru algorithm 24

4.1 Theoretical Advancements and Mathematical Innovations behind the Ubaru Algorithm 25

4.2 Stopping Criteria of the Ubaru Algorithm 26

4.3 Comments and Insights 29

4.4 Implementation of the Ubaru Algorithm in Matlab 31

4.4.1 The Ubaru Algorithm Depiction 31

4.4.2 Reproducing Results Using the Ubaru Algorithm 32

5 Review of Stopping Criteria 34

5.1 Introduction 34

5.2 Information Theoretic Criterion 35

5.3 Spectral Density Estimation Criterion . . . 36

5.4 Matrix Perturbation-based Criterion . . . 37

5.5 Optimizing PCA with Cross-Validation: A Mathematical Perspective . . . 38

5.6 Justification for the Use of Hybrid Spectral Density Estimation Criterion . . . 40

6 The Revised Ubaru Algorithm 42

6.1 Experiments . . . 44

6.1.1 Experiment Setup . . . 44

6.2 Results of Experiments . . . 44

6.3 Comment . . . 45

7 Discussion and Conclusions 46

A Matlab code 48

A.1 The Ubaru Algorithm . . . 48

A.2 The Revised Ubaru Algorithm . . . 51

B The Components and Reasoning Behind Spectral Density Estimation Criteria 53

B.1 Kernel Density Estimation . . . 53

B.2 Key Theoretical Results for Spectral Density Estimation . . . 56

B.3 Real World Example: Image Processing . . . 58

List of Tables

1 Comparison of Dissertation and Reproduction Results 33

2 Comparison of Algorithm Performance on Synthetic and Real Data 45

1 Introduction

High-dimensional data are becoming increasingly ubiquitous in the digital age, and effectively analyzing these datasets has become a pressing concern across numerous fields, from machine learning and signal processing to image processing and data analysis.

Covariance matrices have emerged as a reliable means of encapsulating interactions within high-dimensional data, with Principal Component Analysis (PCA) being a favored method for identifying the dominant subspace within these matrices to reduce multicollinearity and orthogonalize the data set. [1]

However, the efficacy of PCA is contingent upon the stopping criterion used, a feature often overlooked despite its pivotal role.

This study seeks to cast light upon this dimensionality reduction process by investigating alternative stopping criteria based on the theoretical underpinnings of a recently published Ubaru. The researchers have proposed an innovative approach that combines dimensionality estimation and approximation in one cost-effective package. [2]

Their method, utilizing the Krylov subspace methodology, diverges from traditional PCA by operating on a model selection framework, employing a selection criterion derived from random matrix perturbation theory.[2]

This paper will begin by diving into PCA and its stopping criterion—which determines when the algorithm has extracted enough information from the dataset—and then proceed to explore Krylov subspace methodologies and their unique features. We will then review the literature on existing stopping criteria, weighing the strengths and weaknesses of each.

By centering the study on a groundbreaking Ubaru that merges dimensionality estimation and approximation using a novel selection criterion, the intention is to bring forth a nuanced understanding of the stopping criterion's role within PCA.

This exploration will not only contribute to the broader academic discourse on dimensionality reduction but also provide practical insights that could guide researchers and data scientists in their quest to effectively analyze high-dimensional data. This study ultimately strives to illuminate the path towards more efficient and versatile dimensionality reduction algorithms.

2 Variational Characterizations of Eigenvalues

In this section we examine symmetric matrices and their eigenvalues—a topic that plays a central role both in establishing optimal low-rank approximations and in the analysis of PCA. We derive the well-known minimax and maximin characterizations (the Courant–Fischer theorem) and then deduce several useful consequences, including Weyl's inequality and Cauchy's interlacing theorem. Our exposition follows the approach outlined in [3].

Let

$$\mathcal{S}^n := \{A \in \mathbb{R}^{n\times n} : A^\top = A\},$$

and define the inner product $\langle x, y\rangle = y^\top x$ for $x, y \in \mathbb{R}^n$. For any $A \in \mathcal{S}^n$, all eigenvalues are real and there exists an orthogonal matrix Q and a diagonal matrix D such that

$$A = QDQ^\top,$$

with the eigenvalues (the diagonal entries of D) ordered as

$$\lambda_1(A) \le \lambda_2(A) \le \cdots \le \lambda_n(A).$$

A direct consequence is the Rayleigh quotient bound: for any $x \in \mathbb{R}^n$,

$$\lambda_1(A)\|x\|_2^2 \le \langle Ax, x\rangle \le \lambda_n(A)\|x\|_2^2. \tag{1}$$

In particular, if $\|x\|_2 = 1$ then

$$\lambda_1(A) = \min_{\|x\|_2=1} \langle Ax, x\rangle \quad \text{and} \quad \lambda_n(A) = \max_{\|x\|_2=1} \langle Ax, x\rangle.$$

2.1 Courant–Fischer Theorem

Theorem 1 (Courant–Fischer)**.** *For $A \in \mathcal{S}^n$ and any $k = 1, \ldots, n$,*

$$\lambda_k(A) = \min_{\substack{V\subset\mathbb{R}^n \\ \dim(V)=k}} \max_{\substack{x\in V \\ \|x\|_2=1}} \langle Ax, x\rangle = \max_{\substack{W\subset\mathbb{R}^n \\ \dim(W)=n-k+1}} \min_{\substack{x\in W \\ \|x\|_2=1}} \langle Ax, x\rangle. \tag{2}$$

Proof. Let $q_1, \ldots, q_n$ be an orthonormal eigenbasis of A with corresponding eigenvalues $\lambda_1(A), \ldots, \lambda_n(A)$. First, set

$$U = \operatorname{span}\{q_1, \ldots, q_k\}.$$

For any unit vector $x \in U$, writing $x = \sum_{j=1}^k c_j q_j$ (thus $\sum c_j^2 = 1$) yields

$$\langle Ax, x\rangle = \sum_{j=1}^{k} \lambda_j(A) c_j^2 \le \lambda_k(A).$$

Thus,

$$\max_{\substack{x\in U\\ \|x\|_2=1}} \langle Ax, x\rangle \le \lambda_k(A),$$

and hence

$$\min_{\dim(V)=k} \max_{\substack{x\in V\\ \|x\|_2=1}} \langle Ax, x\rangle \le \lambda_k(A).$$

Conversely, for an arbitrary k-dimensional subspace V, consider the subspace

$$W = \operatorname{span}\{q_k, q_{k+1}, \dots, q_n\},$$

which has dimension $n-k+1$. Since $\dim(V\cap W) \ge 1$, there exists a unit vector $x \in V \cap W$ satisfying $\langle Ax, x\rangle \ge \lambda_k(A)$. Taking the minimum over all such V completes the proof. The second equality can be obtained by considering the eigenvalues of $-A$. □

2.2 Weyl's Inequality

Theorem 2 (Weyl's Inequality)**.** *For any $A, B \in \mathcal{S}^n$ and every $k = 1, \dots, n$,*

$$\lambda_k(A) + \lambda_1(B) \le \lambda_k(A+B) \le \lambda_k(A) + \lambda_n(B).$$

Proof. Using the Courant–Fischer representation, for any k-dimensional subspace V we have

$$\lambda_k(A+B) = \min_{\dim(V)=k} \max_{\substack{x\in V\\ \|x\|_2=1}} \langle (A+B)x, x\rangle.$$

Since for each unit vector x it holds that

$$\langle (A+B)x, x\rangle = \langle Ax, x\rangle + \langle Bx, x\rangle \le \langle Ax, x\rangle + \lambda_n(B),$$

it follows that

$$\lambda_k(A+B) \le \lambda_k(A) + \lambda_n(B).$$

A similar argument, applied to $-B$ (and noting that $\lambda_n(-B) = -\lambda_1(B)$), yields the lower bound. □

2.3 Cauchy's Interlacing Theorem

Theorem 3 (Cauchy's Interlacing Theorem)**.** *Let $A \in \mathcal{S}^n$ and let A_s be an $s \times s$ principal submatrix of A, obtained by deleting $n-s$ rows and the corresponding $n-s$ columns, where $1 < s < n$. Then, for $k = 1, \dots, s$,*

$$\lambda_k(A) \le \lambda_k(A_s) \le \lambda_{k+n-s}(A).$$

Proof. Let S be the index set of the rows and columns retained in A_s. For any $x \in \mathbb{R}^s$, define $\tilde{x} \in \mathbb{R}^n$ by

$$\tilde{x}_i = \begin{cases} x_i, & i \in S, \\ 0, & \text{otherwise.} \end{cases}$$

For any k-dimensional subspace $V \subset \mathbb{R}^s$, define

$$\tilde{V} = \{\tilde{x} \mid x \in V\}.$$

Since $\dim(\tilde{V}) = \dim(V)$, we apply the Courant–Fischer theorem. For any linear subspace V of $\mathbb{R}^s$ with $\dim V = k$, we obtain

$$\max_{\substack{x \in V \\ \|x\|_2=1}} \langle A_s x, x \rangle = \max_{\substack{x \in V \\ \|x\|_2=1}} \langle A\tilde{x}, \tilde{x} \rangle = \max_{\substack{\tilde{x} \in \tilde{V} \\ \|\tilde{x}\|_2=1}} \langle A\tilde{x}, \tilde{x} \rangle \geq \lambda_k(A).$$

Taking the minimum over all k-dimensional subspaces V yields $\lambda_k(A_s) \geq \lambda_k(A)$.

Similarly, for any linear subspace $V \subset \mathbb{R}^s$ with $\dim V = s - k + 1 = n - (k + n - s) + 1$,

$$\min_{\substack{x \in V \\ \|x\|_2=1}} \langle A_s x, x \rangle = \min_{\substack{x \in V \\ \|x\|_2=1}} \langle A\tilde{x}, \tilde{x} \rangle = \min_{\substack{\tilde{x} \in \tilde{V} \\ \|\tilde{x}\|_2=1}} \langle A\tilde{x}, \tilde{x} \rangle \geq \lambda_{k+n-s}(A).$$

Taking the maximum over all $(s - k + 1)$-dimensional subspaces V gives $\lambda_k(A_s) \leq \lambda_{k+n-s}(A)$, proving the theorem. □

The variational characterizations discussed above not only establish a rigorous mathematical foundation for eigenvalue behavior but also directly motivate the practical applications in data reduction. In the next section, we leverage these theoretical insights to explain how PCA transforms high-dimensional data into a more tractable form.

3 Theoretical Foundations of Principal Component Analysis (PCA)

Principal Component Analysis (PCA) serves as a cornerstone statistical methodology for reducing dimensionality, enhancing data visualization, and mitigating noise within datasets. At its core, PCA seeks to reconstitute a potentially correlated set of variables into a new ensemble of uncorrelated variables, termed principal components. These components are essentially linear amalgamations of the original variables, strategically chosen to encapsulate the maximum variance present within the dataset. Such a transformation not only simplifies the data structure but also aids in uncovering the underlying patterns by focusing on the most significant features.[4]

From a mathematical standpoint, PCA's objective is to diagonalize the covariance matrix associated with the dataset. This process entails calculating the eigenvectors (principal components) and eigenvalues of the covariance matrix, where the eigenvectors delineate the directions of the newly defined space, and the eigenvalues quantify the variance captured by each principal component. Conventionally, the eigenvalues are arranged in descending order, with the selection of principal components being based on the largest eigenvalues. This approach ensures that the retained components account for a substantial portion of the dataset's total variance. The eigenvalue criterion for stopping typically involves retaining those components whose eigenvalues surpass a defined threshold, often set to represent a desired proportion of the total variance (for instance, 95%), thereby achieving an efficient dimensionality reduction while preserving essential data characteristics[5]

3.1 Evolution of Krylov Subspace Methods: From Krylov to Lanczos

Krylov subspace methods, developed from the foundational work of Krylov and later refined by Hessenberg, Arnoldi, and Lanczos, offer an elegant and efficient framework for solving large-scale linear systems and eigenvalue problems. Rooted in projection techniques such as the Galerkin approach, these methods have become central in computational mathematics. In this section we present an historic expose and thereafter give proofs of central facts.

The Krylov Subscpace methodolgy was published 1931. In the article, Krylov describes a new procedure for computing the characteristic polynomial of an arbitrary square matrix.[6]

Given a matrix $A \in \mathbb{R}^{n \times n}$ with characteristic polynomial:

$$p_n(t) = t^n - \mu_{n-1}t^{n-1} - \cdots - \mu_1 t - \mu_0,$$

and a nonzero vector v_1 of grade n, the vectors $v_1, Av_1, \ldots, A^{n-1}v_1$ are linearly independent, whereas $v_1, Av_1, \ldots, A^n v_1$ are linearly dependent, satisfying:

$$A^n v_1 - \mu_{n-1}A^{n-1}v_1 - \cdots - \mu_1 Av_1 - \mu_0 v_1 = 0.$$

This relationship relies on the generic property that a randomly chosen vector $v_1 \in \mathbb{R}^n$ is of grade n, meaning the Krylov sequence $v_1, Av_1, \ldots, A^{n-1}v_1$ is linearly independent, while $v_1, Av_1, \ldots, A^n v_1$ is linearly dependent. This property holds for almost all v_1, as the set of vectors generating a sequence of lower.grade has measure zero, ensuring the Krylov subspace $\mathcal{K}_n(A, v_1) = \text{span}\{v_1, Av_1, \ldots, A^{n-1}v_1\}$ achieves dimension n. This establishes the basis for forming the Krylov sequence:

$$v_{j+1} = Av_j, \quad j = 1, \ldots, n.$$

This sequence leads to the equation:

$$\mu_0 v_1 + \mu_1 v_2 + \cdots + \mu_{n-1} v_n = v_{n+1},$$

where solving for the coefficients μ_i involves an ill-conditioned linear system with the matrix $[v_1, v_2, \ldots, v_n]$ and right-hand side v_{n+1}.

A critical insight from Krylov's method is reducing A to a companion form H via a similarity transformation, expressed as:

$$AV = VH, \quad \text{with} \quad V^{-1}AV = H,$$

where $V = [v_1, v_2, \ldots, v_n]$ is an invertible matrix composed of column-vectors v_i, and H is the companion matrix:

$$H = \begin{pmatrix} 0 & 0 & \cdots & 0 & \mu_0 \\ 1 & 0 & \cdots & 0 & \mu_1 \\ 0 & 1 & \cdots & 0 & \mu_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & \mu_{n-1} \end{pmatrix}.$$

This transformation was groundbreaking in 1931, offering a novel way to compute eigenvalues and eigenvectors at a time when such tasks were daunting for even small matrices, marking a significant departure from the Leverrier method established in 1840. [6]

The near linear dependence of the v_i's in Krylov's method will cause numerical difficulties in most practical circumstances.[6]

A solution to this problem was offered by Hessenberg in 1942.

The essence of the Hessenberg method, developed to address numerical difficulties due to near linear dependence in Krylov sequences, can be mathematically condensed as follows:

Hessenberg proposed a technique to transform a matrix A into an upper Hessenberg form H, characterized by $h_{ij} = 0$ for $i > j + 1$, modifying the Krylov sequence generation to enhance numerical stability. For each vector v_j in the sequence, compute $v_{j+1} = Av_j - \sum_{i=1}^{j} h_{ij}v_i$, where h_{ij} are coefficients ensuring v_{j+1} is orthogonal to a selected set of vectors $g_1, g_2, ..., g_j$, and normalize v_{j+1} with a scaling factor $h_{j+1,j}$.

