FACULTY OF INFORMATION TECHNOLOGY AND ELECTRICAL ENGINEERING

Charmin Asirimath

EXPANSIONS AND FACTORIZATIONS OF MATRICES AND THEIR APPLICATIONS

Master's Thesis
Degree Programme in Biomedical Engineering: Signal and Image Processing
June 2020

Asirimath C. (2020) Expansions and Factorizations of Matrices and Their Applications. University of Oulu, Degree Programme in Biomedical Engineering: Signal and Image Processing, 58 p.

ABSTRACT

Linear algebra is a foundation to decompositions and algorithms for extracting simple structures from complex data. In this thesis, we investigate and apply modern techniques from linear algebra to solve problems arising in signal processing and computer science. In particular, we focus on data that takes the shape of a matrix and we explore how to represent it as products of circulant and diagonal matrices. To this end, we study matrix decompositions, approximations, and structured matrix expansions whose elements are products of circulant and diagonal matrices. Computationally, we develop a matrix expansion with DCD matrices for approximating a given matrix. Remarkably, DCD matrices, i.e., a product of diagonal matrix, circulant matrix, and another diagonal matrix, give an natural extension to rank-one matrices. Inspired from the singular value decomposition, we introduce a notion of a matrix rank closely related to the expansion and compute the rank of some specific structured matrices. Specifically, Toeplitz matrix is a sum of two DCD matrices. Here, we present a greedy algorithmic framework to devise the expansion numerically. Finally, we show that the practical uses of the DCD expansion can be complemented by the proposed framework and perform two experiments with a view towards applications.

Keywords: Circulant matrix, Diagonal matrix, Generalized diagonal matrix, Toeplitz matrix, Matrix factorizations, Matrix expansions, SVD, PCA, FFT, JL-Transforms.

TABLE OF CONTENTS

FOREWORD

About one year ago, I walked a few steps, but at a high scientific amplitude, with Marko while discussing some research ideas. Apparently, he proceeded to show me novel and generalized versions of the ideas that I had been thinking about. Since then, I have benefited to work on very fascinating research topic under the guidance of Marko. Therefore, I would like to first thank my supervisor Prof. Marko Huhtanen for his endless support, understanding, and motivation throughout this research. His constant encouragement and personal example have been instrumental in developing my Master's thesis.

I am also truly thankful to Prof. Tapio Seppänen for his advice and great flexibility. He was always very enthusiastic during our research work and I am really greatful for his understandings me at difficult times. Further, I would like to thank Tapio again for giving me a great opportunity for being part of a leading research group CMVS.

I am also grateful for numerous helpful discussions with Kai and Nisansala. I learned a huge amount from working with you. Last but not least, I would like to thank my parents and my brother for the immense support and love they have provided me during my studies.

Oulu, 26th May, 2020
Charmin Asirimath

University Supervisor & Technical Supervisor: Prof. Marko Huhtanen
Second Examiner: Prof. Tapio Seppänen

LIST OF ABBREVIATIONS AND SYMBOLS

$\mathbf{1}$	Vector with all components one.
$\mathbf{I}$	Identity matrix.
$\mathbf{D}$	Diagonal matrix.
$\mathbf{C}$	Circulant matrix.
$\mathbf{T}$	Toeplitz matrix.
$\mathbf{F}$	Fourier matrix.
$\mathbf{X}^T$	Transpose of matrix $\mathbf{X}$.
$\mathbf{X}^*$	Hermitian (complex conjugate) transpose of matrix $\mathbf{X}$.
$\mathbf{X}^{-1}$	Inverse of matrix $\mathbf{X}$.
$\mathbf{Diag}(\mathbf{X})$	Diagonal of matrix $\mathbf{X}$.
$\mathbf{Tr}(\mathbf{X})$	Trace of matrix $\mathbf{X}$.
$x_1 * x_2$	Element-wise multiplication of vectors x_1 and x_2.
$\overline{x}$	Complex conjugate of x.
$\mathbb{N}$	Natural numbers.
$\mathbb{R}$	Real numbers.
$\mathbb{R}^n$	Real n-vectors.
$\mathbb{R}^{n\times m}$	Real $n \times m$ matrices.
$\mathbb{C}$	Complex numbers.
$\mathbb{C}^n$	Complex n-vectors.
$\mathbb{C}^{n\times m}$	Complex $n \times m$ matrices.
$\mathcal{C}$	Set of all circulant matrices.
$\mathcal{D}$	Set of all diagonal matrices.
σ_i	i^{th} singular value of a matrix.
$\|x\|_2$	Euclidean norm of x.
$\|\mathbf{X}\|_F$	Frobenius norm of matrix $\mathbf{X}$.
$x < y$	Strict componentwise inequality between vectors x and y.
$x \leq y$	Componentwise inequality between vectors x and y.
$arg(\cdot)$	argument.
$f_1 \circ f_2$	Composition of functions f_1 and f_2.
$\mathbf{dom}(f)$	Domain of function f.
∇f	Gradient of function f.
$\mathbf{J}$	Jacobian matrix of a function.

1. INTRODUCTION

With the proliferation of advanced data acquisition techniques in recent years, *data sets* have grown exponentially in size. It has been a motive for designing novel algorithmic paradigms for approximating such data, particularly in the area of signal processing, electrical circuit design, computer vision, and many others. One example is *compressed sensing* (CS) which exploits sparsity patterns of the data with specialized methods. At the heart of CS, one computes the probability of intersecting a convex cone K with the apex at origin and a randomly sampled m-dimensional linear subsapce in a n-dimensional Euclidean space with $m \ll n$ [1]. For a low dimensional configuration, see Figure 1. This geometric problem leads to designing new kind of algorithms in data compression with mathematical promises on fast computations under certain conditions [1].

However, entries of higher dimensional data structures (e.g., matrices and tensors) may be equipped with a complexity reduction of some sort beyond sparsity patterns (e.g., circulant and symmetric). To this end, one can consider factoring the original matrix into structured factors (e.g., decomposing a super resolution image into rank-one components) such that the original matrix can be parameterized by few parameters. An advantage of such a factorization is that once computed, these factors can be reused to solve new problems. Specifically, these factors often can be updated with respect to small changes in the original data.

In contrast to factoring, another approach to deal with data is approximating a large matrix with an ***expansion of structured elements***. This becomes a cornerstone problem in a huge variety of data driven applications and mathematics. A structure of the elements is generally imposed as a constraint to each factor of the expansion such as sparsity patterns, non-negativeness, rank constraint, and etc. One of the main applications of structured expansion is that we can exploit structural properties of the elements to reduce memory foot print of various algorithms. Another application is that structured expansions enable us to efficiently represent real world data in a succinct and parsimonious manner. For example, the SVD can be written as an expansion of rank-one matrices, see Corollary 13, such that each factor or rank-one matrix can be stored as a 2-dimensonal array.

Nevertheless, handling and analysing structured factorizations and expansions of higher dimensional dense data may cost a lot of computational resources and storage. For example, structured matrix approximation problems are in general

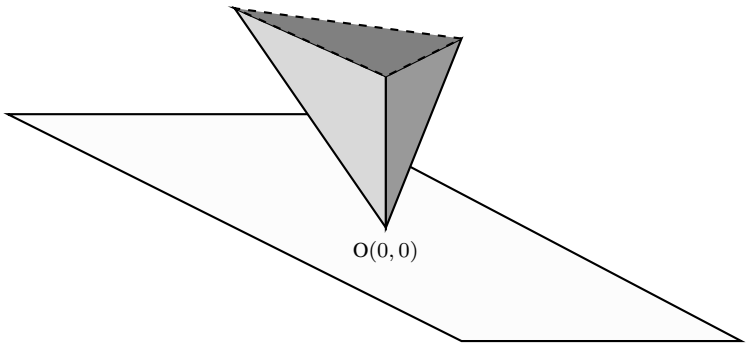

Figure 1. Intersection of a convex cone and a linear subspace.

NP hard problems; for rank constraint problems see [2]. Thus devising efficient methods to approximate data is of paramount importance and has opened challenging mathematical problems.

In summary, state of the art large matrix factorizations and approximations are far from being developed to meet modern data processing requirements. Hence, a line of research is going on understanding and developing algorithmic matrix factorizations and expansions with a view towards pushing the boundaries of state of the art. To this end, this thesis is focused on exploring computational structured matrix expansions, factorization, and their applications. A fundamental challenge is approximating a given matrix $\mathbf{A}$ with a structured matrix which has inherently proper computational properties. Among many, there has been a lot of attention on approximating a matrix using products of circulant and diagonal matrices. Thus, we describe methods studied in this thesis based on using diagonal and circulant matrices for factoring, expanding, and approximating matrices. Moreover, we analyze such factorizations and approximations towards developing novel algorithms for applications.

Objectives:

This thesis is concerned with the problem of exploring algorithms with sums of $\mathbf{DCD}$ matrices to approximate $\mathbf{A} \in \mathbb{C}^{n\times n}$, i.e.,

$$\min_{\mathbf{D}_i,\mathbf{D}_j\in\mathcal{D},\mathbf{C}_k\in\mathcal{C}} \left\| \mathbf{A} - \sum_{i,j,k}^{r} \mathbf{D}_i\mathbf{C}_k\mathbf{D}_j \right\|_F , \tag{1}$$

where $\mathcal{D}$ and $\mathcal{C}$ denote diagonal and circulant matrix subspaces respectively.

To this end, we construct $\sum_{i,j,k}^{r} \mathbf{D}_i\mathbf{C}_k\mathbf{D}_j$ iteratively by fitting an approximately optimal $\mathbf{DCD}$ matrix for residue which is obtained from the previous approximations. Therefore, our main objective is developing algorithms to solve following optimization problem for a given $\hat{\mathbf{A}} \in \mathbb{C}^{n\times n}$,

$$\min_{\mathbf{D}_1,\mathbf{D}_2\in\mathcal{D},\mathbf{C}_1\in\mathcal{C}} \left\| \hat{\mathbf{A}} - \mathbf{D}_1\mathbf{C}_1\mathbf{D}_2 \right\|_F .$$

Efficient computation is central on developing a new kind of algorithms. Greedy algorithms and alternate iteration can be potential directions. In addition, we introduce a notion of a matrix rank, denoted as $\mathbf{DCD}$-rank, which is closely related to the expansion and we compute the rank of some specific structured matrices.

Why Approximation with Diagonal and Circulant Matrices?

Approximating with diagonal and circulant matrices has lots of advantages:

- Sum of $\mathbf{DCD}$ matrices extends rank-k matrices in a natural, although different way, see for Example 8. One of the main difference comes from the fact that these matrices, already for small k, are typically nonsingular.
- The $\mathbf{DCD}$-rank of Toeplitz matrix is already two. So both the $\mathbf{DCD}$-rank one and two are very interesting, since the $\mathbf{DCD}$-rank one is an extension of rank-one matrices, see Section 4.0.2.

- $\mathbf{DCD}$ matrices are fundamental blocks in multiplicative matrix approximation, see Section 4.0.2.
- For small values of the $\mathbf{DCD}$-rank, $\mathbf{DCD}$ approximation gives an important dense structure for which matrix-vector products are cheap.
- $\mathbf{DCD}$ approximation enables to perform Fast Fourier Transform (FFT) based fast computations in certain dimensions, see Section 4.0.1.