Here we ensure that $v_{j+1} \perp g_1, g_2, ..., g_j$ by setting $g_{j+1}^T v_{j+1} = 1$, where g_i typically represents the i-th column of the identity matrix. Now let us state this in algorithmic terms.

Given a Matrix $A \in \mathbb{R}^{n\times n}$ where $m = n - 1$, we can compute the Hessenberg matrix. For each $j = 1$ to m, follow these steps:

1. Compute $h_{ij} = e_i^T(Av_j)$, for $i = 1$ to j.
2. Update v as $v = Av_j - \sum_{i=1}^{j} h_{ij}v_i$.
3. Normalize v_{j+1} by setting $h_{j,j+1} = e_{j+1}^T v$ and $v_{j+1} = v/h_{j+1,j}$.

This procedure results in $V = [v_1, v_2, ..., v_n]$ and an upper Hessenberg matrix H, where H contains the nonzero entries h_{ij} as specified. The transformation $AV = VH$ effectively reduces A to Hessenberg form via a similarity transformation when V is invertible, improving numerical stability for eigenvalue and eigenvector computations.[6]

The realization came that Hessenberg's method for matrix transformation might result in numerical instability, rendering the calculations for eigenvalues and eigenvectors unreliable. This instability arises because the transformation matrix V can become ill-conditioned, potentially causing significant errors in the computational results. To address this issue, the Arnoldi process was developed, employing an orthogonal matrix V to ensure numerical stability.[6]

The Arnoldi method transforms a matrix A into Hessenberg form by constructing an orthonormal basis of the Krylov subspace:

- for each $j = 1, 2, \ldots, m$:
  - Compute the scalar products: $h_{ij} = v_i^T Av_j$ for $i = 1$ to j.
  - Update the vector v: $v = Av_j - \sum_{i=1}^{j} h_{ij}v_i$.
  - Compute the norm of v: $h_{jj+1} = \|v\|_2$.

- Normalize and obtain the next vector: $v_{j+1} = v/h_{jj+1}$.

Arnoldi's major contribution was introducing a new perspective: viewing Lanczos's methods for solving linear systems and eigenvalue problems as types of projection methods. This innovative approach significantly advanced the field at the time.[6]

Building on Arnoldi's pivotal realization that Lanczos's methods could be seen through the lens of projection techniques for both linear systems and eigenvalue problems, we can directly connect this insight to the Galerkin projection approach. Arnoldi's work laid the groundwork for appreciating the power of projection methods in simplifying complex computational tasks. By employing an orthonormal basis $V = [v_1, v_2, \ldots, v_m]$ for a chosen subspace K, Arnoldi's method effectively demonstrates a practical application of projection principles to approximate solutions to linear systems, embodying the essence of the Galerkin condition.[6]

The Galerkin method, which seeks approximate solutions $\tilde{x} = Vy$ within the subspace K, echoes Arnoldi's emphasis on projection's utility by enforcing the condition $V^T AVy = V^T b$. This not only achieves a dimensionality reduction in solving $Ax = b$ but also provides a more computationally feasible pathway to accurate approximations. Thus, Arnoldi's conceptual contribution serves as a bridge to the Galerkin approach, highlighting a unified strategy of leveraging projections to tackle both linear and eigenvalue problems with increased efficiency and insight.[6]

To solve a linear system $Ax = b$, where $A \in \mathbb{R}^{n \times n}$, $b \in \mathbb{R}^n$, and $x \in \mathbb{R}^n$, and given a subspace K of dimension $m \leq n$ with an orthonormal basis $V = [v_1, v_2, \ldots, v_m]$, we aim to find an approximate solution $\tilde{x} \in K$. The approximation $\tilde{x}$ is obtained by projecting onto K such that $\tilde{x} = Vy$, where $y \in \mathbb{R}^m$, and ensuring the Galerkin condition $V^T AVy = V^T b$. This leads to solving a smaller system to find $\tilde{x} = Vy$.[6]

For eigenvalue problems ($Ax = \lambda x$), a similar approach finds an approximate eigenvalue $\tilde{\lambda} \in \mathbb{C}$ and an eigenvector $\tilde{u} = Vy$ by satisfying $V^H(A - \tilde{\lambda}I)\tilde{u} = 0$, resulting in a reduced dimension eigenvalue problem $(V^H AV - \tilde{\lambda}I)y = 0$.[6]

The Galerkin method can also employ a different subspace L for an oblique projection, leading to a non-orthogonal projection method. In essence, Krylov subspace methods apply the Galerkin approach within Krylov subspaces K_m, initiated by a vector v, to efficiently approximate solutions to linear and eigenvalue problems.[6]

Arnoldi's adaptation of Lanczos's work lays the foundational bridge to the symmetric Lanczos algorithm. In both the Arnoldi method and Lanczos's approach, Krylov subspaces serve as the crucial subspace K for the Galerkin projection. These methods effectively generate an orthonormal basis for K_m through a Gram-Schmidt-like process. A pivotal insight emerges when dealing with Hermitian matrices (A): the matrix $V^H AV$ is also Hermitian, rendering the Hessenberg matrix H_m, derived via Arnoldi's procedure, tridiagonal. This observation streamlines the Arnoldi process into the symmetric Lanczos algorithm, showcasing a transition and simplification pivotal for algorithmic efficiency.[6]

In 1950, the Lanczos algorithm was introduced, building on the foundational concepts of the time.

Lanczos approached solving $Ax = b$, with $A \in \mathbb{R}^{n\times n}$, $b \in \mathbb{R}^n$, and x as the solution vector in $\mathbb{R}^n$, by initiating the formation of a Krylov subspace $K_m(A, v) = \text{span}\{v, Av, A^2v, \ldots, A^{m-1}v\}$, where v is an initial non-zero vector and $m \leq n$.[6]

He proposed forming an orthonormal basis $V = [v_1, v_2, \ldots, v_m]$ for K_m via a process akin to Gram-Schmidt, aiding the projection onto K_m.[6]

Lanczos then aimed to find an approximate solution $\tilde{x} = Vy$, where $y \in \mathbb{R}^m$, by minimizing the residual $r = b - A\tilde{x}$ within the subspace defined by V. This approximation $\tilde{x}$ was to meet the Galerkin condition, thereby suggesting $V^T AV y = V^T b$, which leads to a manageable $m \times m$ system.[6]

Significantly, he observed that for a Hermitian matrix A, $V^H AV$ would yield a tridiagonal matrix, thus simplifying the process. Accordingly, the Hessenberg matrix H_m becomes tridiagonal when A is Hermitian.[6]

The Lanczos method is particularly effective for symmetric (or Hermitian) matrices, such as the covariance matrix $\Sigma = \frac{1}{n-1}XX^T$ in PCA, because it generates an orthonormal basis $V = [v_1, v_2, \ldots, v_m]$ for the Krylov subspace $\mathcal{K}_m(A, v_1)$, where the projection $H_m = V^T AV$ is a tridiagonal matrix:

$$H_m = \begin{bmatrix} \alpha_1 & \beta_1 & & \\ \beta_1 & \alpha_2 & \ddots & \\ & \ddots & \ddots & \beta_{m-1} \\ & & \beta_{m-1} & \alpha_m \end{bmatrix}.$$

This tridiagonal structure enables efficient eigenvalue computation, making Lanczos ideal for PCA. For non-symmetric matrices, the Arnoldi method is used, which generates a Hessenberg matrix, requiring more computational effort.

Lanczos's method employs iterative refinement of $\tilde{x}$ through a one-dimensional projection technique, updating $\tilde{x} := x + \alpha d$, where d is the direction of search and α is determined such that $b - A\tilde{x}$ is orthogonal to a constraint direction e, fulfilling $(r, e)/(Ad, e) = \alpha$.[6]

This iterative mechanism applies a polynomial of A to the initial residual, $r_{k+1} = p_{k+1}(A)r_0$, where p_{k+1} denotes the residual polynomial. This process encapsulates the core of Krylov subspace optimization.[6]

By leveraging Krylov subspaces, projection techniques, and polynomial iterations, the Lanczos method offers an efficient solution to linear and eigenvalue problems, marking a crucial development in the computational methods of numerical linear algebra.

3.2 Krylov Subspace Methods and Its Application in PCA

The utilization of Krylov subspace methods within Principal Component Analysis (PCA) is primarily driven by the need for efficient computation of the principal components, especially when dealing with large-scale datasets. Traditional eigendecomposition methods become computationally intensive as the size of the data matrix increases. Krylov subspace methods offer a scalable alternative by focusing on the computation of a few dominant eigenvalues and their corresponding eigenvectors.

A Krylov subspace generated by a matrix $\mathbf{A}$ and a nonzero vector $\mathbf{u}$ is defined as

$$\mathcal{K}_m(\mathbf{A}, \mathbf{u}) = \operatorname{span}\{\mathbf{u}, \mathbf{A}\mathbf{u}, \mathbf{A}^2\mathbf{u}, \ldots, \mathbf{A}^{m-1}\mathbf{u}\}.$$

Thus, every vector $\mathbf{x}$ in $\mathcal{K}_m(\mathbf{A}, \mathbf{u})$ can be written as a linear combination

$$\mathbf{x} = \alpha_1\mathbf{u} + \alpha_2\mathbf{A}\mathbf{u} + \cdots + \alpha_m\mathbf{A}^{m-1}\mathbf{u} = \mathbf{K}_m\mathbf{z},$$

where $\mathbf{z} \in \mathbb{R}^m$ contains the coefficients and $\mathbf{K}_m$ is the matrix whose columns are $\mathbf{u}, \mathbf{A}\mathbf{u}, \ldots, \mathbf{A}^{m-1}\mathbf{u}$. Note that by definition, if $\mathbf{x} \in \mathcal{K}_m(\mathbf{A}, \mathbf{u})$, then trivially $\mathbf{x} \in \mathcal{K}_{m+1}(\mathbf{A}, \mathbf{u})$; furthermore, multiplication by $\mathbf{A}$ yields

$$\mathbf{A}\mathbf{x} = \alpha_1\mathbf{A}\mathbf{u} + \alpha_2\mathbf{A}^2\mathbf{u} + \cdots + \alpha_m\mathbf{A}^m\mathbf{u} \in \mathcal{K}_{m+1}(\mathbf{A}, \mathbf{u}).$$

Theorem 4 (Properties of the Krylov Subspace)**.** *Let $\mathbf{A}$ be an $n \times n$ matrix, and let $\mathbf{u} \in \mathbb{R}^n$ be a nonzero vector. Let m be an integer such that $0 < m < n$. Then the Krylov subspace of order m generated by $\mathbf{A}$ and $\mathbf{u}$ is defined as*

$$\mathcal{K}_m(\mathbf{A}, \mathbf{u}) = \operatorname{span}\{\mathbf{u}, \mathbf{A}\mathbf{u}, \mathbf{A}^2\mathbf{u}, \ldots, \mathbf{A}^{m-1}\mathbf{u}\}.$$

Then, any vector $\mathbf{x}$ in $\mathcal{K}_m(\mathbf{A}, \mathbf{u})$ satisfies the following properties: first, $\mathbf{x}$ can be expressed in the form $\mathbf{x} = \mathbf{K}_m\mathbf{z}$ for some $\mathbf{z} \in \mathbb{R}^m$; second, $\mathbf{x}$ automatically belongs to $\mathcal{K}_{m+1}(\mathbf{A}, \mathbf{u})$; and third$\mathbf{A}\mathbf{x}$ lies in $\mathcal{K}_{m+1}(\mathbf{A}, \mathbf{u})$.

Proof. Assume that $\mathbf{x} \in \mathcal{K}_m(\mathbf{A}, \mathbf{u})$. Then there exist coefficients $\alpha_1, \alpha_2, \ldots, \alpha_m$ such that

$$\mathbf{x} = \alpha_1 \mathbf{u} + \alpha_2 \mathbf{A}\mathbf{u} + \cdots + \alpha_m \mathbf{A}^{m-1}\mathbf{u}.$$

Let $\mathbf{z} = \begin{bmatrix} \alpha_1 & \alpha_2 & \cdots & \alpha_m \end{bmatrix}^T$, so that by definition $\mathbf{x} = \mathbf{K}_m \mathbf{z}$. Moreover, by appending a term $0 \cdot \mathbf{A}^m \mathbf{u}$, it follows that $\mathbf{x} \in \mathcal{K}_{m+1}(\mathbf{A}, \mathbf{u})$. Finally, multiplying by $\mathbf{A}$ gives

$$\mathbf{A}\mathbf{x} = \alpha_1 \mathbf{A}\mathbf{u} + \alpha_2 \mathbf{A}^2\mathbf{u} + \cdots + \alpha_m \mathbf{A}^m\mathbf{u},$$

which by definition lies in $\mathcal{K}_{m+1}(\mathbf{A}, \mathbf{u})$. This completes the proof. □

In many applications, Krylov subspace methods are employed to solve the linear system $\mathbf{A}\mathbf{x} = \mathbf{b}$ or to compute dominant eigenvalues and eigenvectors. For instance, when solving $\mathbf{A}\mathbf{x} = \mathbf{b}$, one starts with an initial guess $\mathbf{x}_0$ and computes the residual $\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$.

Note that the Krylov subspace is typically generated from the initial residual $\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$, not from the initial guess $\mathbf{x}_0$ itself. Therefore, the Krylov-based iterate $\mathbf{x}_m$ is expressed as

$$\mathbf{x}_m = \mathbf{x}_0 + \mathbf{v}, \quad \text{where } \mathbf{v} \in \mathcal{K}_m(\mathbf{A}, \mathbf{r}_0).$$

It is not required that $\mathbf{x}_0 \in \mathcal{K}_m(\mathbf{A}, \mathbf{r}_0)$; rather, the method constructs a correction to $\mathbf{x}_0$ within the Krylov space. A correction vector $\mathbf{z}$ is then sought from the Krylov subspace generated by $\mathbf{r}_0$, that is,

$\mathcal{K}_m(\mathbf{A}, \mathbf{r}_0) = \operatorname{span}\{\mathbf{r}_0, \mathbf{A}\mathbf{r}_0, \ldots, \mathbf{A}^{m-1}\mathbf{r}_0\}$, such that the improved approximation $\mathbf{x} = \mathbf{x}_0 + \mathbf{z}$ minimizes the norm $\|\mathbf{b} - \mathbf{A}(\mathbf{x}_0 + \mathbf{z})\|$. Similarly, for eigenvalue problems $\mathbf{A}\mathbf{v} = \lambda\mathbf{v}$, one generates the Krylov subspace from an initial vector $\mathbf{v}_0$ and then constructs an orthonormal basis (using methods such as Arnoldi or Lanczos) to project $\mathbf{A}$ onto a smaller subspace, where the eigenvalue problem is easier to solve.

In the context of PCA, the covariance matrix Σ of a zero-mean dataset X is given by

$$\Sigma = \frac{1}{n-1} X X^T.$$

The eigenvectors of Σ represent the principal components, while the corresponding eigenvalues quantify the variance along these directions. For large-scale problems, computing a full eigendecomposition of Σ is often computationally prohibitive. Krylov subspace methods offer an efficient alternative by approximating the dominant eigenvectors through iterative matrix–vector products [7]. These methods begin with a randomly chosen initial vector $\mathbf{v} \in \mathbb{R}^n$, and construct the Krylov subspace of order k as

$$\mathcal{K}_k(\Sigma, \mathbf{v}) = \operatorname{span}\{\mathbf{v}, \Sigma\mathbf{v}, \Sigma^2\mathbf{v}, \ldots, \Sigma^{k-1}\mathbf{v}\}.$$

This subspace captures increasing amounts of spectral information as k grows and is especially effective for approximating the leading eigenvectors when the spectrum of Σ is dominated by a few large eigenvalues. An orthogonalization procedure, such as the Arnoldi or Lanczos algorithm (for symmetric matrices), is used to form an orthonormal basis for this subspace. The covariance matrix Σ is then projected onto this subspace, yielding a smaller matrix whose eigendecomposition provides approximate principal components and their corresponding variances. This process is especially attractive for large, sparse matrices where full eigendecomposition is computationally prohibitive.