Some Applications:

- **Linear dimension reduction** is a core step in many machine learning and data analysis applications. A classical example is principle component analysis, see Section 3 for more details. Note that linear dimensional reduction techniques can be modeled as a matrix $\mathbf{A} \in \mathbb{R}^{m\times n}$ with $m < n$, where matrix-vector product costs $\mathcal{O}(mn)$ operations. An interesting question is that how well can circulant like structured matrices approximate matrix $\mathbf{A}$? In particular, approximating $\mathbf{A}$ with partially circulant matrices[1] becomes more interesting, since these approximations allow us to enjoy low time and storage complexities. For example, authors in [3] proposed an algorithm based on partially circulant matrices for approximating a given *fat* matrix $\mathbf{A}$, and demonstrated the usage of the approximant in real world data.
- **Random projection** has recently emerged as a technique that projects very high dimensional data onto a lower dimensional subspace using a random matrix. To this end, $\mathbf{CD}$ matrices play a critical role in random circular projections over unstructured randomized projections. For example, linear mapping $f : \mathbb{R}^n \to \mathbb{R}^m$ defined in Johnson-Lindenstrauss lemma can be reformulated as a product of circulant and diagonal matrices, i.e., $f(x) = \mathbf{MD}x$, where $\mathbf{M}$ is a partially circulant matrix and $\mathbf{D}$ is a diagonal matrix with independent Bernoulli variables [4]. Computationally, this representation allows to accelerate Fast Johnson-Lindenstrauss Transforms. For more information, see Section 4.0.3 and references therein.
- **Compression of deep learning frameworks** has become very important, since deep learning, a statistical model with millions of parameters, is a paradigm shift in learning algorithms for complex data. For example, *video classification* is a challenging task for many machine learning algorithms. However, researchers have proposed different deep learning architectures by arranging layers and operations among them intelligently (e.g., recurrent networks and 3D convolutional networks) to gain high classification accuracy. A bottleneck is that these systems are computationally more intensive. To circumvent aforementioned issue, authors in [5] proposed a compact neural network model to replace the two fully connected layers of *AlexNet* by circulant and diagonal matrices, where circulant is learned by optimization algorithms and diagonal matrix is sampled at random $\{1, -1\}$. They reported that complexity of the

[1] We define a partial circulant matrix $\mathbf{M} \in \mathbb{C}^{m\times n}$ by $\mathbf{M} = \mathbf{SC}$, where $\mathbf{S}$ is a randomly selected m rows of $\mathbf{I} \in \mathbb{C}^{n\times n}$ and $\mathbf{C} \in \mathbb{C}^{n\times n}$ is a circulant matrix.

proposed model can be reduced by a factor of 10, while maintaining competitive error rates.

Thesis Outline:

In Section 2, we will give an introduction to the rich area of factorings and expansions of matrices. In Section 3, we will review the properties of the SVD and study an algorithm that computes the SVD numerically. After all, the SVD is an ideal prototype of simultaneous factoring and expansion. In Section 4, we will first present some classical properties of the circulant and diagonal matrices and then focus on matrix factorizations, approximations, and expansions consisting products of circulant and diagonal matrices. Moreover, we will introduce a new notion of a matrix rank associated with $\mathbf{DCD}$ expansion. In Section 5, we will propose a greedy framework for computing sums of $\mathbf{DCD}$ matrices. In Section 6, we will evaluate and illustrate the performance of the framework numerically. We will compare our simulation results with the SVD. Section 7 concludes the thesis, presenting some future research directions.

2. A TOUR OF MATRIX FACTORIZATIONS AND EXPANSIONS

In this section, we give a rudimentary introduction to state of the art matrix factorizations, expansions, and approximation techniques. In particular, we study how the different kind of mathematics reveal the beauty of these techniques complemented by modern applications. We start our discussion by outlining some classical matrix decompositions in numerical linear algebra. The importance of these decompositions for engineering and computer science can hardly be overestimated, since these factorizations are extensively used in a wide range of mathematical subroutines.

However, often times in probability and statistics, the original matrix as well as its factors turn out to have some specific positivity structures. Roughly speaking, such a decomposition is called a nonnegative factorization and it decomposes a original nonnegative matrix into nonnegative factors. One of the most appealing characteristics of this decomposition is the geometrical interpretations based on nested polyhedra. This is our second topic and it finds many applications (e.g., statistical mixture models, matrix completion, and etc).

Next, we turn our attention to structured matrix decompositions that are allowed to decompose a complex matrix into a product of structured factors. Therefore, it is more crucial than nonnegative matrix factorization, not only because it provides more hidden information about our underlying experiments, but also because it may allow us to solve large systems more efficiently. To this end, we will first revisit a classical result in linear algebra: any square matrix is a product of two complex symmetric matrices. With these motivations, we will next study a modern factorization: decomposing a generic square matrix into a product of Toeplitz matrices. However, one of the obstacles, apparently an open research problem, in the field is that developing algorithms to realize these factorizations efficiently.

The fairly recent growth of structured matrix expansions for approximating large matrices probably masks the hardness of structured matrix factorizations. This is fueled by recent advancement of theories and sophisticated algorithms. To this end, we will briefly discuss two matrix expansions appearing in statistics: matrix Gaussian series and matrix Rademacher series. Specifically, we will notice that Gaussian Toeplitz matrix can be equivalently written as an expansion of structured Gaussian matrices.

2.0.1. Classical Matrix Factorizations

Let's start our journey on matrix factorization by recalling some classical matrix decompositions in numerical linear algebra. These decompositions revolutionized computer science, matrix analysis, and applications. Specifically, these matrix decompositions are extremely useful, stable to compute, and one can perform backward rounding error analysis.

- Cholesky decomposition: Given a positive definite matrix $\mathbf{A}$, there is a unique upper triangular matrix $\mathbf{R}$ whose diagonal has positive elements such that $\mathbf{A} = \mathbf{R}^*\mathbf{R}$. Equivalently, one can decompose the $\mathbf{A}$ as following: $\mathbf{A} = \mathbf{L}\mathbf{D}\mathbf{L}^*$,

where $\mathbf{D}$ and $\mathbf{L}$ are diagonal and lower triangular matrices, respectively. State of the art algorithms require approximately $\mathcal{O}(\frac{n^3}{6})$ operations to compute this factorization.

- $\mathbf{QR}$ decomposition: If $\mathbf{A}$ is a complex $m \times n$ matrix with $n \leq m$, then there is a unitary matrix such that $\mathbf{Q}^*\mathbf{A} = [\mathbf{R}^T \, \mathbf{0}^T]^T$. There are two classes of algorithms to compute this decomposition: Modified Gram–Schmidt algorithm and orthogonal triangularization. Both of these algorithms require approximately $\mathcal{O}(mn^2)$ operations when $m \lll n$.

- Spectral decomposition: If $\mathbf{A}$ is a Hermitian matrix, then $\mathbf{A} = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^*$, where $\mathbf{Q}$ is a unitary matrix and $\mathbf{\Lambda}$ is a diagonal matrix with eigenvalues of $\mathbf{A}$. There are some algorithms based on divide and conquer strategy or Jacobi's algorithm to compute the decomposition. Generally, these algorithms require $\mathcal{O}(n^3)$ operations.

- Schur decomposition: If $\mathbf{A}$ is a $n \times n$ square matrix, then there is a unitary matrix $\mathbf{Q}$ such that $\mathbf{A} = \mathbf{Q}\mathbf{T}\mathbf{Q}^*$. In this decomposition, $\mathbf{T}$ is an upper triangular matrix.

- Singular value decomposition (SVD): SVD has been considered as the most important and useful matrix factorization for two reasons: (a). Sophisticated algorithms have been developed to compute the decomposition numerically stable and reliable ways. (b) The SVD has many appealing properties towards numerous applications. For example, the SVD geometrically consists an elegant rank-one matrix expansion and it turns out that those rank-one factors are orthogonal. Hence, the SVD lies at the core of the modern algorithmic toolbox and we will discuss about the SVD in more details in our next chapter.

2.0.2. Nonnegative Matrix Factorization for Real Matrices

It is a topic of major interest to factorize real matrices, since the prevalence of matrix data coming from applications are real and nonnegative. It is arguably more crucial to compute variations of nonnegative factorizations, since such factorizations may capture seemly hidden information about underlying experiments. To this end, we are going to revisit a fundamental notion in linear algebra: *rank* of a matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$. Even though rank is defined as the dimension of the span of the rows or span of the columns usually, one can also define it as minimal r that admits factoring $\mathbf{A} = \mathbf{X}\mathbf{Y}$ with $\mathbf{X} \in \mathbb{R}^{m \times r}$ and $\mathbf{Y} \in \mathbb{R}^{r \times n}$. Equivalently, rank of $\mathbf{A}$ can be defined as the minimal r for which there exist vectors $x_1, \ldots, x_m \in \mathbb{R}^r$ (corresponding to rows of $\mathbf{X}$) and $y_1, \ldots, y_n \in \mathbb{R}^r$ (corresponding to columns of $\mathbf{Y}$) such that $\mathbf{A}(i,j) = x_i^T y_j$, for all $1 \leq i \leq m, 1 \leq j \leq n$. Interestingly, this factorization can be explained geometrically using a pair of nested polytopes when $\mathbf{A}$ has nonnegative entries: outer polytope is defined by inequalities $\mathbf{X}x \geq 0$ whereas inner polytope is defined by a convex hull generated by columns of $\mathbf{Y}$, see Figure 2.

To see more details, we adapt some definitions from convex geometry: By definition, convex hull of a set $\{v_1, \ldots, v_m\}$ is $conv(v_1, \ldots, v_m) = \left\{ \sum_{i=1}^m \theta_i v_i \, : \, \theta_i \geq 0, \sum_{i=1}^m \theta_i = 1 \right\}$ and a polytope is a convex hull of a finite set in $\mathbb{R}^r$. In this

thesis, a polyhedron is a finite intersection of closed halfspaces, where a closed halfspace is a set of the form $\{x \in \mathbb{R}^r : v^T x \leq z, v \in \mathbb{R}^r, z \in \mathbb{R}\}$. Suppose that $P = conv(v_1, \ldots, v_m) \subset \mathbb{R}^{r-1}$ is a polytope with vertex description and $Q = \{x \in \mathbb{R}^{r-1} : h_i^T x \leq z_i, i = 1, \ldots, n\} \subset \mathbb{R}^{r-1}$ is a polyhedron for some $h_i, \ldots, h_n \in \mathbb{R}^{r-1}$ and some $z_i, \ldots, z_n \in \mathbb{R}$. If Q is bounded, then it is also a polytope. Slack matrix of the pair (P, Q), denoted by $\mathbf{S}_{P,Q}$, is a $m \times n$ matrix whose (i, j)-th entry defined by $z_i - h_i^T v_j$. Finally, we denote $\mathbf{1}$ as a vector with all elements one.

Then the following lemma shows that one can compute bounded nested polytopes P and Q for a given nonnegative matrix, i.e., $P \subseteq Q \subseteq \mathbb{R}^{r-1}$. We will outline its proof and refer to [6] for its ramifications.