It is worth noting that numerical challenges can arise in Krylov subspace methods when the generated vectors become nearly linearly dependent. Such near-dependencies, due to loss of orthogonality or poor conditioning of the matrix, can hinder the accurate computation of eigenvalues and eigenvectors. These issues are discussed in detail in [6].

Theorem 5 (Error Bound under Finite Precision in Krylov Subspace Methods)**.** *Let A be an $n \times n$ symmetric matrix with condition number $\kappa(A) = |\lambda_{max}|/|\lambda_{min}|$, and suppose an orthonormal basis $Q = [q_1, q_2, \ldots, q_m]$ for the Krylov subspace $\mathcal{K}_m(A, u)$ is computed via the Lanczos algorithm in finite precision arithmetic, so that:*

$$\|I_m - Q^T Q\|_2 \approx \epsilon,$$

where ϵ denotes machine precision and $\|\cdot\|_2$ is the spectral norm. If $\tilde{\lambda}$ is an eigenvalue of the computed tridiagonal matrix $T = Q^T A Q$ and λ is the corresponding true eigenvalue of A, then there exists a constant C, depending on eigenvalue separation and algorithm specifics, such that:

$$|\tilde{\lambda} - \lambda| \leq C\epsilon\kappa(A).$$

The spectral norm $\|I_m - Q^T Q\|_2$ measures the loss of orthogonality in Q, which is critical for the accuracy of eigenvalue approximations in PCA. Reorthogonalization techniques, such as Gram-Schmidt, are often used to ensure $\|I_m - Q^T Q\|_2$ remains small, improving numerical stability.

Proof. In exact arithmetic, the Lanczos algorithm constructs an orthonormal basis Q for $\mathcal{K}_m(A, u)$ satisfying $Q^T Q = I_m$, and the tridiagonal matrix $T = Q^T A Q$ has eigenvalues (Ritz values) approximating those of A. In finite precision, rounding errors result in $\|I_m - Q^T Q\| \approx \epsilon$, yet T is still computed as tridiagonal.

Since $Q^T Q = I_m + G$ with $\|G\| \approx \epsilon$, write $Q = US$ where $S = (Q^T Q)^{1/2} \approx I_m$ and $U = QS^{-1}$ is orthonormal. Thus, $T = Q^T A Q = SU^T AUS$. Approximating $S = I_m + K$ with $\|K\| \approx \epsilon$, we get:

$$T = (I_m + K)U^T AU(I_m + K) = U^T AU + E,$$

where $E = KU^T AU + U^T AUK + KU^T AUK$, and:

$$\|E\| \leq 2\epsilon\|A\| + \epsilon^2\|A\| \approx 2\epsilon\|A\|.$$

For symmetric matrices, if $\tilde{\lambda}$ is an eigenvalue of T and μ an eigenvalue of $U^T AU$, perturbation theory gives:

$$|\tilde{\lambda} - \mu| \leq \|E\| \approx 2\epsilon\|A\|.$$

Since U is orthonormal and spans $\mathcal{K}_m(A, u)$, μ approximates some eigenvalue λ of A, with error $|\lambda-\mu|$ typically small in exact arithmetic. In finite precision with $\|I_m - Q^T Q\| \approx \epsilon$, subspace accuracy is maintained, suggesting $|\lambda - \mu| \leq c\epsilon\|A\|$. Thus:

$$|\lambda - \tilde{\lambda}| \leq |\lambda - \mu| + |\mu - \tilde{\lambda}| \leq c\epsilon\|A\| + 2\epsilon\|A\|.$$

The relative error is:

$$\frac{|\lambda - \tilde{\lambda}|}{|\lambda|} \leq \frac{(c+2)\epsilon\|A\|}{|\lambda|}.$$

Since $\|A\| \approx |\lambda_{\max}|$ and $|\lambda| \geq |\lambda_{\min}|$, we have $\|A\|/|\lambda| \leq \kappa(A)$, so:

$$\frac{|\lambda - \tilde{\lambda}|}{|\lambda|} \leq (c+2)\epsilon\kappa(A).$$

Defining $C = c + 2$, where c depends on eigenvalue separation and subspace quality, the bound holds. This completes the proof. □

Overall, the use of Krylov subspace methods in PCA provides an efficient and robust framework for approximating the principal components while significantly reducing computational costs.

3.3 Relationship between Lanczos Algorithm and The Conjugate Gradient Method

In order to elucidate the equivalence between the Lanczos algorithm and the Conjugate Gradient (CG) method for solving $Ax = b$ with $A \in \mathbb{R}^{n\times n}$ symmetric positive definite, we decompose the argument into two key lemmas followed by a corollary.

Lemma 1 (Construction of the Krylov Subspace via Lanczos)**.** *Let $A \in \mathbb{R}^{n\times n}$ be a symmetric positive definite matrix and let $r_0 = b - Ax_0$ be the initial residual for some initial guess x_0. Define the Krylov subspace of order k as*

$$\mathcal{K}_k(A, r_0) = \operatorname{span}\{r_0, Ar_0, A^2 r_0, \dots, A^{k-1} r_0\}.$$

If we set $q_1 = r_0/\|r_0\|$ and apply the Lanczos algorithm, then for $j = 1, \dots, k$, the process generates an orthonormal basis

$$Q_k = [q_1, q_2, \dots, q_k]$$

for $\mathcal{K}_k(A, r_0)$ and a tridiagonal matrix $T_k \in \mathbb{R}^{k\times k}$ such that

$$AQ_k = Q_k T_k + \beta_k q_{k+1} e_k^T,$$

where e_k is the k-th canonical unit vector, and the recurrence is given by

$$\beta_{j+1} q_{j+1} = Aq_j - \alpha_j q_j - \beta_j q_{j-1}, \quad j = 1, \dots, k,$$

with $\alpha_j = q_j^T A q_j$, $\beta_1 = 0$, and β_j chosen to normalize q_{j+1}.

Proof. The Lanczos algorithm begins with $q_1 = r_0/\|r_0\|$ and iteratively constructs an orthonormal basis for $\mathcal{K}_k(A, r_0)$ using a three-term recurrence. For each j, it computes $v = Aq_j$ and orthogonalizes v against the previous vectors. Due to the symmetry of A, it suffices to orthogonalize against only q_{j-1} and q_j:

- Compute $\alpha_j = q_j^T A q_j$, the projection of Aq_j onto q_j.
- Subtract this projection: $v' = Aq_j - \alpha_j q_j$.
- If $j > 1$, subtract the component along q_{j-1}: $v'' = v' - \beta_j q_{j-1}$, where $\beta_j = q_{j-1}^T A q_j$ (and $\beta_1 = 0$ for $j = 1$).
- Normalize the result: $q_{j+1} = v''/\beta_{j+1}$, where $\beta_{j+1} = \|v''\|$.

This recurrence, $\beta_{j+1} q_{j+1} = Aq_j - \alpha_j q_j - \beta_j q_{j-1}$, ensures that q_{j+1} is orthogonal to all previous vectors $q_1, \dots, q_j$, since A's symmetry implies $q_i^T A q_j = 0$ for $|i - j| > 1$. Thus, Q_k is orthonormal, and $\mathcal{K}_k(A, r_0) = \operatorname{span}\{q_1, \dots, q_k\}$.

Writing the recurrence for all vectors in matrix form, we have:

$$AQ_k = Q_k T_k + \beta_k q_{k+1} e_k^T,$$

where T_k is tridiagonal with diagonal entries $\alpha_1, \dots, \alpha_k$ and off-diagonal entries $\beta_1, \dots, \beta_k$. The term $\beta_k q_{k+1} e_k^T$ accounts for Aq_k's component outside $\mathcal{K}_k(A, r_0)$. This construction is standard; see [8] or [9] for details. □

Lemma 2 (Representation of CG Iterates in the Krylov Subspace)**.** *For the same system $Ax = b$ and initial residual r_0, the Conjugate Gradient method generates an approximate solution x_k in the affine space*

$$x_0 + \mathcal{K}_k(A, r_0).$$

Specifically, there exists $y_k \in \mathbb{R}^k$ such that

$$x_k = x_0 + Q_k y_k.$$

Projecting the system onto the Krylov subspace via Q_k^T, we obtain

$$Q_k^T A Q_k y_k = Q_k^T r_0.$$

Since $Q_k^T A Q_k = T_k$ and $Q_k^T r_0 = \|r_0\| e_1$ (with e_1 the first canonical unit vector), this becomes

$$T_k y_k = \|r_0\| e_1.$$

Additionally, in CG, the search directions p_j are A-conjugate, satisfying $p_i^T A p_j = 0$ for $i \neq j$, and are updated via

$$p_{j+1} = r_{j+1} + \beta_j p_j,$$

starting with $p_0 = r_0$. The residuals satisfy $r_i^T p_j = 0$ for $i < j$, ensuring the Krylov subspace aligns with that from Lanczos.

Proof. The CG method minimizes the A-norm of the error, $\|x_k - x^*\|_A$, over $x_0 + \mathcal{K}_k(A, r_0)$, where $x^* = A^{-1} b$. Thus, $x_k = x_0 + Q_k y_k$, with Q_k an orthonormal basis for $\mathcal{K}_k(A, r_0)$. This minimization implies the residual $r_k = b - A x_k$ is orthogonal to $\mathcal{K}_k(A, r_0)$:

$$Q_k^T r_k = 0.$$

Since $r_k = r_0 - A(x_k - x_0) = r_0 - A Q_k y_k$, we compute:

$$Q_k^T r_k = Q_k^T r_0 - Q_k^T A Q_k y_k = 0.$$

Given $r_0 = \|r_0\| q_1$ and $Q_k^T q_1 = e_1$, we have $Q_k^T r_0 = \|r_0\| e_1$. Since $Q_k^T A Q_k = T_k$ (from Lemma 1), this becomes:

$$T_k y_k = \|r_0\| e_1.$$

In CG, search directions p_j are constructed to be A-conjugate, satisfying $p_i^T A p_j = 0$ for $i \neq j$. Starting with $p_0 = r_0$, the update $p_{j+1} = r_{j+1} + \beta_j p_j$ (with β_j chosen to enforce conjugacy) ensures $p_0, \ldots, p_{k-1}$ span $\mathcal{K}_k(A, r_0)$. The residuals $r_j = b - A x_j$ are orthogonal, $r_i^T r_j = 0$ for $i \neq j$, and $r_i^T p_j = 0$ for $i < j$, aligning with the orthogonality of Lanczos vectors q_j (up to scaling). □

Corollary 1 (Equivalence of Lanczos and CG for Positive Definite A)**.** *For a symmetric positive definite matrix A, the Conjugate Gradient (CG) method is equivalent to solving the reduced tridiagonal system*

$$T_k y_k = \|r_0\| e_1,$$

obtained from the Lanczos process. The normalized CG search directions are equivalent, up to scaling, to the Lanczos vectors. Moreover, CG's convergence is bounded by:

$$\|x_k - x^*\|_A \leq 2 \left(\frac{\sqrt{\kappa(A)} - 1}{\sqrt{\kappa(A)} + 1} \right)^k \|x_0 - x^*\|_A,$$

where $\kappa(A) = \lambda_{\max}(A)/\lambda_{\min}(A)$ is the condition number of A.

Proof. From Lemma 1, the Lanczos process constructs an orthonormal basis Q_k and a tridiagonal matrix T_k such that:

$$AQ_k = Q_k T_k + \beta_k q_{k+1} e_k^T.$$

From Lemma 2, the CG method produces the iterate $x_k = x_0 + Q_k y_k$, where y_k satisfies the reduced system:

$$T_k y_k = \|r_0\| e_1.$$

Since both methods operate within the same Krylov subspace $\mathcal{K}_k(A, r_0)$ and solve the same reduced system, they are equivalent for symmetric positive definite A.

The CG search directions p_j are A-conjugate (i.e., $p_i^T A p_j = 0$ for $i \neq j$) and span $\mathcal{K}_k(A, r_0)$. These directions correspond to the Lanczos vectors q_j, up to scaling, because both sets form bases for the same subspace, with q_j being orthonormal and p_j satisfying A-orthogonality. The convergence bound:

$$\|x_k - x^*\|_A \leq 2 \left(\frac{\sqrt{\kappa(A)} - 1}{\sqrt{\kappa(A)} + 1} \right)^k \|x_0 - x^*\|_A,$$

arises from CG's minimization of the A-norm of the error over $\mathcal{K}_k(A, r_0)$, leveraging Chebyshev polynomials, and depends on the condition number $\kappa(A)$; see Greenbaum (1997) for details.

Remark. If A is symmetric positive semi-definite rather than positive definite (i.e., $x^T Ax \geq 0$, with possible zero eigenvalues), the equivalence and convergence guarantees no longer hold universally. Specifically: - The Lanczos process remains valid, producing an orthonormal basis Q_k and a tridiagonal T_k. However, T_k may have zero eigenvalues corresponding to those of A, reflecting

A's singularity. - In CG, the step size involves terms like $p_j^T A p_j$, which must be positive for the algorithm to proceed. If A has zero eigenvalues, $p_j^T A p_j = 0$ can occur for nonzero p_j, causing the algorithm to break down due to undefined steps. - The convergence bound becomes meaningless as $\lambda_{\min}(A) = 0$, making $\kappa(A) \to \infty$, which eliminates the guaranteed geometric error reduction.

Thus, while the Lanczos process can still function, the direct equivalence with CG is lost, and CG may fail to converge or terminate prematurely. For semidefinite systems, alternative methods like MINRES, which handle singularity, are more appropriate. □

3.4 The Best Low-Rank Matrix Approximation (Eckart–Young–Mirsky Theorem)

Theorem 6 (Eckart–Young–Mirsky Theorem)**.** *Let $A \in \mathbb{R}^{m\times n}$ have the singular value decomposition*

$$A = U\Sigma V^T, \quad \textit{with} \quad \Sigma = \mathrm{diag}(\sigma_1, \sigma_2, \ldots, \sigma_r, 0, \ldots, 0),$$

where $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r > 0$ and $r = \mathrm{rank}(A)$. For any $k < r$, define the truncated SVD of A by

$$A_k = U\Sigma_k V^T, \quad \textit{with} \quad \Sigma_k = \mathrm{diag}(\sigma_1, \ldots, \sigma_k, 0, \ldots, 0).$$

Then A_k is the best rank-k approximation to A in both the Frobenius norm and the spectral norm, meaning that

$$\|A - A_k\|_F = \sqrt{\sum_{i=k+1}^{r} \sigma_i^2} \quad \textit{and} \quad \|A - A_k\|_2 = \sigma_{k+1},$$

and no other matrix B with $\mathrm{rank}(B) \leq k$ satisfies $\|A - B\| < \|A - A_k\|$ for these norms.

Proof. The goal is to prove that A_k, the truncated SVD of A, minimizes the approximation error $\|A - B\|$ over all matrices B of rank at most k under both the Frobenius norm ($\|\cdot\|_F$) and the spectral norm ($\|\cdot\|_2$).

Both norms are unitarily invariant, meaning that for any orthogonal matrices U and V, and any matrix X,

$$\|X\| = \|U^T X V\|.$$

This property, detailed in [10, 11], simplifies the problem significantly.

Consider the SVD of $A = U\Sigma V^T$, where $U \in \mathbb{R}^{m\times m}$ and $V \in \mathbb{R}^{n\times n}$ are orthogonal, and $\Sigma \in \mathbb{R}^{m\times n}$ is diagonal with entries $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r > 0$, and $\sigma_i = 0$ for $i > r$. Let B be any matrix with $\text{rank}(B) \leq k$. Since U and V have full column rank, we can express $B = UCV^T$, where $C \in \mathbb{R}^{m\times n}$ satisfies $\text{rank}(C) \leq k$.

The approximation error is:

$$\|A - B\| = \|U\Sigma V^T - UCV^T\| = \|U(\Sigma - C)V^T\|.$$

By unitary invariance, this equals $\|\Sigma - C\|$. Thus, we need to minimize $\|\Sigma - C\|$ over all C with $\text{rank}(C) \leq k$.

Since Σ is diagonal, we first justify why the optimal C can be taken as diagonal. For the Frobenius norm,

$$\|\Sigma - C\|_F^2 = \sum_{i=1}^{\min(m,n)} \sum_{j=1}^{\min(m,n)} (\Sigma_{ij} - C_{ij})^2.$$

As $\Sigma_{ij} = 0$ for $i \neq j$, any non-zero off-diagonal C_{ij} contributes positively to the sum without reducing the diagonal terms $(\sigma_i - C_{ii})^2$. Hence, setting $C_{ij} = 0$ for $i \neq j$ minimizes the norm, making C diagonal. Similarly, for the spectral norm (the largest singular value), off-diagonal elements in C can increase $\|\Sigma - C\|_2$, and the optimal low-rank approximation to a diagonal matrix is achieved with a diagonal C, as established in [11].

Thus, let $C = \text{diag}(c_1, c_2, \ldots, c_p)$, where $p = \min(m, n)$, and c_i are to be determined, with at most k non-zero entries due to the rank constraint.

Then, for the Frobenius norm,

$$\|\Sigma - C\|_F^2 = \sum_{i=1}^{p} (\sigma_i - c_i)^2,$$

where $\sigma_i = 0$ for $i > r$. To minimize this, each c_i should be as close as possible to σ_i, but since $\text{rank}(C) \leq k$, at most k of the c_i can be non-zero. Given $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r > 0$, the optimal choice is $c_i = \sigma_i$ for $i = 1, \ldots, k$, and $c_i = 0$ for $i > k$. Then:

- For $i = 1, \ldots, k$: $(\sigma_i - c_i)^2 = (0)^2 = 0$,
- For $i = k + 1, \ldots, r$: $(\sigma_i - 0)^2 = \sigma_i^2$,
- For $i > r$: $(0 - 0)^2 = 0$.

Thus:

$$\|\Sigma - \Sigma_k\|_F^2 = \sum_{i=k+1}^{r} \sigma_i^2,$$

so $\|A - A_k\|_F = \sqrt{\sum_{i=k+1}^{r} \sigma_i^2}$, matching the theorem and consistent with [12, 13].

The spectral norm $\|\Sigma - C\|_2$ is the largest singular value of $\Sigma - C$. By a standard result (e.g., Weyl's inequalities in [11]), for any C with $\operatorname{rank}(C) \leq k$,

$$\|\Sigma - C\|_2 \geq \sigma_{k+1}.$$

When $C = \Sigma_k$, $\Sigma - \Sigma_k = \operatorname{diag}(0, \ldots, 0, \sigma_{k+1}, \ldots, \sigma_r, 0, \ldots, 0)$, whose largest singular value is σ_{k+1}. Thus, $\|A - A_k\|_2 = \sigma_{k+1}$, achieving the lower bound.

For any B with $\operatorname{rank}(B) \leq k$, $\|A - B\|_F \geq \|A - A_k\|_F$ and $\|A - B\|_2 \geq \|A - A_k\|_2$, with equality only if B matches A_k's action on the top k singular directions (up to unitary transformations). Hence, A_k is the best rank-k approximation. □

4 The Ubaru algorithm

Building on the mathematical and computational foundations established in Sections 2 and 3, we now introduce the Ubaru Algorithm. This novel approach synergizes PCA with Krylov subspace methods, aiming to improve both computational efficiency and accuracy in high-dimensional settings.

The algorithm, from now labelled as the Ubaru algorithm[2], under investigation introduces a randomized approach to the PCA and Krylov subspace methodology, merging both techniques to obtain an efficient solution.

Randomization in this context refers to the use of random vectors in the initial stages of the method, offering a probabilistic guarantee of capturing dominant features of the data.What sets this algorithm apart is its unique stopping criterion. Instead of relying solely on traditional eigenvalue thresholds, it employs a model selection framework derived from random matrix perturbation theory. In essence, this stopping criterion assesses the significance of each principal component by considering its inherent randomness, thus providing a more nuanced and potentially robust decision on when to halt the dimensionality reduction process.[7]

This approach not only offers a potential speedup over traditional PCA methods but also provides a deeper understanding of the inherent structure of the data, thanks to the insights from random matrix theory and Krylov subspaces.[7]

In conclusion, by merging traditional PCA techniques with Krylov subspace methodologies and incorporating a randomized approach, the algorithm offers an innovative way to tackle the challenges of high-dimensional data analysis.

Its unique stopping criterion, grounded in random matrix perturbation theory, promises a fresh perspective on dimensionality reduction and its implications.

4.1 Theoretical Advancements and Mathematical Innovations behind the Ubaru Algorithm

This section aims to bridge the gap between the conceptual introduction of the Ubaru Algorithm and its detailed operational framework.

Leveraging the foundational principles of the Lanczos algorithm 4.1, particularly its efficient use of Krylov subspaces for projection, the Ubaru Algorithm advances the field of dimensionality reduction by incorporating these concepts with an innovative iterative refinement and selection process. This approach employs an Information Criterion (IC) to determine the optimal number of dimensions, ensuring an efficient balance between dimensionality reduction and the preservation of significant data characteristics.

The Ubaru Algorithm:[14]

```
1. Initialize $IC = \mathbf{zeros}(p, 1)$; $Q = [\ ]$; $k = 1$; $m = \log(p)/\sqrt{\epsilon}$.
2. For $k = 1$ to $p$ do
    (a) Generate a random vector $v_k$ with $\|v_k\|_2 = 1$.
    (b) $K = \frac{1}{n}[Xv_k; (XX^T)Xv_k; \ldots; (XX^T)^{m-1}Xv_k]$.
    (c) $Q = \mathrm{orth}([Q; K])$, $Q = Q(:, 1 : k)$.
    (d) $T = \frac{1}{n}Q^T XX^T Q$.
    (e) $[V, \Theta] = \mathrm{eig}(T)$.
    (f) $IC(k) = n\left(\|X\|_F^2 - \sum_{i=1}^{k} \Theta_i\right) - C_n \frac{(p-k)(p-k+1)}{2}$.
    (g) If $(k > 1 \,\mathrm{and}\, IC(k) > IC(k-1))$ then break.
3. End for
4. $q = k - 1$. Output $q$ and $Y = QV$.
```

This approximation hinges on several critical parameters: the noise variance denoted by σ, a regularization or penalty parameter C_n, and an error tolerance ϵ. The goal is to distill the essence of the data into a lower-dimensional space, encapsulated by a dimension q and the corresponding principal subspace approximation Y_q.

The journey begins with the initialization of several key components. An Information Criterion vector, IC, is set to zero for each feature dimension, providing a basis for evaluating the efficacy of the dimensionality reduction at each step. An empty matrix Q is prepared to store orthogonal vectors that will span the principal subspace. The iterative process is kick-started with a counter $k = 1$, and a parameter $m = \log(p)/\sqrt{\epsilon}$ is calculated to govern the number of power iterations, a reflection of the balance between accuracy and computational efficiency. Furthermore, a formula Φ combines norms of the data matrix, noise variance, and a parameter ρ, setting the stage for the iterative optimization process.

The Ubaru algorithm unfolds in a loop, iterating over each dimension from 1 to p. Within each iteration, a random unit vector V_k is generated to initiate the exploration of the data's structure. This vector undergoes a series of matrix-vector products involving X and its transpose, enhancing its alignment with the principal directions of the data. The algorithm then orthogonalizes a temporary matrix K against the current basis Q, refining the subspace spanned by Q. Following this, the projection T of the data onto the orthogonal vectors in Q is computed, from which the eigenvectors V and eigenvalues Θ are extracted. These eigenvectors and eigenvalues encapsulate the principal components of the data in the current iteration. The Information Criterion IC is updated to reflect the contribution of the current dimension, incorporating the eigenvectors, eigenvalues, and noise variance into its calculation. A pivotal decision point arrives when the Information Criterion suggests that adding more dimensions no longer enhances the model, marked by $\text{IC}(k) > \text{IC}(k-1)$ for $k > 1$, prompting a break from the loop.

Upon completion of this iterative exploration, the algorithm settles on an optimal dimensionality $q = k - 1$, signifying the identification of a subspace that balances dimensionality reduction with the retention of significant data characteristics. The output of this process is not just the dimension q but also the approximation to the principal subspace $Y = QV$, marking the culmination of a sophisticated yet efficient approach to uncovering the underlying structure of high-dimensional data. Through this meticulous algorithmic journey, informed by the insights of Ubaru et al., we navigate the complexities of dimensionality reduction, achieving a nuanced understanding of the data's principal components.

After outlining the operational steps of the Ubaru Algorithm, we now focus on a critical component of its design—the stopping criterion. This next section details how the Information Criterion is integrated into the algorithm to balance dimensionality reduction with information retention.

4.2 Stopping Criteria of the Ubaru Algorithm

Continuing with the exploration of dimensionality reduction and principal component analysis, a critical component in determining the optimal number of dimensions to retain is the Information Criterion (IC). IC is designed for a specific purpose: to balance model complexity and accuracy in determining the optimal number of dimensions. While other criteria like the Akaike Information Criterion (AIC) and Bayesian Information Criterion (BIC) also aim to balance fit and complexity, they do so through different mathematical formulations and penalization approaches.

The IC is mathematically defined as:

$$\mathrm{IC}(k) = n\left(\Phi - \sum_{i=1}^{k} (\theta_i - \sigma)^2\right) - C_n \frac{(p-k)(p-k-1)}{2}, \tag{3}$$

where Φ is a composite measure that incorporates the data matrix X, its noise variance σ, and the dimensionality p of the feature space, given by:

$$\Phi = \frac{1}{n^2\|X\|_F^4} - \frac{2\sigma}{\|X\|_F^2} + p\sigma^2. \tag{4}$$

encapsulating the balance between the magnitude of the data, as measured by the Frobenius norm $|X|_F$, and the inherent noise variance. Let us prove the strong strong consistency of k.

Theorem 7 (Strong Consistency of $\hat{k}$)**.** *Let $\hat{k}$ be the estimator defined as*

$$\hat{k} = \arg\min_k \mathrm{IC}(k),$$

where $\mathrm{IC}(k)$ *denotes the information criterion evaluated at model dimension k. Then, under the conditions specified in [2], $\hat{k}$ converges almost surely to the true parameter k_0 as the sample size n tends to infinity.*

Proof. The proof proceeds by contradiction, by considering two cases.

Case 1: $\hat{k} > k_0$.
Assume $\hat{k} > k_0$. Consider the difference between the information criteria at $\hat{k}$ and k_0:

$$\begin{aligned}\mathrm{IC}(\hat{k}) - \mathrm{IC}(k_0) &= \frac{n}{2\sigma_0^2}\left(\sum_{i=\hat{k}+1}^{p} (\lambda_i - \sigma)^2 - \sum_{i=k_0+1}^{p} (\lambda_i - \sigma)^2\right) \\ &\quad - C_n\left(\frac{(p-\hat{k})(p-\hat{k}-1)}{2} - \frac{(p-k_0)(p-k_0-1)}{2}\right) \\ &= -\frac{n}{2\sigma_0^2}\sum_{i=k_0+1}^{\hat{k}} (\lambda_i - \sigma)^2 - C_n\left(\frac{(\hat{k}-k_0)(\hat{k}+k_0-2p+1)}{2}\right).\end{aligned}$$

Normalizing by n and applying the law of the iterated logarithm to the λ_i, we deduce that as $n \to \infty$ the difference $\mathrm{IC}(\hat{k}) - \mathrm{IC}(k_0)$ becomes negative. This contradicts the assumption that $\hat{k}$ minimizes the information criterion. Hence, $\hat{k} > k_0$ cannot occur for large n almost surely.

Case 2: $\hat{k} < k_1$.

Now, assume $\hat{k} < k_1$, for some k_1 with $k_0 < k_1$. The difference between the information criteria at $\hat{k}$ and k_1 is given by

$$
\begin{aligned}
\mathrm{IC}(\hat{k}) - \mathrm{IC}(k_1) = {} & \frac{n}{2\sigma_0^2}\left(\sum_{i=\hat{k}+1}^{p} (\lambda_i - \sigma)^2 - \sum_{i=k_1+1}^{p} (\lambda_i - \sigma)^2\right) \\
& - C_n \left(\frac{(p-\hat{k})(p-\hat{k}-1)}{2} - \frac{(p-k_1)(p-k_1-1)}{2}\right) \\
= {} & \frac{n}{2\sigma_0^2}(k_1 - \hat{k})\, O\left(\sqrt{\frac{\log\log n}{n}}\,\sigma\right) - C_n \left(\frac{(\hat{k}-k_1)(\hat{k}+k_1-2p+1)}{2}\right).
\end{aligned}
$$

Normalizing by C_n and letting n increase, the above difference also becomes negative, contradicting the assumption that $\hat{k}$ minimizes the information criterion. Therefore, $\hat{k} < k_1$ cannot hold for large n almost surely.

Since both $\hat{k} > k_0$ and $\hat{k} < k_1$ lead to contradictions, it follows that $\hat{k}$ must converge almost surely to k_0 as n tends to infinity.

This establishes the strong consistency of the estimator $\hat{k}$, ensuring that the probability of $\hat{k}$ diverging from k_0 is zero in the limit. □

For an example, let's consider a sports analytics dataset where X represents various performance metrics of athletes, such as speed, endurance, and strength, across different sports, with p being the number of these metrics and n the number of athletes. The noise variance σ captures variability in performance due to factors like equipment quality, weather conditions, and day-to-day athlete condition.

Applying Φ to this dataset, the term $\frac{1}{n^2|X|_F^4}$ could undervalue the dataset when the performance metrics are inherently varied and high. The component $-\frac{2\sigma}{|X|_F^2}$ attempts to adjust for noise, but might not fully account for the complex ways in which external factors influence performance metrics. Lastly, the term $p\sigma^2$ suggests that adding more performance metrics linearly increases the complexity of the dataset, which may not accurately reflect the nuanced way in which different metrics interact to explain athlete performance.

Further dissecting the IC formula, the summation $\sum_{i=1}^{k} (\theta_i - \sigma)^2$ aggregates the squared deviations of the eigenvalues θ_i from the noise variance σ for the first k principal components. This summation effectively captures the total variance explained by these components, adjusted for the noise, thus quantifying the data representation's fidelity up to the k-th dimension.

The regularization term $-C_n \frac{(p-k)(p-k-1)}{2}$ introduces a nuanced penalty mechanism. Through the parameter C_n, it imposes a cost on the model's complexity, growing as the number of retained components decreases. This design inherently discourages the retention of superfluous components, aligning the model towards simplicity and robustness against overfitting.