Lemma 1. *Let $\mathbf{A} \in \mathbb{R}^{m \times n}$ be a nonnegative matrix and assume that $\mathbf{A1} = \mathbf{1}$. Let rank of $\mathbf{A}$ is r. Then, there exist the polytopes $P, Q \subseteq \mathbb{R}^{r-1}$ such that $P \subseteq Q$ and $\mathbf{A}$ is a slack matrix of the pair (P, Q).*

Sketch of proof: Choose r linearly independent set of rows of $\mathbf{A}$ and define $\mathbf{Y} \in \mathbb{R}^{r \times q}$ as a submatrix of $\mathbf{A}$ using these rows. Note that $\mathbf{Y}$ is a full rank matrix. Thus, we can find a unique $\mathbf{X}$ such that $\mathbf{A} = \mathbf{XY}$. Further, we have $\mathbf{1} = \mathbf{A1} = \mathbf{XY1} = \mathbf{X1}$, since the set of the rows of $\mathbf{Y}$ is a linearly independent maximal subset of $\mathbf{A}$ and $\mathbf{A1} = \mathbf{1}$, then $\mathbf{Y1} = \mathbf{1}$. Therefore, we conclude that we can compute the factorization $\mathbf{A} = \mathbf{XY}$ such that $\mathbf{X1} = \mathbf{1}$ and $\mathbf{Y1} = \mathbf{1}$.

With these properties write $\mathbf{X}$ and $\mathbf{Y}$ as follows:

$$\mathbf{X} = \begin{bmatrix} x_1^T & t_1 \\ x_2^T & t_2 \\ \vdots & \vdots \\ x_m^T & t_m \end{bmatrix} \text{ and } \mathbf{Y} = \begin{bmatrix} y_1^T \\ y_2^T \\ \vdots \\ y_r^T \end{bmatrix},$$

where each $x_i \in \mathbb{R}^{r-1}$, $t_i \in \mathbb{R}$, and $y_i \in \mathbb{R}^n$. Note that $\mathbf{X1} = \mathbf{1} \iff t_i = 1 - \mathbf{1}^T x_i$. Define a polytope P and a polyhedron Q as follows:

$$P = conv(x_1, \cdots, x_m) \subset \mathbb{R}^{r-1}$$

and

$$Q = \left\{ z \in \mathbb{R}^{r-1} : (z^T, 1 - \mathbf{1}^T z)\mathbf{Y} \geq 0 \right\}.$$

Consider, any $x = \alpha_1 x_1 + \cdots + \alpha_m x_m \in P$. Then $(x^T, 1 - \mathbf{1}^T x)\mathbf{Y} \geq 0$, since $\mathbf{XY} \geq 0$ and each $\alpha_i \geq 0$. It implies that $P \subseteq Q$. In addition, one can easily check that $\mathbf{A}$ is a slack matrix of the pair (P, Q). Next, we need to show that Q is bounded. Let's prove it using a contradiction: suppose Q is unbounded, then there exists a vector $z_0 + \alpha z \in Q$ for all $\alpha \geq 0$ with $z_0 \in Q$. It follows that $(z^T, -\mathbf{1}^T z)\mathbf{Y} \geq 0$ and $((z^T, -\mathbf{1}^T z)\mathbf{Y})^T \mathbf{1} = 0 \implies (z^T, -\mathbf{1}^T z)\mathbf{Y} = 0$. This means that $z = 0$, since $\mathbf{Y}$ is a full rank matrix. Therefore, Q is bounded. One can thus compute the bounded nested polytopes P and Q for a given nonengative matrix.

Oftentimes in applications, we impose the constrains on $\mathbf{X}$ and $\mathbf{Y}$ such that both have nonnegative coefficients, i.e., $\mathbf{X}(i, j) \geq 0$ and $\mathbf{Y}(i, j) \geq 0$ for all i and j. We call such a factorization as nonnegative matrix factorization. Properties of the factorization can be characterised by *nonnegative rank* which is defined as minimum number of

nonnegative rank-one matrices needed additively to reconstruct the original exactly. On the negative side, computing the nonnegative rank and the corresponding rank-one factors are computationally hard problems [6]. Geometrically, determining the rank is related to find a polytope with minimum number of vertices nested between given two polytopes. For more information refer [6]. To see more details, we outline the following lemma for completeness.

Lemma 2. *Let* $\mathbf{A} \in \mathbb{R}^{m \times n}$ *be a nonnegative matrix with rank* r *and* $\mathbf{A}\mathbf{1} = \mathbf{1}$*. Then* $P \subseteq Q \subset \mathbb{R}^{r-1}$ *can be obtained from Lemma 1. Then* $\mathbf{A}$ *has a size* q *nonnegative factorization if and only if there exists a polytope* S *with* q*-vertices such that* $P \subseteq S \subseteq Q$*.*

Proof: See [7].

Example 1. *See Figure 2.*

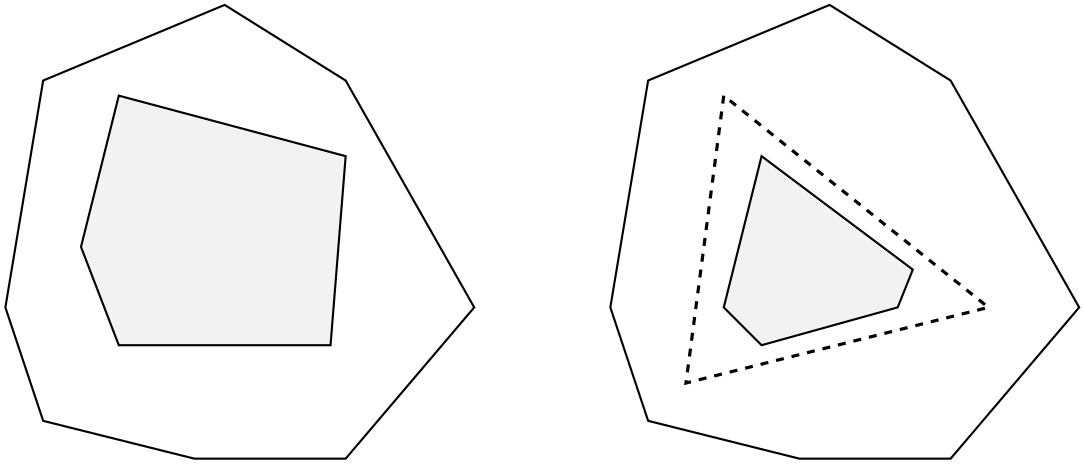

Figure 2. Left: Nested polytopes P and Q of a nonnegative matrix $\mathbf{A}_1$ whose nonnegative factorization is greater than three, since we could not compute a triangle between them. Right: Nested polytopes P and Q of a nonnegative matrix $\mathbf{A}_2$ whose nonnegative factorization is three.

In addition to theoretically pleasing properties, nonnegative matrix factorization has a lot of applications in computer science (e.g., computational geometry), statistics (e.g., Netflix prize problem), bioinformatics (e.g., super resolution imaging), and etc. Moreover, approximately computing nonnegative factorization has direct applications in numerous fields, including statistical machine learning and image processing. For example, suppose that there is a nonnegative matrix $\mathbf{A} \in \mathbb{R}^{n \times m}$ consisting m vectorized face images, i.e., each column of $\mathbf{A}$ represents a face image. Computing or learning nonnegative $\mathbf{Y} \in \mathbb{R}^{r \times m}$ such that $\mathbf{A} \approx \mathbf{W}\mathbf{Y}$ with $\mathbf{W} \in \mathbb{R}^{n \times r}$ and $r \leq \min(m, n)$ can be understood as learning the representations of the original faces in a low dimensional space: A column vector of $\mathbf{Y}$ can be used as a low dimensional representation of a original face in a space spanned by $\mathbf{W}$. For more information, see [8].

2.0.3. Some Non-Standard Matrix Decompositions

A natural question arising in the numerical solution of practical problems is that can we factorize a given matrix $\mathbf{A} \in \mathbb{C}^{n\times n}$ into a product of structured matrices such that each factor has a structure beyond the non-negativeness of its entries? In other words, we are interested on factoring a complex valued matrix $\mathbf{A}$ such that $\mathbf{A} = \mathbf{S}_1 \cdots \mathbf{S}_k$, where each factor $\mathbf{S}_i$ has complex valued entries with some pattern in its structure. To this end, one of the interesting simple structures is symmetricity. Therefore, we will first study a classical result in linear algebra: the factorization of a square matrix into a product of two complex symmetric matrices. To prove this, we heavily rely on Jordan normal form of a matrix and its related theorems. For more information, see [9].

Definition 3. *Jordan block* $\mathbf{J}(\lambda)_k$ *is a* $k \times k$ *upper triangular matrix of the form following:*

$$\mathbf{J}(\lambda)_k = \begin{bmatrix} \lambda & 1 & \cdots & 0 \\ 0 & \lambda & \ddots & \vdots \\ \vdots & & \ddots & 1 \\ 0 & \cdots & \cdots & \lambda \end{bmatrix}.$$

Definition 4. *Jordan matrix* $\mathbf{J}$ *is a* $n \times n$ *block diagonal matrix of the form following:*

$$\mathbf{J} = \begin{bmatrix} \mathbf{J}(\lambda_1)_{k_1} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{J}(\lambda_2)_{k_2} & \ddots & \vdots \\ \vdots & & \ddots & \mathbf{0} \\ \mathbf{0} & \cdots & \cdots & \mathbf{J}(\lambda_r)_{k_r} \end{bmatrix}, \text{ where } k_1 + k_2 \cdots + k_r = n.$$

Lemma 5. *Every Jordan matrix is a product of a real symmetric matrix and a complex symmetric matrix.*

Sketch of Proof: First, note that one can have $\mathbf{J}(\lambda)_k = \mathbf{S}_k\mathbf{H}_k$, where

$$\mathbf{S}_k = \begin{bmatrix} 0 & 0 & \cdots & 0 & 1 \\ 0 & \cdots & 0 & 1 & 0 \\ \vdots & & & & \vdots \\ 1 & 0 & \cdots & \cdots & 0 \end{bmatrix} \text{ and } \mathbf{H}_k = \begin{bmatrix} 0 & 0 & \cdots & 0 & \lambda \\ 0 & \cdots & 0 & \lambda & 1 \\ \vdots & & & & \vdots \\ \lambda & 1 & \cdots & \cdots & 0 \end{bmatrix}.$$

It follows that $\mathbf{J}(\lambda)_k$ can be factorized as a product of a real symmetric matrix $\mathbf{S}_k$ and a complex symmetric matrix $\mathbf{H}_k$. Next set $\mathbf{S}$ (a real symmetric matrix) and $\mathbf{H}$ (a complex symmetric matrix) such that they are block diagonal matrices consisting $\mathbf{S}_k$ and $\mathbf{H}_k$ matrices for each λ_k, respectively. Then one can compute product $\mathbf{SH} = \mathbf{J}$.

Lemma 6. *Suppose that* $\mathbf{A}$ *is a complex square matrix. Then* $\mathbf{A} = \mathbf{BJB}^{-1}$ *for some Jordan matrix* $\mathbf{J}$ *and some nonsingular matrix* $\mathbf{B}$.

Proof: See [10].

Theorem 7. *Every square complex matrix* $\mathbf{A}$ *is a product of two complex symmetric matrices.*

Sketch of Proof: We can obtain following factorization for $\mathbf{A}$ using Lemma 5 and Lemma 6: $\mathbf{A} = \mathbf{BJB}^{-1}$ with $\mathbf{J} = \mathbf{SH}$, where $\mathbf{S}$ is a real symmetric matrix and $\mathbf{H}$ is a complex symmetric matrix. Thus, $\mathbf{A} = (\mathbf{BSB}^T)(\mathbf{B}^{-T}\mathbf{HB}^{-1})$. Note that these factors are complex symmetric matrices.