The algorithm employs this IC as a critical decision-making tool. By evaluating the IC at each iteration and comparing it with the previous step's value, the algorithm identifies the moment when the inclusion of additional components ceases to yield proportional benefits in terms of explained variance, after accounting for the complexity penalty. This decision point marks the optimal balance between model simplicity and explanatory power, where each component's inclusion is justified by a tangible improvement in data representation.

In essence, the IC encapsulates a comprehensive evaluation of the trade-offs involved in principal component selection. It harmonizes the insights drawn from the data's inherent structure, the impact of noise, and the penalties associated with model complexity. Through this criterion, the algorithm navigates the intricate landscape of dimensionality reduction, stopping at the juncture where extending the model no longer aligns with the principles of parsimony and effectiveness in capturing the essence of the data. This methodology, rooted in a deep understanding of the data and its characteristics, ensures that the derived subspace is both representative and efficient, embodying the core objectives of principal component analysis.

4.3 Comments and Insights

Exploring the algorithm's design reveals its grounding in sophisticated statistical theories and its adaptation to practical scenarios. This exploration uncovers a series of insights and considerations that are pivotal for understanding and implementing the algorithm effectively.

The Ubaru algorithm builds on classical multivariate statistics, using the sample covariance matrix S_n as a consistent estimator for the true covariance matrix Σ. This approach, grounded in foundational results from Muirhead [4], highlights the importance of statistical consistency for reliable dimensionality reduction and principal component analysis.

Further, the algorithm employs eigendecomposition and perturbation techniques to dissect the covariance matrices. The reliability of the top q eigenvectors G_q, as assured under specific conditions, is a testament to the method's robustness, drawing on principles from Saad's research [15]. This step is crucial for isolating significant components of the data's variance structure, providing a clear pathway to dimensionality reduction.

Central to the algorithm's decision-making process is the Information Criterion (IC), which synthesizes the estimated noise variance, eigenvalues of the sample covariance matrix, and a regularization parameter C_n. This criterion, detailed by Ubaru et al. [2], guides the algorithm to halt when further improvements in IC do not justify the inclusion of additional principal components, thus identifying the optimal dimensionality for the data representation.

Employing Krylov subspace methods enhances the algorithm's efficiency in computing the partial spectrum of large matrices. This technique, pivotal for managing computational resources, is guided by the error tolerance ϵ and the number of power iterations m, reflecting Saad's contributions to numerical linear algebra [16].

In dimensionality reduction, choosing the right noise variance (σ) and regularization parameter (C_n) is crucial for making the algorithm work well, as highlighted by Ubaru in his thesis. These parameters are key to balancing how accurate and complex our model is. For example, in datasets with lots of zeros (sparse data), a smaller σ helps to keep important details from being lost, while the right C_n prevents the model from getting too complicated and focusing on irrelevant data. Practically, finding the best values for σ and C_n often involves using cross-validation, tapping into what experts know about the data, and refining choices based on trial and error. This approach ensures the model is both accurate and manageable.[14]

The algorithm's utility in finite samples, despite its design for asymptotic accuracy, is another vital aspect. Its performance in practical scenarios hinges on the conditions for correctly estimating principal components, as evidenced by foundational statistical theory [17]. This consideration ensures the algorithm remains relevant and applicable across various data scales and conditions.

Lastly, the computational cost, influenced by factors such as the sparsity of the data matrix, the dimension q, and the number of iterations m, underscores the algorithm's efficiency. Particularly for sparse data or when q is small relative to the number of features p, this aspect becomes crucial, ensuring the method's applicability in large-scale data analysis scenarios [18].

These insights and considerations, derived from theoretical derivations and practical applications, underscore the algorithm's complexity and sophistication. They highlight the importance of statistical consistency, computational efficiency, and careful parameter selection, ensuring the algorithm's robustness and reliability in extracting meaningful insights from high-dimensional data.

4.4 Implementation of the Ubaru Algorithm in Matlab

Delving into the nuanced realm of dimensionality reduction, the Krylov subspace-based dimension estimation algorithm emerges as a sophisticated tool designed to distill the essence of a data matrix X into an estimated dimension q and its corresponding principal subspace Y_q. Initiated with inputs such as the data matrix X, noise variance σ, and potentially augmented by a regularization parameter C_n and an error tolerance ϵ, this algorithm is poised to tackle the challenges posed by complex datasets [16].

The heart of this algorithm beats within the realms of a Krylov subspace method, a choice that imbues the process with the precision and efficiency required for large-scale eigenvalue problems. This method, when intertwined with an information criterion, provides a robust framework for navigating the intricacies of dimension estimation. The methodology's efficacy is further demonstrated through the generation of outputs that not only quantify the reduced dimension q but also articulate the structure of Y_q, offering a lens through which the data's fundamental characteristics can be observed and analyzed.

A hallmark of this algorithm's design is its implementation using the Lanczos method, a decision that underscores its adaptability to large datasets and its resilience in the face of data variability. This implementation detail, coupled with a conscientious incorporation of noise considerations, speaks volumes about the algorithm's robustness, ensuring its performance remains steadfast across diverse data landscapes [19].

4.4.1 The Ubaru Algorithm Depiction

Below, I provide a concise discussion of the Ubaru Algorithm. For details on the fully operationalized code, consult the appendix Details in Appendix.

Initially, a synthetic data matrix X is generated employing sparse matrix techniques, with added noise to closely emulate real-world data scenarios. Essential parameters such as the dimensions n and k, the noise level σ_{noise}, and the number of Lanczos steps m_{steps} are specified to set the foundation for the subsequent rank estimation process.

Rank Estimation via Krylov Subspace Method
The process begins by initializing the Information Criterion IC array alongside parameters that include the Frobenius norm of X ($\|X\|_F^2$), facilitating the approximation quality monitoring.

An empty set for the orthonormal basis V and an empty tridiagonal matrix T are prepared.

Throughout each iteration, from 1 to a predefined maximum rank, the algorithm performs a Lanczos update to enrich the basis V with new vectors derived from the data matrix X, simultaneously updating the matrix T to mirror the structure of the Krylov subspace.

The eigenvalues θ of the updated tridiagonal matrix T are computed, and the information criterion $IC(k)$ is updated accordingly, based on these eigenvalues and the norm of X.

The stopping criterion is evaluated based on the progression of the Information Criterion; if $IC(k)$ begins to increase, suggesting diminishing returns on further rank estimation, the iteration is terminated. The estimated rank is then determined as $q = k - 1$, where k is the iteration count at which the loop was exited.

Output and Visualization
Finally, the singular values of X are computed and plotted to visualize the results of the rank estimation. The estimated rank q is highlighted on the plot to offer a visual confirmation of the algorithm's effectiveness, providing an insightful representation of the underlying data structure as perceived through the lens of the Krylov subspace method.

4.4.2 Reproducing Results Using the Ubaru Algorithm

In this section, we present efforts to reproduce the results from the dissertation titled Algorithmic "Advances in Learning from Large Dimensional Matrices and Scientific Data" by Shashanka Ubaru. The dissertation focuses on the Krylov Subspace method, specifically the discussed algorithm, for numerical rank estimation in large matrices. [14]

Experiment Setup
The experiment aims to replicate the findings in Table 4.1 of the dissertation, detailing the performance of the Krylov Subspace method when applied to synthetic sparse random matrices. The setup involves varying parameters such as matrix size n, actual and estimated numerical rank q and $\tilde{q}$, signal strength λ_q, noise level σ, Frobenius norm error, and runtime of the algorithm. [14]

Methodology
We constructed synthetic sparse random matrices in the form $X = A\Lambda A^T + N$, where A is a sparse signal matrix, Λ a diagonal matrix, and N a Gaussian sparse random matrix. The number of Lanczos steps per iteration was fixed at $m = 10$. [14]

We compare the results to the reproduction efforts with those reported in the dissertation in the following table:

Table 1: Comparison of Dissertation and Reproduction Results

Parameter	n	Actual q	λ_q	Estimated $\tilde{q}$	$\|A - Y_q Y_q' X\|_F$	Runtime (secs)
Dissertation	500	50	5	50	1.1e4	0.72 secs
Reproduction	500	50	5	50	879.51	1.66 secs
Dissertation	4000	100	2	100	5.0e4	20.56 secs
Reproduction	4000	100	2	100	5057	66.7 secs

Comparison and Discussion of Results

The operationalized algorithm in MATLAB does not fully replicate the dissertation's results; however, it demonstrates noteworthy aspects:

The MATLAB-based implementation of the Krylov subspace dimension estimation algorithm, juxtaposed with findings from the original dissertation, unfolds a complex narrative. While it doesn't perfectly echo the dissertation's outcomes, it elucidates several critical aspects. Notably, the algorithm exhibits precision in rank estimation for synthetic sparse data matrices, aligning with the dissertation's findings and underscoring its adeptness at discerning dimensionality in sparse scenarios.

However, an unexpected divergence surfaces in the Frobenius norm error metrics; the reproduction reports significantly lower errors—879.51 versus the dissertation's 11,000 for $n = 500$ and $q = 50$, and 5057 against 50,000 for $n = 4000$ and $q = 100$. This discrepancy not only questions the initial hypothesis but also hints at possible variances in data generation, algorithmic implementation, or computational environments between the two studies.

This observation propels a reevaluation of the MATLAB implementation's efficiency, particularly in handling errors for sparse synthetic data, suggesting a potential edge in specific conditions or under certain optimizations. Additionally, despite a runtime uptick in larger-scale experiments, the consistent performance reaffirms the algorithm's scalability and operational reliability, positioning it as a formidable tool for analyzing extensive datasets.

In essence, the MATLAB replication, despite its variance in error metrics, reaffirms the algorithm's rank estimation fidelity and opens new dialogues on its error management efficacy. This divergence beckons further exploration into algorithmic fine-tuning and methodological adaptations, promising to deepen the collective grasp on the Krylov Subspace method's applications in sparse data analysis.

5 Review of Stopping Criteria

In this section, we examine various stopping criteria and introduce a novel criterion, which serves as a hybrid of existing ones. Consequently, we question whether this new criterion offers greater efficiency compared to the Information Criterion (IC) suggested by Ubaru.

5.1 Introduction

In the exploration [20] of dimensionality reduction techniques like Principal Component Analysis (PCA), the strategy for halting the inclusion of components, termed stopping criteria, plays a vital role. The scholarly landscape offers a spectrum of these criteria, each designed to pinpoint the optimal component count in a nuanced manner. For instance, the Eigenvalue Threshold method advocates for retaining components whose eigenvalues exceed a certain benchmark, typically one, underlining the belief that such components significantly contribute to the dataset's variance. Conversely, the Percentage of Total Variance approach opts for a cumulative variance coverage, such as 95% or 99%, as the determinant for component selection, ensuring a substantial proportion of the data's variability is encapsulated.

Furthermore, the Scree Plot introduces a visual tactic by plotting eigenvalues in descending order, seeking the 'elbow' where the slope flattens as a natural demarcation for component relevance. Parallel Analysis, on the other hand, benchmarks the dataset's eigenvalues against those from randomly generated datasets, a method that inherently incorporates randomness to guard against overestimation of component count. The Minimum Average Partial (MAP) Criterion delves into the correlations among variables, identifying the number of components to extract before the average squared partial correlation reaches its nadir, thereby acknowledging the data's underlying structure.

These criteria are not without their strengths and weaknesses, as outlined in the literature [20]. The simplicity and ease of implementation of the Eigenvalue Threshold are balanced by its potential arbitrariness and possible detachment from the data's intrinsic structure. The intuitive appeal of the Percentage of Total Variance and Scree Plot methods is occasionally marred by their subjective nature and reliance on arbitrary cutoffs. Parallel Analysis, while robust against overestimation, demands considerable computational resources, and its outcomes can be swayed by the choice of random datasets. The MAP Criterion, although thorough in considering data structure, might present complexities in understanding and application.

Selecting an apt stopping criterion hinges on a confluence of theoretical and practical considerations, including the data's nature, the analysis's objectives, computational constraints, and the need for interpretability. The nature of the data—whether it possesses a strong underlying structure—may favor methods like Parallel Analysis, while the primary goal of the analysis, whether for visualization or rigorous data reduction, could influence the choice towards more intuitive or quantitative criteria.

In sum, the selection among these diverse stopping criteria necessitates a tailored approach, harmonizing the analysis's specific demands with both the theoretical underpinnings and practical feasibilities. This balanced consideration ensures the chosen criterion not only aligns with the dataset's characteristics but also with the broader objectives of the dimensionality reduction endeavor.

In light of the limitations and challenges observed with the traditional stopping criteria, this section proposes novel alternatives that potentially address these issues. Each criterion's theoretical underpinning and mathematical justification will be explored to provide a comprehensive understanding of its potential benefits and application.

5.2 Information Theoretic Criterion

The Information Theoretic Criterion in PCA embodies a nuanced approach towards balancing the maximization of information content against the inherent loss entailed in dimensionality reduction. This methodology, rooted deeply in the principles of information theory, pivots on the entropy of the data distribution to quantify the uncertainty or randomness inherent in a dataset X. Entropy serves as a measure of this uncertainty, with its calculation in the PCA context reflecting the variance of data projected along principal components [21].

Given a dataset X, with its variance-covariance matrix having eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_p$, the goal of PCA is to select k principal components that maximize the retained information while minimizing the dimensionality of the transformed data. The information content can be quantified by entropy, H, which is calculated as:

$$H = -\sum_{i=1}^{k} \frac{\lambda_i}{\sum_{j=1}^{p} \lambda_j} \log \left(\frac{\lambda_i}{\sum_{j=1}^{p} \lambda_j} \right),$$

where λ_i are the eigenvalues corresponding to the variance captured by each principal component, and k is the number of principal components selected.

The optimization problem can then be stated as:

$$\max_k H = -\sum_{i=1}^{k} \frac{\lambda_i}{\sum_{j=1}^{p} \lambda_j} \log \left(\frac{\lambda_i}{\sum_{j=1}^{p} \lambda_j} \right)$$

subject to:

$$1 \leq k \leq p,$$

where p is the total number of principal components (or the dimensionality of X) before reduction, and k is the number of principal components chosen to represent the data after dimensionality reduction.[21]

This optimization problem seeks to find the value of k that provides the best balance between information retention (as quantified by entropy) and dimensionality reduction. It involves selecting the top k eigenvalues (and thus principal components) that capture the most variance of the data, which corresponds to maximizing the entropy H, reflecting the information content of the reduced-dimensionality data. This selection process aims to keep the transformed data as informative as possible while reducing its complexity.

5.3 Spectral Density Estimation Criterion

The spectral density estimation criterion excels in high-dimensional data analysis, offering insights into variance distribution across dimensions. It's ideal for dimensionality reduction, identifying informative features, and noise reduction in datasets where discerning signal from noise is crucial. This criterion is most effective when linear relationships dominate, aiding in optimizing data representation and enhancing machine learning models by focusing on the most significant dimensions. It's particularly useful in exploratory data analysis and when preparing data for complex analyses, ensuring critical information is preserved while redundant details are minimized.. This criterion delves into the eigenvalue spectrum of the covariance matrix C, leveraging the distribution of eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_p$ to shed light on the data's underlying structure [22].

Spectral density, in this scenario, encapsulates the eigenvalue distribution of C, offering insights into the variance across different dimensions. The formulation of spectral density $f(\lambda)$ as

$$f(\lambda) = \frac{1}{p} \sum_{i=1}^{p} \delta(\lambda - \lambda_i),$$

where δ denotes the Dirac delta function, serves as a quantitative representation of how eigenvalues are dispersed, highlighting the empirical distribution of these crucial markers of variance [22].