Next, we will spend some time on studying a factorization of a generic square matrix into a product of Toeplitz matrices, i.e., $\mathbf{A} = \mathbf{T}_1 \cdots \mathbf{T}_r$, where each $\mathbf{T}_i$ is a Toeplitz matrix. Main motivation behind this study is that these factors, Toeplitz matrices, have pretty computational characteristics for developing numerically stable superfast algorithms. We mean *superfast* here is that these algorithms require less than $\mathcal{O}(n^2)$ operations to solve Toeplitz like systems $\mathbf{T}x = b$. To this end, many superfast stable Toeplitz solvers use trigonometric transformations (e.g., Fast Fourier Transforms and Discrete Cosine Transforms) to covert original Toeplitz matrix into much simpler factors which can be done in $\mathcal{O}(nlog(n))$ operations [11]. Therefore, if $\mathbf{A}$ has a known Toeplitz decomposition with r-Toeplitz factors, one can solve $\mathbf{A}x = \mathbf{T}_1 \cdots \mathbf{T}_r x = b$ recursively using $\mathcal{O}(rnlog(n))$ operations.

We begin our studies about the decomposition by recalling the definition of Toeplitz matrices: a Toeplitz matrix $\mathbf{T} \in \mathbb{C}^{n\times n}$ is a square matrix whose entries along its diagonals are constants, i.e.,

$$\mathbf{T} = \begin{bmatrix} t_0 & t_1 & & \cdots & t_n \\ t_{-1} & t_0 & t_1 & & \\ t_{-2} & t_{-1} & t_0 & \ddots & \\ & \ddots & \ddots & \ddots & t_1 \\ t_{-n} & & t_{-2} & t_{-1} & t_0 \end{bmatrix}.$$

Remarkably, set of all Toeplitz matrices form a linear subspace over $\mathbb{C}^{n\times n}$ with basis $\mathbf{B}_k = \{\delta_{i,j+k}\}_{i,j=1}^n$ for $-n+1 \leq k \leq n-1$. For instance,

$$\mathbf{B}_1 = \begin{bmatrix} 0 & 1 & & \cdots & 0 \\ 0 & 0 & 1 & & \\ 0 & 0 & 0 & \ddots & \\ & \ddots & \ddots & \ddots & 1 \\ 0 & & 0 & 0 & 0 \end{bmatrix}, \quad \mathbf{B}_{-1} = \begin{bmatrix} 0 & 0 & & \cdots & 0 \\ 1 & 0 & 0 & & \\ 0 & 1 & 0 & \ddots & \\ & \ddots & \ddots & \ddots & 0 \\ 0 & & 0 & 1 & 0 \end{bmatrix},$$

and $\mathbf{B}_0 = \mathbf{I}$. Thus, $\mathbf{T} = \sum_{i=-n+1}^{n-1} t_i \mathbf{B}_i$.

To explore the decomposition in more detail, we will utilize some basic algebraic geometric tools: A monomial in $x_1, \ldots, x_n$ is a product of a form $x_1^{\gamma_1} \ldots x_n^{\gamma_n}$, where all exponents $\gamma_1, \ldots, \gamma_n$ are nonnegative integers. A polynomial f_0 in $x_1, \ldots, x_n$ with coefficients in $\mathbb{C}$, denoted as $f_0 \in \mathbb{C}(x_1, \ldots, x_n)$, is a linear combination of monomials. Algebraic variety of a set of polynomials $f_1, \ldots, f_r \in \mathbb{C}(x_1, \ldots, x_n)$ is a set $\{a \in \mathbb{C}^n : f_i(a) = 0, i = 1, \ldots, r\}$. Moreover, a subvariety is a subset of a variety which itself is a variety. For instance, set of all $n \times n$ Toeplitz matrices forms a subvariety in $\mathbb{C}^{n\times n}$ which is defined by a set of linear equations. Specifically, we denote r-tuple of $n \times n$ Toeplitz matrices over $\mathbb{C}$ as $\mathcal{T}_n^r(\mathbb{C})$ and $\mathcal{T}_n^r(\mathbb{C})$ is a variety in $\mathbb{C}^{rn^2}$. A map $f : X \to Y$ defined on algebraic varieties X and Y is a dominant, if its image is dense. To prove the basis of Toeplitz decompositions, the authors in [12] focused on following Lemmas.

Lemma 8. *Let $f : X \to Y$ be a map between algebraic varieties. If f is dominant, then $f(X)$ contains an open dense subset of Y.*

Proof: See [12].

Lemma 9. *Suppose $f : X \to Y$ is a map between algebraic varieties X and Y with $dim(X) \geq dim(Y)$. If there is a point $x \in X$ such that the differential of f at x is full rank, then the differential at generic point will also have the full rank.*

Proof: See [12].

Moreover, we adapt the following definitions: $x \in X$ is a point of closure of X, if every neighbourhood of x contains at least a point in X. A point $x \in X$ is called a generic point whose closure is whole set X.

Example 2. *Suppose L is a vector subspace in $\mathbb{C}^n$. A generic point $x \in \mathbb{C}^n$ is not contained in L.*

Putting all factors together, authors in [12] prove following theorem. We will sketch their proof by outlining basic steps explicitly and we will adapt the following abbreviation "x's" to denote a variable x with multiple entries.

Theorem 10. *Let $f : \mathcal{T}_n^r(\mathbb{C}) \to \mathbb{C}^{n\times n}$ be the map defined by $f(\mathbf{T}_{n-r}, \ldots, \mathbf{T}_{n-1}) = \mathbf{T}_{n-r} \cdots \mathbf{T}_{n-1}$. Then, for a generic point $x \in$ with $r \geq \lfloor n/2 \rfloor + 1$, the differential of f at x is full rank. Thus, for a generic $\mathbf{A} \in \mathbb{C}^{n\times n}$, there exist $r = \lfloor n/2 \rfloor + 1$ Toeplitz matrices $\mathbf{T}_1 \ldots \mathbf{T}_r$ such that $\mathbf{A} = \mathbf{T}_1 \ldots \mathbf{T}_r$.*

Sketch of proof: Let $r = \lfloor n/2 \rfloor + 1$ and set $y = (\mathbf{Y}_{n-r}, \ldots, \mathbf{Y}_{n-1}) \in \mathcal{T}_n^r(\mathbb{C})$ with $\mathbf{Y}_{n-i} = \sum_{j=-n+1}^{n-1} y_{n-i,j}\mathbf{B}_j$ for $i = 1, \ldots, r$. Note that differential of f at $x = (\mathbf{T}_{n-r}, \ldots, \mathbf{T}_{n-1}) \in \mathcal{T}_n^r(\mathbb{C})$, denoted as $\mathbf{D}f_x$, is a $n \times n$ matrix with the following form:

$$\mathbf{D}f_x(y) = \sum_{i=1}^{r} \mathbf{T}_{n-r} \cdots \mathbf{T}_{n-i-1}\mathbf{Y}_{n-i}\mathbf{T}_{n-i+1} \cdots \mathbf{T}_{n-1}.$$

Thus $(p, q)^{th}$ entry of this matrix, denoted as $l_{p,q}$, is a linear form of y's. However, $\mathbf{D}f_x$ can be also modeled as a $n^2 \times (2n-1)r$ coefficient matrix $\mathbf{M}$, since $\mathbf{D}f_x$ is a linear map from $\mathcal{T}_n^r(\mathbb{C})$ with dimension $(2n-1)r$ to $\mathbb{C}^{n\times n}$ with dimension n^2. In other words, one can devise $\mathbf{M}$ as follows: arrange rows of $\mathbf{M}$ corresponding to $l_{p,q}$'s and columns of $\mathbf{M}$ corresponding to the y's. Our purpose of following discussion is to find a point $x \in \mathcal{T}_n^r(\mathbb{C})$ such that $\mathbf{M}$ is a full rank matrix, i.e., $\mathbf{M}$ has a nonzero $n^2 \times n^2$ minor. Recall that $n^2 \times n^2$ minor of $\mathbf{M}$ is the determinant of a submatrix obtained from selecting n^2 columns of $\mathbf{M}$.

To this end, the authors in [12] have suggested to select a specific point $x^* = (\mathbf{T}_{n-r}, \ldots, \mathbf{T}_{n-1}) \in \mathcal{T}_n^r(\mathbb{C})$ such that

$$\mathbf{T}_{n-i} = \mathbf{B}_0 + t_{n-i}(\mathbf{B}_{n-i} - \mathbf{B}_{-n+i})$$

for $i = 1, \ldots, r$.

For this setting, $l_{p,q}$ is a linear form in y's and its coefficients are polynomial in t's. In particular, we have following form for $l_{p,q}$:

$$l_{p,q} = \sum_{i=1}^{r} y_{i,q-p} + \mathcal{O}(t).$$

Therefore, entries of $\mathbf{M}$ are polynomials in t's and constant terms are 1's. Thus, any $n^2 \times n^2$ minor of $\mathbf{M}$ is a polynomial in t's. By carefully exploiting the structure of $\mathbf{M}$, one can observe that any $n^2 \times n^2$ minor of $\mathbf{M}$ is a polynomial in t's of degree at least $(n-1)^2$. We demonstrate this fact using a toy example: consider $n = 3$ which yields $r = 2, p \in \{1,2,3\}, q \in \{1,2,3\}$, and a 9×10 coefficient matrix $\mathbf{M}$ as follows:

$$\begin{bmatrix} l_{1,1} \\ l_{1,2} \\ l_{1,3} \\ l_{2,1} \\ l_{2,2} \\ l_{2,3} \\ l_{3,1} \\ l_{3,2} \\ l_{3,3} \end{bmatrix} = \underbrace{\begin{bmatrix} * & * & * & * & 1 & 1 & * & * & * & * \\ * & * & * & * & * & * & 1 & 1 & * & * \\ * & * & * & * & * & * & * & * & 1 & 1 \\ * & * & 1 & 1 & * & * & * & * & * & * \\ * & * & * & * & 1 & 1 & * & * & * & * \\ * & * & * & * & * & * & 1 & 1 & * & * \\ 1 & 1 & * & * & * & * & * & * & * & * \\ * & * & 1 & 1 & * & * & * & * & * & * \\ * & * & * & * & 1 & 1 & * & * & * & * \end{bmatrix}}_{\mathbf{M}} \begin{bmatrix} y_{1,-2} \\ y_{2,-2} \\ y_{1,-1} \\ y_{2,-1} \\ y_{1,0} \\ y_{2,0} \\ y_{1,1} \\ y_{2,1} \\ y_{1,2} \\ y_{2,2} \end{bmatrix}.$$

Note that the notation $*$ is used to denote the polynomials in t's of $l_{p,q}$. One can easily observe that any 9×9 minor in $\mathbf{M}$ is a polynomial in t's whose degree is at least 4. Indeed, it is really interesting to see that there exists nonzero 9×9 minor that contains a monomial of degree 4 in t's whose coefficients are nonzero. Therefore, our main idea in this example is to seek a specific minor in $\mathbf{M}$ which contains a monomial $t_2^4 1^5$. To see this, we consider a linear transformation of y's to simplify our computations:

$$\begin{bmatrix} z_j \\ z_{2,j} \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} y_{1,j} \\ y_{2,j} \end{bmatrix},$$

for $-2 \leq j \leq 2$. Thus, $l_{p,q}$ has following form with new variables:

$$l_{p,q} = z_{q-p} + (t_{q-1} z_{1-p,q-1} + \ldots) - (t_{3-q} z_{3-p,3-q} + \ldots) + [t_{3-p}(z_{q-3} - z_{3-p+1,q-3}) + \ldots] - [t_{p-1}(z_{q-1} - z_{p,q-1}) + \ldots].$$

With this arrangement, one can notice that t_2 only appears in $l_{i,1}$ and $l_{i,3}$ for $i = 1, 2, 3$. To compute the required monomial, first one needs to select exactly one 1 from the rows corresponding to $l_{1,2}, l_{1,3}, l_{2,2}, l_{3,1}$ and $l_{3,2}$. Next, one can select four t_2's from the rows corresponding to $l_{1,1}, l_{2,1}, l_{2,3}$ and $l_{3,3}$. This yields a monomial $t_2^4 1^5$ with nonzero coefficents. It follows that the minor corresponding to the monomial is nonzero and rank of $\mathbf{M}$ thus is 9.