Estimation of spectral density can employ methodologies like kernel density estimation or principles from random matrix theory, aiming to smooth out the empirical distribution $\hat{f}(\lambda)$ to discern the data's signal from noise. This estimation seeks to pinpoint 'edges' within the eigenvalue spectrum, zones where a significant shift from larger (signal) to smaller (noise) eigenvalues occurs, thus identifying components that are truly informative [22].

The stopping condition for component selection, thus, revolves around analyzing $\hat{f}(\lambda)$ to identify the largest eigenvalue λ_k that stands distinctly apart from the spectrum's bulk, marking the transition from signal to noise. Such a criterion not only mathematically substantiates the bifurcation of signal and noise within the dataset but also illuminates the intrinsic dimensionality critical for PCA's dimensionality reduction goals [22].

In essence, the Spectral Density Estimation Criterion introduces a methodical, mathematically grounded approach to discerning significant principal components within PCA. By focusing on the eigenvalue spectrum and its nuanced interpretation, it delineates a path to differentiating meaningful components from the superfluous, thus enhancing PCA's efficiency and interpretability. This criterion, anchored in the principles of statistical signal processing and random matrix theory, ensures a principled analysis that profoundly respects the data's spectral properties, offering a clear vista into the dimensionality and structure intrinsic to the dataset.

5.4 Matrix Perturbation-based Criterion

Within the framework of Principal Component Analysis (PCA), the Matrix Perturbation-based Criterion introduces an advanced objective: to identify orthogonal principal components that not only maximize the variance of the projected data but also demonstrate resilience to data perturbations. This dual focus acknowledges the inevitability of noise and minor alterations within real-world datasets, aiming to ensure the derived components remain robust under such conditions [23].

The essence of matrix perturbation lies in evaluating the stability of principal components against slight modifications to the original data matrix A, yielding a perturbed variant A'. The stability measure δ, defined as the norm difference $\|A - A'\|$, quantifies the impact of these perturbations, with norms like the Frobenius norm offering a practical measure of deviation [23].

Eigenvalue perturbation theory underpins this criterion by establishing that minor adjustments to the matrix induce proportional shifts in its eigenvalues and eigenvectors. Since PCA's principal components are directly derived from the eigenvectors of the covariance matrix, the criterion prioritizes components that exhibit minimal variation in their eigenvalues and eigenvectors in response to perturbations [23].

Stability assessment hinges on monitoring the eigenvalues and eigenvectors' responsiveness to changes, employing theoretical constructs such as the Davis-

Kahan theorem for a precise quantification of stability. The algorithm identifies an optimal stopping point by scrutinizing the stability measure δ, with a significant uptick in δ signaling a component's instability or susceptibility to noise, thereby guiding the decision to exclude such components from the final model[23].

This approach underscores the importance of robustness in the components selected by PCA, positing that stability amidst perturbations signifies a component's authenticity in capturing the data's inherent structure, rather than mere artifacts or noise. Moreover, it emphasizes the generalizability of stable components, positing that their consistency across varied conditions renders them more reliable for depicting the true data structure[23].

In conclusion, the Matrix Perturbation-based Criterion enriches PCA with a mathematically solid strategy for component selection, integrating eigenvalue perturbation theory to safeguard against the inclusion of components that, while potentially variance-maximizing, lack stability and might not accurately represent the underlying data structure. This criterion thus elevates the generalizability and reliability of PCA, ensuring the model's efficacy in capturing genuine data characteristics amidst the omnipresent challenge of dataset perturbations.

5.5 Optimizing PCA with Cross-Validation: A Mathematical Perspective

Theorem 8 (Optimality of the PCA Projection Matrix)**.** *Let $X \in \mathbb{R}^{n\times p}$ be a centered data matrix. Consider the optimization problem*

$$\max_{W\in\mathbb{R}^{p\times k}} \operatorname{Tr}(W^T X^T X W) \quad \textit{subject to } W^T W = I.$$

Then, a maximizer W^* exists and is given by the matrix whose columns are the eigenvectors corresponding to the k largest eigenvalues of $X^T X$. Moreover, the gradient of the function

$$f(W) = \operatorname{Tr}(W^T X^T X W)$$

with respect to W is given by

$$\nabla_W f(W) = 2X^T X\, W.$$

Proof. Since the set

$$\{W \in \mathbb{R}^{p\times k} : W^T W = I\}$$

(i.e., the Stiefel manifold) is compact and $f(W)$ is continuous, the Weierstrass theorem guarantees the existence of a maximizer.

An alternative argument relies on matrix analysis. Extend the columns of W to form an orthogonal matrix $\tilde{W} \in \mathbb{R}^{p \times p}$. Then, the matrix

$$\tilde{W}^{\top}(X^T X)\tilde{W}$$

has the same eigenvalues as $X^T X$, and $W^T(X^T X)W$ is a principal submatrix of it. By the Cauchy Interlacing Theorem (see Proof), for each $j = 1, \ldots, k$,

$$\lambda_j(X^T X) \;\; \geq \;\; \lambda_j\big(W^T(X^T X)\,W\big).$$

Summing these inequalities over j shows that

$$\sum_{j=1}^{k} \lambda_j(X^T X) \;\; \geq \;\; \mathrm{Tr}\big(W^T(X^T X)\,W\big).$$

Thus, the maximum of $\mathrm{Tr}(W^T X^T X W)$ is attained when W consists of the eigenvectors corresponding to the k largest eigenvalues of $X^T X$.

Finally, because $X^T X$ is symmetric, standard results in matrix calculus yield

$$\nabla_W \, \mathrm{Tr}(W^T X^T X W) = 2X^T X \, W.$$

□

Constructing a Practical Solution. While this theorem identifies the *optimal* rank-k subspace (the one spanned by the top k eigenvectors of $X^T X$), it does not by itself specify how to *compute* those eigenvectors in practice. In the next part, we present the steps and proofs that underlie a practical method for obtaining W^*. We then integrate cross-validation to choose the number of principal components k.

Reconstruction Error and Cross-Validation. The reconstruction error in PCA is defined as

$$E(k) = \|X - X\,W\,W^T\|_F^2,$$

which quantifies the loss incurred when projecting X onto the subspace spanned by the columns of W. Cross-validation is then used to determine the optimal number of principal components k^* by partitioning the data and selecting the k that minimizes the average reconstruction error over the validation sets [20]. This data-driven approach balances model complexity with reconstruction fidelity, thereby enhancing generalizability.

Hence, by leveraging the above theoretical result in tandem with a constructive eigen-decomposition (or SVD) and a cross-validation scheme, we obtain both a rigorous foundation for PCA and a practical strategy for model selection.

5.6 Justification for the Use of Hybrid Spectral Density Estimation Criterion

Historically, criteria such as the Eigenvalue Threshold, Percentage of Total Variance, Scree Plot, Parallel Analysis, and the Minimum Average Partial (MAP) Criterion have provided frameworks for determining the stopping point in PCA. These methodologies range from heuristic to statistically rigorous approaches, each with its unique perspective on capturing the essence of the data's structure and variance.

While these criteria have proven effective in various contexts, they are not without limitations. For instance, the Eigenvalue Threshold and Scree Plot are somewhat subjective and may not consistently capture the underlying data complexity across different scenarios. On the other hand, Parallel Analysis and the MAP Criterion, despite their robustness, might present computational challenges and interpretability issues.

Given these considerations, an alternative stopping criterion that seeks to address these limitations, while harnessing the strengths of existing methods, is justified. The criterion is inspired by the Information Theoretic Criterion (IC), which utilizes entropy to balance information content against dimensionality reduction. However, the new approach diverges from the IC by incorporating insights from spectral density estimation and matrix perturbation theories to enhance the criterion's adaptability and robustness, particularly in the face of high-dimensional and noisy datasets.

The motivation stems from the desire to offer a more dynamic and context-sensitive criterion that adapts to the varying signal-to-noise ratios inherent in real-world data, providing a more nuanced understanding of the dataset's structure. Furthermore, it ensures robustness by factoring in the stability of principal components against data perturbations, thereby enhancing the reliability of the dimensionality reduction process. Finally, using the spectral density estimation to systematically identify the transition from signal-dominant to noise-dominant eigenvalues, offers a data-driven approach to determining the number of components.Mathematical results for KDE

Compared to the Information Criterion proposed by Ubaru, the alternative criterion offers several advantages. By integrating spectral density insights, the criterion dynamically adjusts to the data's inherent complexity, providing a tailored approach to component selection. The incorporation of matrix perturbation analysis ensures that the selected components are not only informative but also stable, making the PCA outcome more reliable across different applications.

In line with the above discussion, an outlined stopping criteria based on spectral edge detection should be feasible. The principal aim of employing spectral edge detection is to accurately identify the inherent dimensionality of the data. This method efficiently separates the significant eigenvalues that represent the true signal from the lesser ones dominated by noise. By doing so, it ensures that dimensionality reduction focuses on retaining the most informative components of the data, which is crucial for applications like Principal Component Analysis (PCA), spectral clustering, or any task that benefits from understanding the underlying structure of the dataset.[24], [25]

Traditional methods often rely on fixed thresholds or predetermined ranks, which may not be suitable across different datasets or in scenarios where the signal-to-noise ratio varies significantly. The proposed criteria adaptively estimate the rank based on the actual distribution of eigenvalues, making it a more flexible and universally applicable approach. This adaptability is particularly beneficial in complex real-world applications where the data properties are not known a priori.

Incorporating a noise threshold in the spectral edge detection process addresses one of the major challenges in automated rank determination—noise. By setting a threshold, the criteria ensure that the identified spectral edge is not merely a product of random fluctuations but a meaningful transition in the eigenvalue spectrum. This robustness to noise is essential for applications involving real-world data, which is invariably imperfect and noisy.

By determining an appropriate stopping point based on the spectral edge, the algorithm can terminate early once the meaningful components have been identified, thereby avoiding unnecessary computations. This efficiency is crucial for handling large datasets or when the algorithm is part of a larger pipeline where computational resources are shared among multiple tasks.

The stopping criteria also includes a fallback strategy that ensures that the algorithm remains practical and produces a reasonable output even when the data does not exhibit a clear spectral edge. This aspect of the criteria guards against potential edge cases, ensuring the algorithm's utility across a wide range of datasets and scenarios.

6 The Revised Ubaru Algorithm

Below, the Hybrid Spectral Density Estimation Criterion is incorporated into the original Ubaru Algorithm framework. Consequently, the information criterion previously employed by Ubaru is replaced with the newly proposed stopping criterion.

Developed Algorithm [26], [16], [18], [24], [25] Details in Appendix

1. Initialize variables: $Q = [\ ]$ to store the orthonormal basis vectors; $k = 1$ to index the current iteration; $m = \log(p)/\sqrt{\epsilon}$ to set the depth of Krylov subspace; and *all_eigenvalues* $= [\ ]$ to accumulate eigenvalues for spectral edge detection. The input matrix $X \in \mathbb{R}^{n\times p}$ represents the data.
2. For $k = 1$ to p do
    (a) Generate a random vector v_k with $\|v_k\|_2 = 1$.
    (b) $K = \frac{1}{n}[Xv_k; (XX^T)Xv_k; \ldots; (XX^T)^{m-1}Xv_k]$.
    (c) $Q = \text{orth}([Q; K])$ to include orthogonal vectors up to the current iteration, ensuring the matrix Q has orthogonal columns.
    (d) $T = \frac{1}{n}Q^TXX^TQ$.
    (e) $[V, \Theta] = \text{eig}(T)$.
    (f) Append the eigenvalues Θ to the list *all_eigenvalues* for subsequent spectral edge detection. **Note:** Although new eigenvalues are computed in each iteration, they are obtained from the small matrix T (of size $m \times m$), which is much more efficient than computing all eigenvalues of the full matrix.
    (g) Every 5 iterations, or at the end, evaluate the revised stopping criteria for spectral edge detection:
        i. If $(k \mod 5 == 0)$ or $(k == p)$:
            A. Use kernel density estimation (KDE) on the accumulated *all_eigenvalues* to identify their distribution.Appendix
            B. Detect the spectral edge by finding significant changes in the density of eigenvalues, indicative of the optimal rank q.
            C. If a spectral edge is identified, set q based on this edge and exit the loop.
3. End for.
4. If no spectral edge is found by the last iteration, choose q based on an alternative criterion, such as the maximum k reached or a predefined threshold.

5. Output the optimal rank q, the dimensionality-reduced data representation $Y = QV$, and the orthonormal basis Q. Here, V corresponds to the eigenvectors selected based on the identified spectral edge or alternative criteria.

The algorithm leverages a series of steps to meticulously compute and analyze the eigenvalues of the Krylov subspace approximation of matrix X, employing the `Lanczos_update` function. These eigenvalues, critical for discerning the matrix's rank, are systematically gathered in the array `all_eigenvalues`. To distill the essence of these eigenvalues, kernel density estimation is performed at every fifth iteration, utilizing the `ksdensity` function to reveal the eigenvalue distribution's landscape.

Intriguingly, the algorithm employs the `findpeaks` function on the inverted density estimate to identify spectral edges—valleys within the density plot that signify the transition from signal to noise in the eigenvalue spectrum. This edge detection is pivotal, marking the threshold beyond which eigenvalues are predominantly associated with noise. An essential step involves adjusting the detected spectral edge (`edge_index`) against a `noise_threshold`, ensuring the edge is not underestimated in the presence of noise characterized by `sigma`.

Rank estimation, denoted as q_{spectral}, hinges on quantifying eigenvalues that surpass the noise-adjusted threshold, thereby encapsulating dimensions that rise above noise. An early stopping mechanism is incorporated, halting further iterations once a spectral edge is detected, signifying a reliable rank estimation based on spectral analysis.

Subsequent to determining q_{spectral}, the algorithm computes the cumulative variance from the eigenvalues, aiming to identify q_{variance}—the count of eigenvalues necessary to satisfy a predefined variance threshold. The final rank q emerges from comparing q_{spectral} and q_{variance}, selecting the maximum as the definitive estimate to ensure a robust account of both spectral and variance information.

The algorithm's stopping criterion transcends mere variance threshold attainment; it opts for the higher rank estimate between q_{spectral} and q_{variance}, ensuring a comprehensive and accurate rank determination. This methodology presupposes the matrix's rank is inferable from its eigenvalues and hinges on an accurate estimation of the noise level (`sigma`) for effectiveness. It acknowledges potential limitations in matrices lacking a clear spectral edge or those with slowly decaying eigenvalues, alongside computational efficiency concerns tied to matrix size and Lanczos iteration count m.

By integrating spectral analysis with variance estimation, this function unveils a nuanced approach to rank estimation, especially apt for large or noiseafflicted datasets, underscoring its utility in extracting meaningful dimensions that genuinely reflect the underlying data structure.

6.1 Experiments

In this section, the results of the four experiments that compare the algorithm proposed in the Ubaru with the hybrid Spectral Density Estimation Criterion is analyzed. The experiments aim to assess the effectiveness of rank estimation and approximation methods on primarily synthetic data.

6.1.1 Experiment Setup

Before delving into the comparative analysis, it is crucial to understand the parameters that define the synthetic data generation and the metrics used for algorithm evaluation. These parameters not only influence the behavior and performance of the algorithms but also frame the context for interpreting the results:

- Synthetic data parameters:
    1. n: Number of samples (5000, 5000, 10000)
    2. q_{actual}: Actual rank (50, 100, 100)
    3. λ_q: Signal strength (5)
    4. Bandwidth: 2% (used in ksdensity)
    5. Sparsity: 0.1
    6. Sigma: σ (0.1 for synthetic data generation, 1 for noise level in Krylov dimension estimation)
    7. m: Number of Lanczos steps per singular value (10)

With these parameters set, we proceed to the empirical evaluation of the algorithms. The table below presents a comprehensive comparison based on computational efficiency, accuracy, and the ability to explain the variance within the data.