The authors in [12] generalized the aforementioned computations and proved that there exists a nonzero $n^2 \times n^2$ minor of $\mathbf{M}$ that contains a monomial of degree $(n-1)^2$ with nonzero coefficient for a any given n at the point x^*. It implies that $\mathbf{D}f_x$ has full rank. Using Lemma 9, one can see that $\mathbf{D}f_x$ has full rank at any generic point $x \in \mathcal{T}_n^r(\mathbb{C})$. It follows that from a consequence of the constant rank theorem, image of f has dimension n^2, since $\mathbf{D}f_x$ has full rank at every generic point x. Thus, $f(\mathcal{T}_n^r(\mathbb{C}))$ is dense and it implies that f is dominant. By Lemma 8, one can conclude that image of f contains an open dense subset of $\mathbb{C}^{n \times n}$.

Next, authors in [12] showed that any $n \times n$ matrix can be decomposed into at most $2n+5$ Toeplitz matrices. Nevertheless, to the best of our knowledge there is no efficient algorithm to compute the required decomposition efficiently.

2.0.4. Some Random Expansions in Statistics

Developing computationally efficient structured matrix factorizations are extremely challenging problems. In parallel with developing such factorizations, approximating a large matrix using a matrix expansion with structured elements becomes a hot topic among the communities in computer science and mathematics. In other words, our main challenge is to develop an expansion for approximating a given $\mathbf{A} \in \mathbb{C}^{m\times n}$ such that

$$\mathbf{A} \approx \sum_{i=1}^{r} \mathbf{S}_i,$$

where each $\mathbf{S}_i \in \mathbb{C}^{m\times n}$ has some kind of structure. Simultaneously, r should be small.

In this section, we will briefly discuss two matrix expansions appearing in statistics. Matrix Gaussian series is a random matrix $\mathbf{V}$, which can be expressed as an expansion of fixed matrices, $\{\mathbf{A_k}\}$, weighted by an independent standard normal random variables α_k, i.e., $\mathbf{V} = \sum_k \alpha_k \mathbf{A_k}$. Matrix Gaussian series can be used to represent many kinds of structured Gaussian random matrices. For example, Gaussian Toeplitz matrix $\mathbf{T_v}$, a Toeplitz matrix whose first row and first column are generated from independent standard normal variables, can be represented as a matrix Gaussian expansion: $\mathbf{T_v} = \alpha_0\mathbf{I} + \sum_k^{n-1} \alpha_k \mathbf{C}^k + \sum_k^{n-1} \alpha_{-k}(\mathbf{C}^k)^*$, where

$$\mathbf{C} = \begin{bmatrix} 0 & 1 & \cdots & 0 \\ 0 & 0 & \ddots & \vdots \\ \vdots & & \ddots & 1 \\ 0 & \cdots & \cdots & 0 \end{bmatrix}.$$

Moreover, Gaussian Toeplitz matrices can be found in the context of image and signal processing relation to Gaussian convolution, the study of heat kernels relation to the diffusion equations, and etc [13]. Similarly, Gaussian Wigner matrix, a real symmetric matrix with zero diagonal, can be written as a matrix Gaussian series. Matrix Rademacher series is closely related to matrix Gaussian expansion, defined by a sum of matrices weighted by Rademacher random variables. Applications of this expansion can be found in signal processing, random matrix theory, and etc. We refer [14] for more information.

2.0.5. Factorizations of Polynomials and Their Relation to Matrices

Algebraically, matrix factorizations are closely related to polynomials. Roughly speaking, matrix factorization of a polynomial f in the polynomial ring $\mathbb{C}[x_1, \cdots, x_n]$ is a pair of square matrices $\mathbf{A}, \mathbf{B} \in \mathbb{C}^{m\times m}$ such that $\mathbf{AB} = f\mathbf{I}$, where $\mathbf{I}$ is the identity matrix.

Example 3. *Given $f = xy + yz + zx \in \mathbb{C}[x, y, z]$, then one can compute matrix factorization of f such that $\mathbf{A} = \begin{bmatrix} z & y \\ x & -x-y \end{bmatrix}$ and $\mathbf{B} = \begin{bmatrix} x+y & y \\ x & -z \end{bmatrix}$.*

Generally, this research focuses on studying the singularities of hypersuraces using algebraic tools. For example, matrix factorizations are closely related to homological

properties of the quotient ring: $\mathbb{C}/f$. Interestingly, such a algebraic structure encode the geometric information of a set $Z = \{x \mid f(x) = 0\}$. However, it is out of the scope of this document and we refer seminal work [15] by D. Eisenbud for more details.

3. SVD AND ITS APPLICATIONS IN COMPRESSION

The singular value decomposition (SVD) of $\mathbf{A} \in \mathbb{C}^{m\times n}$ is a central result from linear algebra which can play two roles: factorization and expansion. The elements of the expansion are rank-one matrices and geometrically those rank-one factors are orthogonal. Therefore, the SVD is a key for many practical applications (e.g. principle component analysis and compression algorithms) and for theoretical matrix analysis problems (e.g. eigenvalue problem, multivariate analysis, matrix nearness problems). Basically, the SVD is a factorization of $\mathbf{A}$ into the product of three matrices as describe in following Theorem.

Theorem 11. *If $\mathbf{A} \in \mathbb{C}^{m\times n}$, then there exists unitary matrices $\mathbf{U} \in \mathbb{C}^{m\times m}$ and $\mathbf{V} \in \mathbb{C}^{n\times n}$ such that*

$$\mathbf{U}^*\mathbf{A}\mathbf{V} = \mathbf{\Sigma} = \mathbf{diag}(\sigma_1, \sigma_2, \cdots, \sigma_p),$$

where $\sigma_1 \leq \sigma_2 \leq \ldots \leq \sigma_p < 0$ and $p = \min\{n, m\}$.

Proof: See [10, Theorem 7.3.5].

In this thesis, we refer that σ_i are *singular values* of $\mathbf{A}$, u_i are *left singular vectors* of $\mathbf{A}$, and v_i are *right singular vectors* of $\mathbf{A}$. Next we discuss some nice properties of the SVD.

One of the important property of the SVD is that it gives us a characterization of fundamental vector subspaces, i.e., null space $N(\mathbf{A})$ and column space $R(\mathbf{A})$ of $\mathbf{A}$.

Corollary 12. *If $\mathbf{A}$ has r singular values, then $rank(\mathbf{A}) = r$, and*

$$R(\mathbf{A}) = span\{u_1 \cdots u_r\},$$

$$N(\mathbf{A}) = span\{v_{r+1} \cdots v_n\}.$$

Proof: $rank(\mathbf{A}) = rank(\mathbf{\Sigma}) = r$, since rank is invariant under multiplication of unitary matrices. In order to prove next assertions, suppose that $m \leq n$. Then compute $\mathbf{A}v_i = \sigma_i u_i$ and $\mathbf{A}^* u_i = \sigma_i v_i$ for $i = 1 \cdots n$.

Above discussion leads to have following geometrical interpretation: the singular values σ_i are length of semi axes of the ellipsoid $E = \{\mathbf{A}x : ||x|| = 1\}$, whereas semiaxes directions are given by each u_i. For example, see Figure 3.

From linear algebra, if $\mathbf{A}$ has rank r, then $\mathbf{A}$ can be written as sums of r rank-1 matrices. One of the most important property of the SVD is that we can write $\mathbf{A}$ as a expansion of rank-1 matrices using the SVD.

Corollary 13. *If $\mathbf{A} \in \mathbb{C}^{m\times n}$ and rank $(\mathbf{A}) = r$, then*

$$\mathbf{A} = \sum_{i=1}^{r} \sigma_i u_i v_i^*. \tag{2}$$

Proof: Compute $\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^*$.

An important problem with diverse applications is that what is best rank-k approximation to a given $\mathbf{A}$. Such matrix approximation problems typically measure the distances between matrices with a norm. The Frobenius norm is one of the pervasive choice, since it is so analytically tractable.

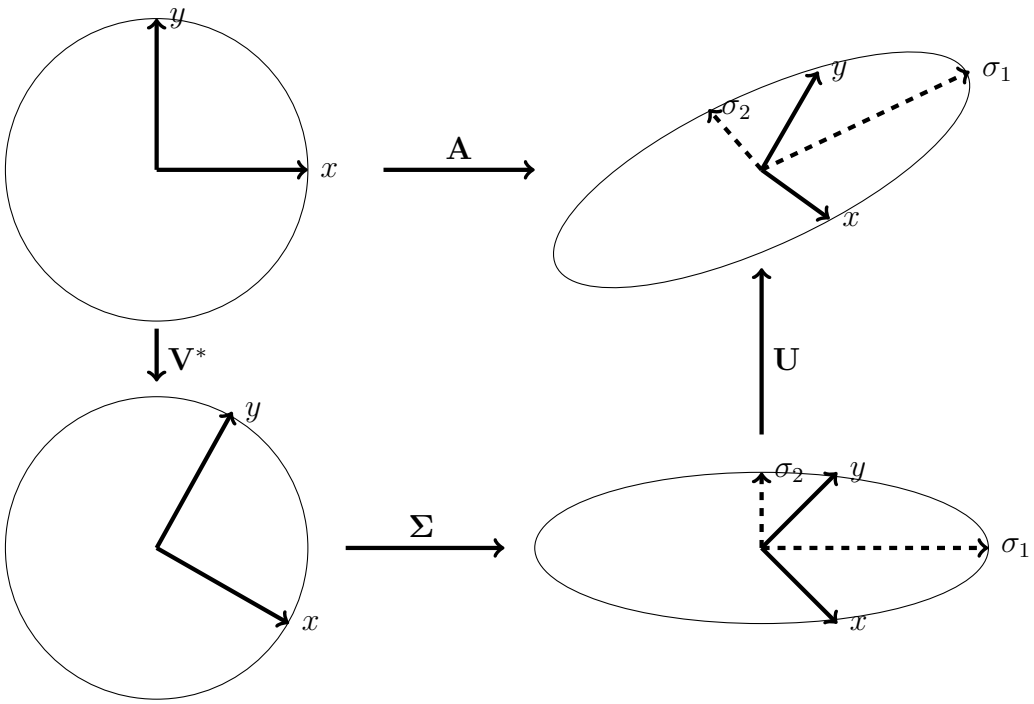

Figure 3. Visualizing the SVD of $\mathbf{A}$.

Definition 14. *Frobenius norm of* $\mathbf{X} \in \mathbb{C}^{n\times n}$ *is a matrix norm defined by*

$$\|\mathbf{X}\|_F = \big(\sum_{i,j=1}^{n} \mathbf{X}_{i,j}^2\big)^{\frac{1}{2}} = \big(\mathbf{Tr}(\mathbf{X}\mathbf{X}^*)\big)^{\frac{1}{2}},$$

where $\mathbf{Tr}$ *denotes the trace of a matrix.*

For example, Eckart and Young gave an elegant algorithmic proof to finding a best rank-k problem in Frobenius norm.

Theorem 15. *For any* $\mathbf{B} \in \mathbb{C}^{m\times n}$ *with rank-*k *at most then*

$$\|\mathbf{A} - \mathbf{A}_k\|_F \leq \|\mathbf{A} - \mathbf{B}\|_F,$$

where $\mathbf{A}_k = \sum_{i=1}^{k} \sigma_i u_i v_i^*$ *is the rank-*k *approximation from SVD of* $\mathbf{A}$.

Proof: See [16].

Inspired from these applications and theoretical substantiation, researchers have being constantly engaging to develop fast and robust algorithms for computing the SVD.

SVD Computation

In this subsection, we will discuss how one can compute the SVD algorithmically. First note that computing SVD is closely related to an eigenvalue problem. For example, consider $\mathbf{A} \in \mathbb{C}^{n\times n}$ and form a larger matrix:

$$\mathbf{H} = \begin{bmatrix} \mathbf{0} & \mathbf{A}^* \\ \mathbf{A} & \mathbf{0} \end{bmatrix}.$$

We can see that singular values of $\mathbf{A}$ are the absolute values of the eigenvalues of $\mathbf{H}$ and singular vectors of $\mathbf{A}$ are the eigenvectors of $\mathbf{H}$. Thus, one can reduce computing SVD into an equivalent symmetric eigenvalue problem which gives the foundations to develop stable SVD algorithms. However, these algorithms never explicitly form a larger matrix like $\mathbf{H}$.

For a general setting, we assume that $\mathbf{A} \in \mathbb{R}^{m \times n}$ and $n \leq m$. Key steps related to SVD algorithms can be outlined as follows:

1. Reduction to upper bidiagonal form $\mathbf{B}$ using the sequence of Householder reflectors $\mathbf{H}_L$ and $\mathbf{H}_R$, i.e., $\mathbf{H}_L^T \mathbf{A} \mathbf{H}_R = \begin{bmatrix} \mathbf{B} \\ \mathbf{0} \end{bmatrix}$ where,

$$\mathbf{B} = \begin{bmatrix} d_1 & s_1 & \cdots & 0 \\ 0 & d_2 & \ddots & \vdots \\ \vdots & & \ddots & s_{n-1} \\ 0 & \cdots & \cdots & d_n \end{bmatrix}.$$

2. Reduction of $\mathbf{B}$ into a diagonal form.

First step can be accomplished by using *Golub-Kahan* bidiagonalization process. For more information, refer [17]. This process can be realized as alternatively applying QR factorization to $\mathbf{A}$ and $\mathbf{A}^*$. In other words, we apply Householder reflectors alternatively on the right and the left of $\mathbf{A}$. Note that applying Householder reflectors on the left produces a column of zeros below the diagonal while applying Householder reflectors on the right produces a row of zeros at the right of the superdiagonal. This process can continue for row and column such that we end up with a upper bidiaganal matrix $\mathbf{B}$, i.e., we apply $m-1$ Householder reflectors at the left and $n-2$ Householder reflectors at the right of $\mathbf{A}$.

The remaining problem is to compute SVD of $\mathbf{B}$. To this end, we can compute tridiaganal symmetric matrix $\mathbf{T} = \mathbf{B}^T \mathbf{B}$, then apply symmetric QR algorithm *implicitly* to $\mathbf{T}$:

- Compute first Givens row rotation $\mathbf{G}$ based on $\mathbf{T} - \mu \mathbf{I}$ required by symmetric QR algorithm, where μ is Wilkinson shift.
- Sequentially, compute and apply Given raw and column rotations of $\mathbf{B}$ such that each rotation set unwanted elements to zero, which are introduced by previous rotations. Continue the process until modified $\mathbf{B}$ is restored as an upper bidiagonal form.

This trick is called *Golub-Kahan SVD* step and can be replaced by faster algorithms developed on divide and conquer paradigm.

Combining everything, we have an algorithm that overwrites $\mathbf{A}$ such that $\mathbf{U}_a^T \mathbf{A} \mathbf{V}_a = \mathbf{\Sigma}_a + \mathbf{D}$, where $\mathbf{U}_a \approx \mathbf{U}$, $\mathbf{V}_a \approx \mathbf{V}$, and $\mathbf{\Sigma}_a \approx \mathbf{\Sigma}$. Computational complexity of first step is $\mathcal{O}(4mn^2 - \frac{4n^3}{3})$ and second step costs generally $\mathcal{O}(n^2)$ operations [18]. Finally, we remark that there are iterative methods to approximate partial SVD of very large matrices.

Application: Principal Component Analysis

Principal Component Analysis (PCA), sometimes called as *Karhunen-Loeve* transform, is closely related to the SVD, and finds many applications in statistical data analysis: data visualization, dimensional reduction, lossy data compression, and feature extraction. Roughly speaking, PCA can be defined as an orthogonal projection onto a lower dimensional *linear space* such that variance of the projected data is maximized, i.e.,

$$\begin{aligned} \text{maximize} \quad & x^T \mathbf{C} x \\ \text{subject to} \quad & x^T x = 1, \end{aligned}$$

where $\mathbf{C} = \frac{1}{n}\mathbf{X}\mathbf{X}^T$ is the covariance matrix of a mean-centered data matrix $\mathbf{X} \in \mathbb{R}^{m \times n}$.

Solving above problem using Lagrangian yields $x^T\mathbf{C}x + \lambda(1 - x^T x)$ and setting derivatives to zero, we have following relationship: $\mathbf{C}x = \lambda x \iff x^T\mathbf{C}x = \lambda$. In other words, maximum variance can be achieved once we have an eigenvector corresponding to largest eigenvalue of $\mathbf{C}$. If we seek r-dimensional linear space ($r \leq n$), one can easily prove that other required directions are eigenvectors corresponding to r largest eigenvalues of $\mathbf{C}$. Therefore, computing the eigenvectors of $\mathbf{C}$ is central in computing PCA.

One approach is to consider eigenvalue decomposition. From the eigenvalue decomposition, we have $\mathbf{C} = \mathbf{Q}\mathbf{D}\mathbf{Q}^T$, where columns of $\mathbf{Q}$ are the eigenvectors of $\mathbf{C}$. Computing the eigen decomposition of covariance matrix $\mathbf{C}$ costs $\mathcal{O}(n^3)$. This computational cost is unaffordable in very high dimensional data sets. Therefore, natural question is to ask that do we have alternatives, but cheaper framework to compute PCA?

From the SVD of $\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$, we can compute $\mathbf{C} = \mathbf{V}\mathbf{\Sigma}^T\mathbf{\Sigma}\mathbf{V}^T = \mathbf{V}\mathbf{S}\mathbf{V}^T$, where $\mathbf{S} = \mathbf{\Sigma}^T\mathbf{\Sigma}$. Comparing above equations, we can conclude that we can extract k-eigentvectors of $\mathbf{C}$ using the right singular vectors of $\mathbf{X}$ and eigenvalues of $\mathbf{C}$ are squares of singular values of $\mathbf{X}$. Putting them all together, the SVD can aid to compute PCA faster with $\mathcal{O}(kn^2)$ operations.

4. PRODUCTS OF CIRCULANT AND DIAGONAL MATRICES IN FACTORING AND EXPANSIONS

In this section, we first give a synopsis of basic computational properties of circulant and diagonal matrices. This will furnish the language to speak about matrix expansions, approximations, and factorizations related with circulant and diagonal matrices. Specifically, we will study following two classical questions: (a). How does the Fourier matrix enable us to develop fast algorithms? (b). Is there any relation between the Fourier matrix and circulant matrices such that one can procure computational advantages from (a) for computations involved in circulant matrices? In addition, we will illustrate the importance of circulant approximation to a given matrix with an application: designing circulant preconditioners for large linear system solvers.

With these motivations, we next consider matrix factorizations and approximations based on circulant and diagonal matrices. We will start our studies with classical results and specific systems which arouse in optical signal processing. Next, we are interested in studying modern matrix decompositions associated with circulant and diagonal factors. Remarkably, these discussions suggest us to introduce a computational and structured matrix expansion to approximate a given matrix $\mathbf{A} \in \mathbb{C}^{n\times n}$. The elements of the expansion are $\mathbf{DCD}$ matrices, i.e., a product of diagonal matrix, circulant matrix, and another diagonal matrix. Among many, $\mathbf{DCD}$ matrices give a natural extension to rank-one matrices. Here, we will study a notion of a matrix rank which is closely related to $\mathbf{DCD}$ expansion. For a given matrix $\mathbf{A} \in \mathbb{C}^{n\times n}$, we denote this rank by $\mathbf{DCD}$-*rank* $(\mathbf{A})$ and we explore the $\mathbf{DCD}$-rank of some specific matrices that are omnipresent in many applications. Finally, we will study a random projection technique that can result in dramatic speed ups once constructed approximately by products of circulant and diagonal matrices.

4.0.1. Review of Circulant Matrices

Circulant matrices naturally arise in many fields of science and mathematics such as linear estimation theory, image processing, numerical analysis, number theory, and etc. In this section, we skim the properties of circulant matrices and for a more detailed analysis refer to [19].

We define n^{th}-order circulant matrix $\mathbf{C} = \mathbf{circ}(c_1, c_2, \ldots, c_n) \in \mathbb{C}^{n\times n}$ as follows:

$$\mathbf{C} = \begin{bmatrix} c_1 & c_2 & c_3 & \cdots & c_n \\ c_n & c_1 & c_2 & \cdots & c_{n-1} \\ c_{n-1} & c_n & c_1 & \cdots & c_{n-2} \\ \vdots & \vdots & & \ddots & \vdots \\ c_2 & c_3 & c_4 & \cdots & c_1 \end{bmatrix},$$

where each row of $\mathbf{C}$ are identical, but are moved one position to the right cyclically. One can easily show that circulant matrices form a linear subspace $\mathcal{C}$ over all $n \times n$

matrices, i.e., if $\mathbf{C_1} = \mathbf{circ}(c_{11}, c_{12}, \ldots, c_{1n})$, $\mathbf{C_2} = \mathbf{circ}(c_{21}, c_{22}, \ldots, c_{2n})$, and $\alpha \in \mathbb{R}$, then

$$\mathbf{C_1} + \mathbf{C_2} = \mathbf{circ}(c_{11} + c_{21}, c_{12} + c_{22}, \ldots, c_{1n} + c_{2n}) \in \mathcal{C} \text{ and}$$
$$\alpha\mathbf{C_1} = \mathbf{circ}(\alpha c_{11}, \alpha c_{12}, \ldots, \alpha c_{1n}) \in \mathcal{C}.$$

Moreover, a circulant matrix can be represented as a matrix polynomial: Suppose that

$$p(z) = c_1 + c_2 z + c_3 z^2 + \ldots + c_n z^{n-1}$$

is a $n-1$ degree polynomial with coefficient in $\mathbb{C}$. Then one can compute a circulant matrix $\mathbf{C} = \mathbf{circ}(c_1, \ldots, c_n)$ as follow:

$\mathbf{C} = p(\pi) = c_1\mathbf{I} + c_2\pi + c_3\pi^2 + \ldots + c_n\pi^{n-1}$, where $\pi = \mathbf{circ}(0, 1, 0, \ldots, 0)$.