6.2 Results of Experiments

The results from the table 2 offer a quantitative perspective on the performance of the two algorithms. Notably, the Revised Ubaru Algorithm consistently exhibits lower computational times across both synthetic and real

Table 2: Comparison of Algorithm Performance on Synthetic and Real Data

Experiment	Method	Total Time (secs)	Estimated Rank ($\hat{q}$)	Frobenius Norm Error	Variance Explained (%)
Synthetic Data ($n = 5000, q_{\text{actual}} = 50$)	Ubaru's Algorithm	50.86	51	3222.10	-
	Revised Ubaru Algorithm	4.34	50	217.79	99.9462
Synthetic Data ($n = 5000, q_{\text{actual}} = 100$)	Ubaru's Algorithm	119.81	100	4742.54	-
	Revised Ubaru Algorithm	4.50	50	2441.46	91.7998
Synthetic Data ($n = 10000, q_{\text{actual}} = 100$)	Ubaru's Algorithm	356.43	101	9345.43	-
	Revised Ubaru Algorithm	15.20	50	5022.06	91.3146
Real Data (lpi_ceria3d)	Ubaru's Algorithm	1.19	100	141.42	-
	Revised Ubaru Algorithm	0.18	50	93.24	61.8404

datasets. For instance, on the synthetic dataset with $n = 5000$ and $q_{\text{actual}} = 50$, the Revised Ubaru Algorithm completes in 4.34 seconds as opposed to 50.86 seconds required by the Ubaru's Algorithm. Furthermore, the Revised Ubaru Algorithm demonstrates remarkable accuracy, with a Frobenius Norm Error of 217.79 compared to 3222.10 by the Ubaru's Algorithm for the same dataset.

The estimated rank $\hat{q}$ provided by the Revised Ubaru Algorithm aligns precisely with the actual rank in the first case, while the Ubaru's Algorithm overestimates slightly with $\hat{q} = 51$. This pattern of efficient estimation by the Revised Ubaru Algorithm and slight overfitting by the Ubaru's Algorithm persists across the other datasets as well.

The variance explained by the Revised Ubaru Algorithm is notably high, especially in the case of synthetic data where it reaches up to 99.9462%. This indicates a high retention of meaningful data characteristics despite the method's computational efficiency.

These numerical observations lay the foundation for the subsequent commentary, which delves into the implications of these findings.

6.3 Comment

Building on the numerical analysis, the series of experiments conducted to compare the algorithm proposed in the Ubaru with the hybrid Spectral Density Estimation Criterion reveal insightful contrasts.

The algorithm from the Ubaru, while robust in its dimensionality reduction capabilities, appears to exhibit a tendency towards overfitting. This is characterized by higher Frobenius norm errors, which suggest an excessive fit to the noise within the data rather than the underlying signal. Additionally, the computational time is significantly longer, indicating a more complex and resource-intensive process.

Conversely, the hybrid Spectral Density Estimation Criterion shows a predilection for efficiency and parsimony. It achieves lower approximation errors and faster computation times, indicating a focus on capturing the most influential data components. This method seems to prioritize marginal utility, emphasizing the extraction of the most pertinent features of the data while discarding the rest as noise.

The implications for practical application are straight forward: if the objective is to capture the fullest extent of the data's variability and computational resources are not a limiting factor, the Ubaru's algorithm may be the preferred choice. However, in scenarios where time efficiency and avoidance of overfitting are paramount, the hybrid Spectral Density Estimation Criterion offers a compelling advantage. It facilitates quicker, more streamlined data processing, which could be particularly beneficial in real-time analytics or when operating under computational constraints.

Ultimately, the selection of an algorithm should be aligned with the specific needs of the task at hand, balancing the trade-offs between complexity, computation time, and the level of detail required in the data representation.

7 Discussion and Conclusions

Reflecting on the exploration of alternative stopping criteria in Principal Component Analysis (PCA), several avenues for future research emerge as particularly promising. First, there's a clear opportunity for refining the Spectral Density Estimation approach, especially for high-dimensional datasets, with the aim of boosting computational efficiency and minimizing approximation errors. Developing customizable stopping criteria that adapt to the unique characteristics of different datasets, especially in rapidly evolving domains like bioinformatics and quantum computing, presents another fertile ground for investigation. Additionally, the potential integration of these advanced stopping criteria within deep learning frameworks could revolutionize training efficiency and model accuracy, marking a significant leap forward in machine learning methodologies.

The comprehensive analysis, juxtaposing traditional PCA with the innovative hybrid Spectral Density Estimation Criterion, has unveiled critical insights. Traditional PCA, despite its prowess in dimensionality reduction, tends to exhibit a propensity for overfitting, as evidenced by elevated Frobenius norm errors, suggesting an undue emphasis on noise. This is coupled with increased computational demands. Conversely, the Spectral Density Estimation Criterion distinguishes itself through enhanced efficiency and a parsimonious approach, characterized by lower approximation errors and expedited processing times, indicating a strategic focus on distilling the most impactful data components.

These findings have profound implications for practical application, suggesting that algorithm selection should be contextually driven. In scenarios where capturing the full spectrum of dataset variability is paramount and resources are plentiful, traditional PCA might be preferable. However, in contexts where minimizing overfitting and maximizing time efficiency are crucial, the Spectral Density Estimation Criterion emerges as the superior choice, especially suitable for real-time analytics and environments with constrained computational resources. This underscores the importance of aligning algorithm choice with the specific demands of each analytical task, carefully weighing computational complexity against the necessity for timely and accurate data representation.

In conclusion, the strategic selection of a dimensionality reduction algorithm in PCA transcends mere technicality, embodying a critical decision that profoundly impacts data analysis outcomes. This research underscores the necessity of a thoughtful, context-sensitive approach to choosing between different algorithms, aiming to harness their respective strengths in accordance with the objectives and constraints of each unique data analysis endeavor. As we navigate the complexities of data science, the insights gleaned from this study promise to illuminate the path toward more informed, effective applications of dimensionality reduction techniques, thereby enhancing the quality and utility of data-driven insights.

A Matlab code

A.1 The Ubaru Algorithm

```matlab
function main
    % Synthetic Data Generation
    % Define the dimensions of the synthetic matrix and sparsity
    n = 4000; % Samples
    k = 100; % Features
    % Generate a sparse random matrix A
    A = sprand(n, k, 0.1);
    % Create a diagonal matrix L with linearly increasing values
    L = diag(5:0.5:54.5);
    % Define the noise level for the synthetic data generation
    sigma_noise = 0.001;
    % Generate the synthetic data matrix X with added noise
    X = A * L * A' + sigma_noise * sprand(n, n, 0.1);

    % Setting Parameters for rank estimation
    % Define the number of Lanczos steps to be performed
    msteps = 10;
    % Define the noise level parameter for the Krylov_dimension function
    sigma_noise_level = 0.01;

    % Computing Rank using the Krylov subspace method
    % Call Krylov_dimension function to estimate the rank (q) and get the principal components (Y)
    [Y, q] = Krylov_dimension(X, msteps, sigma_noise_level, min(k, n));

    % Analyzing and Visualizing Results
    % Compute the singular values of matrix X
    d = svd(X);
    % Plot the singular values
    figure();
    plot(1:min(n, k), d, 'bo-');
    hold on;
    % Highlight the estimated rank on the plot
    stem(q, 1.1 * d(1), 'r*-');
    legend('Singular values', 'Estimated rank');
    xlabel('i');
    ylabel('\sigma_i');
    title('Rank estimation using Krylov method');
    hold off;
end

function [Y,q] = Krylov_dimension(X, m, sigma, max_rank)
    % Initialize variables and compute the Frobenius norm of X
    [p,n] = size(X); % Dimensions of the input matrix
    IC = zeros(min(p,n),1); % Information criterion values
    Xfro = norm(X,'fro')^2; % Frobenius norm squared of X
    Snfro = (Xfro^2 - 2*sigma*Xfro + p*sigma^2); % Adjusted norm for covariance estimation
    Cn = 1*log(n); % Scaling factor based on the size of X

    % Initial random vector for Lanczos iterations
    v = randn(p,1); V = v/norm(v);
    T = []; % Initialize tridiagonal matrix

    % Iteratively perform Lanczos steps and estimate rank
    for k = 1:max_rank
        [U, V, T, Theta] = Lanczos_update(X, V, T, m); % Perform a step of the Lanczos update

        k1 = min(k,length(Theta)); % Adjust for the actual number of steps performed
        % Calculate the information criterion for rank estimation
        IC(k) = (n/(2*sigma^2))*(Snfro - sum(Theta(1:k1))) - Cn*(p-k)*(p-k-1)/2;

        % Break loop if the information criterion increases, indicating optimal rank found
        if (k > 1 && IC(k) > IC(k-1))
            break;
        end
    end

    q = k-1; % Estimated rank
    % Compute the principal components based on the estimated rank
    if size(V,2) > q
        Y = V(:,1:q)*U(1:q,1:q);
    else
        Y = V*U; % Return all computed components if q exceeds the dimensions
    end
end

function [U, V, T, theta] = Lanczos_update(X, V, T, msteps)
    % Initialize variables for Lanczos update
    n = size(X,2); % Number of columns in X
    m = msteps+1; % Adjust for MATLAB indexing
    V1 = zeros(n,m); % Placeholder for new Lanczos vectors (not used, to be removed for clarity)
    Tmat = zeros(m,m); % Placeholder for tridiagonal matrix updates (not used, to be removed for clarity)
    k1 = size(V,2); % Current size of V
    v = V(:,end); % Last computed Lanczos vector
    beta = 0; % Initialization for the off-diagonal elements of T
    vold = v; % Placeholder for the previous Lanczos vector
    orthTol = 1.e-08; % Orthogonality tolerance for reorthogonalization
    wn = 0.0; % Norm tracking for orthogonality test

    % Perform msteps Lanczos iterations
    for k=1:msteps
        w = X*(X'*v); % Matrix-vector product
        w = w - beta*vold; % Subtract the previous vector scaled by beta
        alpha = w'*v; % Diagonal element of T
        wn = wn + alpha*alpha; % Update norm
        T(k1+k-1,k1+k-1) = alpha; % Update T
        w = w - alpha*v; % Orthogonalize w with respect to v

        % Full reorthogonalization
        t = V'*w; % Project w onto the space spanned by V
        w = w - V*t; % Subtract the projection
        beta = w'*w; % Compute the new beta
        % Break if the vector is almost orthogonal to the subspace
        if (beta*k < orthTol*wn)
            break;
        end
        wn = wn+2.0*beta; % Update norm
        beta = sqrt(beta); % Take the square root for the next iteration
        vold = v; % Update old vector
        v = w/beta; % Normalize w for the next vector
        V(:,k1+k) = v; % Append the new vector to V
        T(k1+k-1,k+k1) = beta; % Update off-diagonal elements of T
        T(k+k1,k1+k-1) = beta; % Symmetric update
    end

    [U,theta] = eig(T(1:k1+k-1,1:k1+k-1)); % Compute eigenvalues and eigenvectors of T
    [theta, ~] = sort(diag(theta),'descend'); % Sort eigenvalues in descending order
end
```

A.2 The Revised Ubaru Algorithm

```matlab
function [Y, q] = Krylov_dimension_spectral_sigma(X, m, sigma, max_rank, variance_threshold)
    % Krylov_dimension_spectral: Estimates the rank of a matrix using the
    % Krylov subspace method with spectral density estimation.
    % Inputs:
    % X - Data matrix whose rank is to be estimated
    % m - Number of Lanczos steps per singular value
    % sigma - Noise level
    % max_rank - Maximum rank to consider for the estimation
    % variance_threshold - Threshold for cumulative variance to determine rank
    %
    % Outputs:
    % Y - Matrix formed from the Krylov subspace
    % q - Estimated rank of the matrix

    % Initialization and parameter setup
    [p, n] = size(X);  % Get the dimensions of X
    all_eigenvalues = [];  % Initialize an empty array to store eigenvalues
    q = max_rank;  % Start with q equal to max_rank
    noise_threshold = sigma^2 * p;  % Assuming noise variance of sigma^2 for each dimension

    % Main routine
    v = randn(p, 1);  % Start with a random vector
    V = v / norm(v);  % Normalize the vector
    T = [];  % Initialize T, which will be used in Lanczos update

    for k = 1:max_rank
        % Perform Lanczos update to compute the Krylov subspace
        [U, V, T, Theta] = Lanczos_update(X, V, T, m);
        all_eigenvalues = [all_eigenvalues; Theta];  % Collect all eigenvalues

        % Spectral edge detection
        if mod(k, 5) == 0
            % Density estimation for eigenvalues
            [f, xi] = ksdensity(all_eigenvalues, 'Bandwidth', 0.5);
            [~, locs] = findpeaks(-f);
            if ~isempty(locs)
                % If spectral edge is found, update the rank estimate
                edge_index = xi(locs(1));
                % Adjust edge_index by noise threshold
                edge_index = max(edge_index, noise_threshold);
                q_spectral = sum(all_eigenvalues > edge_index);
                if ~isempty(q_spectral)
                    q = q_spectral;
                end
                break;
            end
        end
    end

    % Default handling if no spectral edge is found
    if isempty(locs)
        q = q_spectral;
        if isempty(q)
            q = min(10, max_rank);  % Set a default rank value
        end
    end

    % Cumulative variance calculation for rank estimation
    sorted_eigenvalues = sort(all_eigenvalues, 'descend');
    cumulative_variance = cumsum(sorted_eigenvalues) / sum(sorted_eigenvalues);
    q_variance = find(cumulative_variance >= variance_threshold, 1, 'first');

    % Final q is the maximum of spectral and variance estimates
    q = max(q, q_variance);

    % Adjust the size of V to match U for matrix multiplication
    if size(V, 2) > size(U, 1)
        V = V(:, 1:size(U, 1));
    end
    Y = V * U;  % Construct Y from the Krylov subspace
end
```

B The Components and Reasoning Behind Spectral Density Estimation Criteria

B.1 Kernel Density Estimation

Before delving into the technical details, we briefly lay the foundation for Kernel Density Estimation (KDE), a non-parametric method used to approximate the probability density function of a random variable. In the following, we derive KDE starting from the sample's characteristic function, then apply Fourier inversion with a damping function to obtain a practical density estimator. This exposition not only clarifies the theoretical underpinnings of KDE but also sets the stage for its application in our spectral edge detection method.

Characteristic Function Estimation: Given a sample $(x_1, x_2, \ldots, x_n)$, the characteristic function $\varphi(t) = \mathbb{E}[e^{itX}]$ is naturally estimated as [24], [25]:

$$\hat{\varphi}(t) = \frac{1}{n} \sum_{j=1}^{n} e^{itx_j}$$

This formula represents an estimator for the characteristic function of the sample.

Probability Density Function Inversion: The corresponding probability density function can be theoretically obtained by applying the Fourier transform to $\hat{\varphi}(t)$. However, this direct inversion often leads to diverging integrals.

To address the divergence issue, the estimated characteristic function $\hat{\varphi}(t)$ is multiplied by a damping function $\psi_h(t) = \psi(ht)$, where ψ is typically a uniform or Gaussian function and h is the bandwidth parameter.