A fundamental property of circulant matrices can be outlined using above property as follows:

Theorem 16. *A circulant matrix* $\mathbf{C}$ *can be diagonalized by the Fourier matrix* $\mathbf{F}$*, i.e.,* $\mathbf{C} = \mathbf{F}^*\mathbf{\Lambda}\mathbf{F}$*, where* $\mathbf{\Lambda}$ *is a diagonal matrix,*

$$\mathbf{F}^* = \frac{1}{\sqrt{n}} \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega & \omega^2 & \cdots & \omega^{n-1} \\ 1 & \omega^2 & \omega^4 & \cdots & \omega^{2n-2} \\ \vdots & \vdots & & \ddots & \vdots \\ 1 & \omega^{n-1} & \omega^{2n-2} & \cdots & \omega \end{bmatrix}, \text{ and } \omega = \exp(\tfrac{2\pi i}{n}).$$

Proof: First consider circulant matrix $\pi = \mathbf{circ}(0, 1, 0, \ldots, 0)$. So $(j,k)^{th}$ element of $\mathbf{F}\pi\mathbf{F}^*$ is

$$= \begin{cases} \omega^{k-1}, & \text{if } k = j, \\ 0, & \text{otherwise.} \end{cases}$$

This implies that $\pi = \mathbf{F}^*\mathbf{\Omega}\mathbf{F}$, where $\mathbf{\Omega} = \mathbf{diag}(1, \omega, \omega^2, \ldots, \omega^{n-1})$. Thus we can compute

$\mathbf{C} = p(\pi) = p(\mathbf{F}^*\mathbf{\Omega}\mathbf{F}) = \mathbf{F}^*p(\mathbf{\Omega})\mathbf{F} = \mathbf{F}^*\mathbf{diag}(p(1), p(\omega), \ldots, p(\omega^{n-1}))\mathbf{F} = \mathbf{F}^*\mathbf{\Lambda}\mathbf{F}$.

Corollary 17. *Eigenvalues of* $\mathbf{C}$ *are* $\lambda_i = p(\omega^{i-1})$ *for* $i = 1, \ldots, n$*.*

Proof: Compute $\mathbf{C}\mathbf{F}^* = \mathbf{F}^*\mathbf{\Lambda}$.

Corollary 18. *Suppose that* $\gamma = (c_1, \ldots, c_n) \in \mathbb{C}^{1\times n}$*. If* $\mathbf{C}$ *is a circulant matrix such that* $\mathbf{C} = \mathbf{circ}(\gamma)$ *and its eigenvalues are* $\{\lambda_1, \ldots, \lambda_n\}$*, then* $\sqrt{n}\mathbf{F}^*\gamma^T = (\lambda_1, \ldots, \lambda_n)^T$ *and* $\mathbf{C} = \mathbf{F}^*(\mathbf{diag}(\sqrt{n}\mathbf{F}^*\gamma^T))\mathbf{F}$.

Proof: Note that $\sqrt{n}\mathbf{F}^*(c_1, \ldots, c_n)^T = (p(1), p(\omega), \ldots, p(\omega^{n-1}))^T$.

Discrete Fourier Transform of a vector x, or equivalently $\mathbf{F}x$, is a fundamental operation in signal processing. This matrix-vector product can be computed using a FFT algorithm in $\mathcal{O}(nlog(n))$ operations, whereas matrix-vector product generally costs $\mathcal{O}(n^2)$ operations. However, these algorithms are developed to work on certain dimensions, especially $n = 2^k$ for $k \in \mathbb{N}$. In these dimensions, one can design different frameworks of FFT based on *divide and conquer* approach that factorize a Fourier

matrix $\mathbf{F}$ into a product of different $log(n)$ sparse matrices. Of course, it is one of the great computational framework development in last century which inherits intrinsic beauty and simplicity of numerical Fourier analysis.

Example 4. *Computing* $\mathbf{F}x$ *in* $\mathcal{O}(nlog(n))$ *operations.*

Proof: In this example, a matrix $\mathbf{A} \in \mathbb{C}^{n\times n}$ is denoted by $\mathbf{A}_n$ to distinguish the different dimensions of matrices. Set $n = 2^l$ for some $l \in \mathbb{N}$. Let's begin with a factorization of the Fourier matrix $\mathbf{F}_n$ as below:

$$\mathbf{F}_n x = \underbrace{\begin{bmatrix} \mathbf{I}_{\frac{n}{2}} & \mathbf{\Omega}_{\frac{n}{2}} \\ \mathbf{I}_{\frac{n}{2}} & -\mathbf{\Omega}_{\frac{n}{2}} \end{bmatrix}}_{\mathbf{B}_n} \begin{bmatrix} \mathbf{F}_{\frac{n}{2}} & \mathbf{0} \\ \mathbf{0} & \mathbf{F}_{\frac{n}{2}} \end{bmatrix} \mathbf{P}_n x = \mathbf{B}_n \begin{bmatrix} \mathbf{F}_{\frac{n}{2}} x_1 \\ \mathbf{F}_{\frac{n}{2}} x_2 \end{bmatrix},$$

where $\mathbf{\Omega}_{\frac{n}{2}} = \mathbf{diag}(1, \omega, \ldots, \omega^{\frac{n}{2}-1})$, $x_1, x_2 \in \mathbb{C}^{\frac{n}{2}}$, and $\mathbf{P}_n$ is a permutation matrix. Note that we can recompute $\mathbf{F}_{\frac{n}{2}} x_1$ and $\mathbf{F}_{\frac{n}{2}} x_2$ as follows:

$$\mathbf{F}_{\frac{n}{2}} x_1 = \underbrace{\begin{bmatrix} \mathbf{I}_{\frac{n}{4}} & \mathbf{\Omega}_{\frac{n}{4}} \\ \mathbf{I}_{\frac{n}{4}} & -\mathbf{\Omega}_{\frac{n}{4}} \end{bmatrix}}_{\mathbf{B}_{\frac{n}{2}}} \begin{bmatrix} \mathbf{F}_{\frac{n}{4}} & \mathbf{0} \\ \mathbf{0} & \mathbf{F}_{\frac{n}{4}} \end{bmatrix} \mathbf{P}_{\frac{n}{2}} x_1 = \mathbf{B}_{\frac{n}{2}} \begin{bmatrix} \mathbf{F}_{\frac{n}{4}} x_3 \\ \mathbf{F}_{\frac{n}{4}} x_4 \end{bmatrix},$$

$$\mathbf{F}_{\frac{n}{2}} x_2 = \underbrace{\begin{bmatrix} \mathbf{I}_{\frac{n}{4}} & \mathbf{\Omega}_{\frac{n}{4}} \\ \mathbf{I}_{\frac{n}{4}} & -\mathbf{\Omega}_{\frac{n}{4}} \end{bmatrix}}_{\mathbf{B}_{\frac{n}{2}}} \begin{bmatrix} \mathbf{F}_{\frac{n}{4}} & \mathbf{0} \\ \mathbf{0} & \mathbf{F}_{\frac{n}{4}} \end{bmatrix} \mathbf{P}_{\frac{n}{4}} x_2 = \mathbf{B}_{\frac{n}{2}} \begin{bmatrix} \mathbf{F}_{\frac{n}{4}} x_5 \\ \mathbf{F}_{\frac{n}{4}} x_6 \end{bmatrix},$$

where $x_3, x_4, x_5, x_6 \in \mathbb{C}^{\frac{n}{4}}$.
It follows that

$$\mathbf{F}_n x = \mathbf{B}_n \begin{bmatrix} \mathbf{B}_{\frac{n}{2}} & \mathbf{0} \\ \mathbf{0} & \mathbf{B}_{\frac{n}{2}} \end{bmatrix} \begin{bmatrix} \mathbf{F}_{\frac{n}{4}} x_3 \\ \mathbf{F}_{\frac{n}{4}} x_4 \\ \mathbf{F}_{\frac{n}{4}} x_5 \\ \mathbf{F}_{\frac{n}{4}} x_6 \end{bmatrix}.$$

In general, $\mathbf{F}_n x = \mathbf{A}_l \cdots \mathbf{A}_1 \mathbf{P}_n x$, where $\mathbf{A}_t = \mathbf{diag}(\underbrace{\mathbf{B}_L, \ldots, \mathbf{B}_L}_{\text{r}})$ with $L = 2^t$ and $L = n/r$. By analysing sparsity patterns of factors, one can observe that each $\mathbf{A}_q$ has $2n$ nonzero entries. Thus, total number of required matrix-vector multiplications for $\mathbf{F}x$ is $2nlog(n)$, i.e., $\mathcal{O}(nlog(n))$.

For more theoretical and implementation details about FFT, we refer reader to [20]. Remarkably, implications of Theorem 16 and FFT algorithms enable fast computations based on circulant matrices in dimensions $n = 2^l$ for any $l \in \mathbb{N}$.

Example 5. *Multiplication of two circulant matrices:* $\mathbf{C_1} = \mathbf{circ}(c_{11}, c_{12}, \ldots, c_{1n})$ *and* $\mathbf{C_2} = \mathbf{circ}(c_{21}, c_{22}, \ldots, c_{2n})$.

Proof: Suppose $\mathbf{C}_1 = \mathbf{circ}(c_{11}, \ldots, c_{1n}) = \mathbf{circ}(\alpha_1), \mathbf{C}_2 = \mathbf{circ}(c_{21}, \ldots, c_{2n}) = \mathbf{circ}(\alpha_2)$, and eigenvalues of $\mathbf{C}_1$ and $\mathbf{C}_2$ are $\{\lambda_{11}, \ldots, \lambda_{1n}\}$ and $\{\lambda_{21}, \ldots, \lambda_{2n}\}$, respectively. Then, consider the following computation: $\mathbf{C}_1 = \mathbf{F}^* \mathbf{\Lambda}_1 \mathbf{F}$ and $\mathbf{C}_2 =$

$\mathbf{F}^*\mathbf{\Lambda}_2\mathbf{F} \implies \mathbf{C}_1\mathbf{C}_2 = \mathbf{F}^*\mathbf{diag}(\lambda_{11}\lambda_{21}, \ldots, \lambda_{1n}\lambda_{2n})\mathbf{F}$. Therefore, the required computation reduces to compute a diagonal matrix.

From Corollary 18, one can have

$$\sqrt{n}\mathbf{F}^*\alpha_1^T = (\lambda_{11}, \lambda_{12}, \ldots, \lambda_{1n})^T,$$
$$\sqrt{n}\mathbf{F}^*\alpha_2^T = (\lambda_{21}, \lambda_{22}, \ldots, \lambda_{2n})^T.$$

Thus $\mathbf{C}_1\mathbf{C}_2 = \mathbf{circ}(\gamma)$ can be found using element-wise multiplication of vectors $*$. In particular,

$$\gamma^T = \sqrt{n}\mathbf{F}[\mathbf{F}^*\alpha_1^T * \mathbf{F}^*\alpha_2^T].$$

If the dimension of the matrices lies at the order of magnitude n^2, we can compute circulant-by-circulant multiplication in $\mathcal{O}(nlog(n))$ operations, whereas matrix-matrix product generally costs $\mathcal{O}(n^3)$ operations [2].