Density Estimator with Damping Function: The density estimator is given by:

$$\hat{f}(x) = \frac{1}{2\pi} \int_{-\infty}^{+\infty} \hat{\varphi}(t)\psi_h(t)e^{-itx}\, dt$$

This integral is the inverse Fourier transform of the product of $\hat{\varphi}(t)$ and $\psi_h(t)$.

By transforming and simplifying the above integral, we obtain the Kernel Density Estimator:

$$\hat{f}(x) = \frac{1}{nh} \sum_{j=1}^{n} \frac{1}{2\pi} \int_{-\infty}^{+\infty} e^{-i(ht)\frac{x-x_j}{h}} \psi(ht)\, d(ht) = \frac{1}{nh} \sum_{j=1}^{n} K\left(\frac{x-x_j}{h}\right)$$

where K is the Fourier transform of the damping function ψ. The scaled kernel is defined as $K_h(x) = \frac{1}{h} K\left(\frac{x}{h}\right)$. The choice of h affects the trade-off between the bias and variance of the estimator.

The KDE, a widely used method for its practicality and effectiveness, is shown to be derived from the characteristic function of a sample using Fourier analysis and a damping function, illustrating its theoretical foundations.

Step 1: Standard KDE
As we have shown, the Kernel Density Estimation, is used to approximate the probability density function (pdf) of a random variable. The KDE at a point x is given by:

$$\hat{f}_h(x) = \frac{1}{nh} \sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right)$$

Here's what each term represents:

- $\hat{f}_h(x)$ is the estimated density at point x.
- x is the point at which to estimate the density.
- x_i are the sample data points.
- n is the number of data points.
- $K(\cdot)$ is the kernel function, which is a non-negative function integrating to one, and often symmetric. Common choices are Gaussian, Epanechnikov, and uniform kernels.
- h is the bandwidth, a parameter that controls the smoothness of the density estimate.

Step 2: Reflection Method
To address boundary bias, data points are reflected around the lower and upper boundaries, L and U. The reflected points are computed as follows:

$$x_i^- = 2L - x_i \quad \text{and} \quad x_i^+ = 2U - x_i$$

Step 3: Modified KDE with Reflections
The KDE is modified to incorporate the reflections:

$$\hat{f}_h(x) = \frac{1}{nh} \sum_{i=1}^{n} \left[K\left(\frac{x - x_i}{h}\right) + K\left(\frac{x - x_i^-}{h}\right) + K\left(\frac{x - x_i^+}{h}\right) \right]$$

Step 4: Cumulative Distribution Function (CDF)
To construct the CDF from the pdf, we integrate the density estimate. The CDF at a point x using KDE is:

$$\hat{F}_h(x) = \int_{-\infty}^{x} \hat{f}_h(t)\, dt$$

When including the reflection, the integral includes the original data points and the reflected data:

$$\hat{F}_h(x) = \frac{1}{n}\sum_{i=1}^{n}\int_{-\infty}^{x}\left[K\left(\frac{t-x_i}{h}\right)+K\left(\frac{t-x_i^-}{h}\right)+K\left(\frac{t-x_i^+}{h}\right)\right]dt$$

Step 5: Integration with Reflection
We integrate each term separately using the cumulative kernel function $G(\cdot)$:

$$G(z) = \int_{-\infty}^{z} K(t)\,dt$$

The CDF estimate with reflections becomes:

$$\hat{F}_h(x) = \frac{1}{n}\sum_{i=1}^{n}\left[G\left(\frac{x-x_i}{h}\right)+G\left(\frac{x-x_i^-}{h}\right)+G\left(\frac{x-x_i^+}{h}\right)\right]$$

Step 6: Normalization over Interval $[L, U]$
To normalize the CDF so that $\hat{F}_h(L) = 0$ and $\hat{F}_h(U) = 1$, we adjust the CDF by subtracting the CDF value at L from the CDF at any point x:

$$\begin{aligned}\hat{F}_h(x) - \hat{F}_h(L) = &\frac{1}{n}\sum_{i=1}^{n}\left[G\left(\frac{x-x_i}{h}\right)+G\left(\frac{x-x_i^-}{h}\right)+G\left(\frac{x-x_i^+}{h}\right)\right]\\ &-\frac{1}{n}\sum_{i=1}^{n}\left[G\left(\frac{L-x_i}{h}\right)+G\left(\frac{L-x_i^-}{h}\right)+G\left(\frac{L-x_i^+}{h}\right)\right]\end{aligned}$$

B.2 Key Theoretical Results for Spectral Density Estimation

The Spectral Density Estimation and Lanczos update use a stochastic sampling and averaging technique to obtain an approximate of Spectral Density. [27]

Theorem 9. *Let A be a real symmetric matrix of dimension $n \times n$ with eigendecomposition $A = \sum_{j=1}^{n} \lambda_j u_j u_j^T$, where $u_i^T u_j = \delta_{ij}$, δ_{ij} being the Kronecker delta symbol. Let v be a vector of dimension n, represented as the linear combination of $\{u_i\}_{i=1}^{n}$ as $v = \sum_{j=1}^{n} \beta_j u_j$. If each component of v is obtained independently from a normal distribution with zero mean and unit standard deviation, then $E[\beta_i \beta_j] = \delta_{ij}$, and the trace of a matrix function $f(A)$ can be approximated by averaging $v^T f(A) v$.*

Proof. Let A be a real symmetric matrix of dimension $n \times n$ with eigendecomposition:

$$A \in \mathbb{R}^{n \times n}, \quad A = \sum_{j=1}^{n} \lambda_j u_j u_j^T$$

where λ_j are the eigenvalues, and u_j are the corresponding orthonormal eigenvectors, satisfying $u_i^T u_j = \delta_{ij}$. Vector v is defined as:

$$v = \sum_{j=1}^{n} \beta_j u_j$$

Each component of v follows a normal distribution with zero mean and unit variance.

Properties of β_j Expectation and Variance:

$$E[v] = 0, \quad E[vv^T] = I$$

Here, I is the identity matrix. The coefficients β_j are linear combinations of normally distributed variables, thus $\beta_j \sim \mathcal{N}(0, 1)$.

Independence:

$$E[\beta_i \beta_j] = \delta_{ij}$$

This indicates that the expectation of the product of different β_i and β_j is 0 for $i \neq j$ and 1 for $i = j$.

Computing the Trace of $f(A)$:

$$\text{Tr}(f(A)) = \sum_{j=1}^{n} f(\lambda_j)$$

Averaging $v^T f(A)v$ expectation calculation:

$$E[v^T f(A)v] = E\left[\left(\sum_{i=1}^{n} \beta_i u_i^T\right) f(A) \left(\sum_{j=1}^{n} \beta_j u_j\right)\right]$$

Expanding and simplifying:

$$E[v^T f(A)v] = E\left[\sum_{i=1}^{n}\sum_{j=1}^{n} \beta_i \beta_j u_i^T f(A) u_j\right]$$

Due to the orthogonality of u_i and u_j, and $E[\beta_i \beta_j] = \delta_{ij}$, this simplifies to:

$$E[v^T f(A)v] = \sum_{j=1}^{n} E[\beta_j^2] f(\lambda_j)$$

Since $E[\beta_j^2] = 1$ for a normal distribution with zero mean and unit variance, we have:

$$E[v^T f(A)v] = \sum_{j=1}^{n} f(\lambda_j)$$

This sum is the trace of $f(A)$, showing that the average of $v^T f(A)v$ converges to the trace of $f(A)$ under the given conditions. □

B.3 Real World Example: Image Processing

We have high-resolution images, and our goal is to compress these images effectively without significant loss of quality. Each image is represented as a matrix.

Step 1: Eigenvalue Calculation
Eigenvalues of the matrix representing each image are calculated. These eigenvalues represent the variance captured by each principal component of the image data.

Step 2: Spectral Edge Detection in Image Compression
After calculating the eigenvalues for each image matrix, we use KDE to identify the spectral knee, which is the point that separates significant eigenvalues (representing important image features) from less significant ones (likely noise or redundant information). Imagine an image with a lot of detail in certain parts and less in others. The spectral knee helps us identify which eigenvalues (or features) are crucial to retain this detail and which can be considered redundant or noise.

Step 3: Rank Estimation
Here, we count the number of eigenvalues greater than the spectral knee. In our image, this would be akin to counting the number of features that are above a certain threshold of importance. If the spectral knee is too low, possibly indicating that it's picking up noise, we adjust it to a predefined threshold. In the image, this means ensuring we don't mistake unimportant details (like minor color variations) as crucial features.

Step 4: Variance-Based Rank Estimation
We also calculate the cumulative variance from the eigenvalues and determine the minimum number of eigenvalues needed to capture a substantial portion of the image's total variance. This method ensures we keep those features (eigenvalues) that contribute most to the image's overall structure and detail.

Step 5: Final Rank Determination
The final rank for image compression is determined by taking the maximum of the ranks obtained from the spectral knee and variance-based method. This ensures we don't miss out on important features (high variance) while also not including too many redundant features (beyond the spectral knee).

Step 6: Matrix Reconstruction
With the final estimated rank, the image is reconstructed using the principal components corresponding to the significant eigenvalues. This means we rebuild the image using only the most important features, reducing its size while maintaining as much of the original quality and detail as possible.

Conclusion

In this image compression scenario, the algorithm effectively balances data size reduction with quality preservation. By focusing on significant dimensions, the compressed image retains essential features, making this method suitable for efficient storage and processing of high-resolution images.

References

[1] Shashanka Ubaru and Yousef Saad. Fast methods for estimating the numerical rank of large matrices. In Maria Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 468–477, New York, New York, USA, 20–22 Jun 2016. PMLR. URL `https://proceedings.mlr.press/v48/ubaru16.html`.

[2] Shashanka Ubaru, Abd-Krim Seghouane, and Yousef Saad. Find the dimension that counts: Fast dimension estimation and krylov pca. 10 2018. URL `https://arxiv.org/abs/1810.03733`.

[3] Richard Bellman. *Introduction to Matrix Analysis.* SIAM, 2nd edition, 1997.

[4] Robb J. Muirhead. *Aspects of Multivariate Statistical Theory.* Wiley Series in Probability and Statistics. John Wiley & Sons, Inc., 1982. doi: 10.1002/9780470316559.

[5] Svante Wold, Kim Esbensen, and Paul Geladi. Principal component analysis. *Chemometrics and Intelligent Laboratory Systems*, 2(1):37–52, 1987. ISSN 0169-7439. doi: https://doi.org/10.1016/0169-7439(87)80084-9. URL `https://www.sciencedirect.com/science/article/pii/0169743987800849`. Proceedings of the Multivariate Statistical Workshop for Geologists and Geochemists.

[6] Y. Saad. The origin and development of krylov subspace methods. *Computing in Science & Engineering*, 24(04):28–39, 2022. ISSN 1558-366X. doi: 10.1109/MCSE.2022.3214388.

[7] Cameron Musco and Christopher Musco. Randomized block krylov methods for stronger and faster approximate singular value decomposition. *Advances in Neural Information Processing Systems*, 28, 2015. URL `https://proceedings.neurips.cc/paper_files/paper/2015/file/1efa39bcaec6f3900149160693694536-Paper.pdf`.

[8] Lloyd N. Trefethen and David III Bau. *Numerical Linear Algebra.* SIAM, 1997.

[9] Yousef Saad. *Iterative Methods for Sparse Linear Systems.* SIAM, 2003.

[10] G. H. Golub and C. F. Van Loan. *Matrix Computations.* Johns Hopkins University Press, 4th edition, 2013.

[11] R. A. Horn and C. R. Johnson. *Topics in Matrix Analysis.* Cambridge University Press, 1991.

[12] C. Eckart and G. Young. The approximation of one matrix by another of lower rank. *Psychometrika*, 1(3):211–218, 1936.

[13] L. Mirsky. Symmetric gauge functions and unitarily invariant norms. *The Quarterly Journal of Mathematics*, 11(1):50–59, 1960.

[14] Shashanka Ubaru. *Algorithmic Advances in Learning from Large Dimensional Matrices and Scientific Data*. Ph.d. dissertation, University of Minnesota, May 2018. URL `https://conservancy.umn.edu/bitstream/handle/11299/199004/Ubaru_umn_0130E_19195.pdf`. Accessed: 2023-11-03.

[15] Y. Saad. *Numerical Methods for Large Eigenvalue Problems*. SIAM, Philadelphia, revised edition, 2011. doi: 10.1137/1.9781611970739.

[16] Y. Saad. Krylov subspace methods for solving large unsymmetric linear systems. *Mathematics of Computation*, 37(155):105–126, 1981. ISSN 00255718, 10886842. URL `http://www.jstor.org/stable/2007504`.

[17] T. W. Anderson. Asymptotic theory for principal component analysis. *The Annals of Mathematical Statistics*, 34(1):122–148, 1963. doi: 10.1214/aoms/1177704248. URL `https://doi.org/10.1214/aoms/1177704248`.

[18] John R. Gilbert, Cleve Moler, and Robert Schreiber. Sparse matrices in matlab: Design and implementation. *SIAM Journal on Matrix Analysis and Applications*, 13(1):333–356, 1992. doi: 10.1137/0613024. URL `https://doi.org/10.1137/0613024`.

[19] Guanghan Xu and Thomas Kailath. Fast estimation of principal eigenspace using lanczos algorithm. *SIAM Journal on Matrix Analysis and Applications*, 15(3):974–994, 1994. doi: 10.1137/S0895479890183848. URL `https://doi.org/10.1137/S0895479890183848`.

[20] Hervé Abdi and Lynne J. Williams. Principal component analysis. *WIREs Computational Statistics*, 2(4):433–459. doi: https://doi.org/10.1002/wics.101. URL `https://wires.onlinelibrary.wiley.com/doi/abs/10.1002/wics.101`.

[21] Cui Yumeng and Fang Yinglan. Research on pca data dimension reduction algorithm based on entropy weight method. In *2020 2nd International Conference on Machine Learning, Big Data and Business Intelligence (MLBDBI)*, pages 392–396, 2020. doi: 10.1109/MLBDBI51377.2020.00084.

[22] Yongchun Li and Weijun Xie. Exact and approximation algorithms for sparse pca. *ArXiv*, abs/2008.12438, 2020. URL `https://api.semanticscholar.org/CorpusID:221370720`.

[23] Justin Eldridge, Mikhail Belkin, and Yusu Wang. Unperturbed: spectral analysis beyond davis-kahan. *ArXiv*, abs/1706.06516, 2017. URL `https://api.semanticscholar.org/CorpusID:4902109`.

[24] M. C. Jones. Simple boundary correction for kernel density estimation. *Statistics and Computing*, 3(3):135–146, sep 1993. ISSN 1573-1375. doi: 10.1007/BF00147776. URL `https://doi.org/10.1007/BF00147776`.

[25] Alejandro Quintela-Del-Río and Graciela Estévez-Pérez. Nonparametric kernel distribution function estimation with kerdiest: An r package for bandwidth choice and applications. *Journal of Statistical Software*, 50, 08 2012. doi: 10.18637/jss.v050.i08.

[26] Tomas Ruzgas and Indrė Drulytė. Kernel density estimators for gaussian mixture models. *Lietuvos statistikos darbai*, 52:14–21, 12 2013. doi: 10.15388/LJS.2013.13919.

[27] Lin Lin, Yousef Saad, and Chao Yang. Approximating spectral densities of large matrices. *SIAM Review*, 58(1):34–65, 2016. doi: 10.1137/130934283. URL `https://doi.org/10.1137/130934283`.