Example 6. *Computing the singular values of a circulant matrix.*

Proof: Suppose that one can have a circulant matrix $\mathbf{C}$ such that $\mathbf{C} = \mathbf{circ}(\gamma)$. Recall that the singular values of a circulant matrix $\mathbf{C}$ are the square roots of the eigenvalues of $\mathbf{CC}^*$. Therefore, let's compute the eigenvalues of $\mathbf{CC}^*$:

$$\mathbf{CC}^* = \mathbf{F}^*(\mathbf{diag}(\sqrt{n}\mathbf{F}^*\gamma^T))\mathbf{F}(\mathbf{F}^*(\mathbf{diag}(\sqrt{n}\mathbf{F}^*\gamma^T))\mathbf{F})^* = \mathbf{F}^*(\mathbf{diag}(n|\mathbf{F}^*\gamma^T|^2))\mathbf{F}.$$

Hence, required singular values are the diagonal elements of $(\mathbf{diag}(n|\mathbf{F}^*\gamma^T|^2))^{\frac{1}{2}}$. It follows that we can compute the singular values of circulant matrix in $\mathcal{O}(nlog(n))$ operations, if $n = 2^l$.

Other properties of circulant matrices can be summarized as follows: If $\mathbf{C}_1$ and $\mathbf{C}_2$ are circulant matrices, then $\mathbf{C}_1^T$, $\mathbf{C}_1^*$, $\mathbf{C}_1 + \mathbf{C}_2$ are circulant. Note that $\mathbf{C}_1$ and $\mathbf{C}_2$ commute. In addition, if $\mathbf{C}_1$ is a non-singular matrix, we can find its inverse by setting $\mathbf{F}^*\mathbf{\Lambda}_1^{-1}\mathbf{F}$.

An interesting question one can pose is that how can we compute a circulant approximation to a given square matrix $\mathbf{A}$? One motivation behind is that designing a best circulant preconditioners for very large linear system solvers. Roughly speaking, if $\mathbf{A}x = b$ is a very large system, we might want to employ iterative methods, e.g., Krylov subspace methods, to approximate solutions. However, the rate of convergence of these iterative methods highly depends on the condition number of $\mathbf{A}$. In other words, poor condition number of $\mathbf{A}$ typically means slow the convergence. To speed up convergence, we need to compute a matrix $\mathbf{M}$ which is a optimal or crude approximation to $\mathbf{A}$ such that $\mathbf{M}^{-1}\mathbf{A} \approx \mathbf{I}$. Thus, the eigenvalues of $\mathbf{M}^{-1}\mathbf{A}$ are close to one. As a consequence, one can expect that an iterative method will converge faster for a preconditioned system, $\mathbf{M}^{-1}\mathbf{A}x = \mathbf{M}^{-1}b$, compared to original system $\mathbf{A}x = b$. Moreover, we can gain an extra advantage, if the preconditioner $\mathbf{M}$ is a circulant matrix due to available efficient tool like FFT. For example, if $\mathbf{A}$ is a Toeplitz matrix, one can devise an iterative method with a circulant preconditioner which can compute a solution in $\mathcal{O}(nlog(n))$ operations whereas direct factorization method requires $\mathcal{O}(n^2)$ operations. A classic reference in this direction is Strang's paper [22]. Of course, this proposal spawned a long line of research which gradually advanced into a more

[2] Until late 1960s, it was believed that computing matrix-matrix product requires essentially $\mathcal{O}(n^3)$ operations. However, Strassen changed the way we multiply matrices by introducing an algorithm which can compute the product in $\mathcal{O}(n^{2.808})$ operations [21].

matured subject. Thus, approximating a given matrix $\mathbf{A} \in \mathbb{C}^{n\times n}$ with circlant matrix is vital and we have following optimization problem:

$$arg\min_{\mathbf{C}\in\mathcal{C}} \|\mathbf{A}-\mathbf{C}\|_F^2 . \tag{3}$$

This is a convex optimization problem, since the objective function and the feasible set are convex. Rewriting (3) yields,

$$arg\min_{\mathbf{C}\in\mathcal{C}} \|\mathbf{A}-\mathbf{F}^*\mathbf{\Lambda}\mathbf{F}\|_F^2 = arg\min_{\mathbf{C}\in\mathcal{C}} \|\mathbf{F}\mathbf{A}\mathbf{F}^*-\mathbf{\Lambda}\|_F^2 , \tag{4}$$

since Frobenius norm is unitarily invariant. This implies that minimum is attained by setting each $\mathbf{F}\mathbf{A}\mathbf{F}^*(i,i) = \mathbf{\Lambda}(i,i)$. Authors in [23] have proposed *Algorithm 1* to compute nearest circulant matrix for a given real matrix.

Algorithm 1: *Computing approximate circulant matrix*

Input:

- $\mathbf{A} \in \mathbb{R}^{n\times n}$.
- $\pi = \mathbf{circ}(0,1,0\ldots,0)$.

Steps:

1. Compute the projections: $c_k = \frac{1}{n}\mathbf{Tr}(\mathbf{A}^T\pi^k), \quad k = 0,\ldots,n-1.$
2. Compute $\mathbf{C} = \mathbf{circ}(c_0, c_1, \ldots, c_{n-1})$.

4.0.2. DCD Factors in Matrix Factorizations and Expansions

In this subsection, we will study some theoretical aspects of factoring a matrix into a product of circulant and diagonal matrices. This idea was originally appeared in optical information processing community as described next:

The Fourier transform is a key to explain optical signal processing systems. Before applying Fourier theory, we need to model optical lenses and their signal propagation by using the Fresnel transform. Originally, the Fresnel transform is an infinite dimensional operator. However, advancement of modern circuit technology (e.g., VLSI) throw new light on developing theories to formulate a finite dimensional description of the Fresnel transforms, since diffractive elements (e.g., metalenses and beam splitters) become more accurate, smaller in size, and thin. In addition, it is well known that the sampling theorem guarantees that information about a periodic signal can be extracted from sampling a vector of finite length. Thus, any modern optical system can be completely described as a system that consisting finite dimensional matrices and vectors. In particular, an optical system can be modeled using a product of discrete Fresnel transforms **Fr** and diagonal matrices. It follows that modeling optical system is equivalent to factoring a given matrix into a product of diagonal matrices and discrete Fresnel transforms. Moreover, diffractive optical systems with properly

chosen lenses such that $\mathbf{DFrD^{-1}} = \mathbf{F}$, are in general modeled as a consecutive product of Fresnel transforms and diagonal matrices [24]. Thus, a diffractive optical system can be thought of consisting a product of circulant and diagonal matrices. Therefore, optical information computability problem is equivalent to factoring a given matrix into a product of circulant and diagonal matrices. In other words, the products of circulant and diagonal matrices can be used to realize the discrete diffractive optical systems for information processing [3]. For more information, refer [24]. Next we will illustrate some special systems which can be readily factorized as the product of circulant and diagonal matrices.

Example 7. *Denoted by* $\mathbf{I}, \mathbf{0}, \mathbf{C} \in \mathbb{C}^{2^{n-1}\times 2^{n-1}}$ *are the identity matrix, the zero matrix, and a circulant matrix respectively. Then following matrices can be factored into a product of circulant and diagonal matrices.*

$$\tfrac{1}{\sqrt{2}}\begin{bmatrix} \mathbf{I} & \mathbf{I} \\ \mathbf{I} & -\mathbf{I} \end{bmatrix} = \tfrac{1}{\sqrt{2}}\begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & -i\mathbf{I} \end{bmatrix}\begin{bmatrix} \mathbf{I} & i\mathbf{I} \\ i\mathbf{I} & \mathbf{I} \end{bmatrix}\begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & -i\mathbf{I} \end{bmatrix}.$$

$$\begin{bmatrix} \mathbf{C} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} = \tfrac{1}{4}\begin{bmatrix} \mathbf{I} & \mathbf{I} \\ \mathbf{I} & -\mathbf{I} \end{bmatrix}\begin{bmatrix} \mathbf{C}+\mathbf{I} & \mathbf{C}-\mathbf{I} \\ \mathbf{C}-\mathbf{I} & \mathbf{C}+\mathbf{I} \end{bmatrix}\begin{bmatrix} \mathbf{I} & \mathbf{I} \\ \mathbf{I} & -\mathbf{I} \end{bmatrix}.$$

$$\begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{C} \end{bmatrix} = \tfrac{1}{4}\begin{bmatrix} \mathbf{I} & \mathbf{I} \\ \mathbf{I} & -\mathbf{I} \end{bmatrix}\begin{bmatrix} \mathbf{C}+\mathbf{I} & \mathbf{I}-\mathbf{C} \\ \mathbf{I}-\mathbf{C} & \mathbf{C}+\mathbf{I} \end{bmatrix}\begin{bmatrix} \mathbf{I} & \mathbf{I} \\ \mathbf{I} & -\mathbf{I} \end{bmatrix}.$$

These are special, concrete instances of following theorem.

Theorem 19. *Every matrix* $\mathbf{M} \in \mathbb{C}^{2^n\times 2^n}$ *can be factored into a product of circulant and diagonal matrices.*

Proof: See [24, Theorem 3].

Application: Circular Discrete Convolution

One important application of such system decompositions is that we can realize *circular discrete convolution* operator as a matrix-vector product "$\mathbf{C}z$". To see more details, we adapt the definition of circular discrete convolution of $c \in \mathbb{C}^n$ and $z \in \mathbb{C}^n$ as follows:

$$b = c \circledast z \;\; := \;\; b(i) = \sum_{m=0}^{n-1} c(i)z(i-m) \text{ for } i = 0, \ldots, (n-1).$$

Note that one can rearrange these set of equations as a circulant linear system $\mathbf{C}z = b$, where c is the first column of $\mathbf{C} \in \mathbb{C}^{n\times n}$. It follows that $\mathbf{C}z = \mathbf{F}^*\mathbf{\Lambda}\mathbf{F}z$, since $\mathbf{C}$ can be diagonalized by Fourier basis. Hence, one can compute b as follows:

- Compute discrete transform $y = \mathbf{F}z$, possibly with FFT.
- Compute x by element-wise multiplication of y and diagonal vector of $\mathbf{\Lambda}$, i.e., $x(i) = \mathbf{\Lambda}(i,i)y(i)$.

[3] It means that the optical system $\mathbf{M}$ with a resolution of 2^n pixels can be factored into optical primitives; circulant, diagonal, and the Fourier matrices.

- $\mathbf{F}^* x$, possibly with FFT, transforms back and result is $c \circledast z$.

This approach can gain an improvement with $\frac{3}{2} nlog(n)$ operations, since a direct implementation requires n^2 operations [22]. This is important, since discrete convolution is a fundamental block in *filtering*: a prominent application in digital signal processing.

A similar approach can be utilized to solve linear system $\mathbf{C}z = b$. From basic properties of discrete Fourier transform, one can have following identity:

$$c \circledast z = b \iff \hat{c} * \hat{z} = \hat{b}.$$

Thus, one can recover $\hat{z}$ by element-wise division of $\hat{c}$ and $\hat{b}$. Note that elements of $\hat{c}$ are proportional to the eigenvalues of $\mathbf{C}$, and one can recover z by invoking the inverse discrete Fourier transform of $\hat{z}$.

A Generalization of Theorem 19 can be found in [25, Theorem 2].

Theorem 20. *Let $\mathcal{F} \notin \{\mathbb{F}_3, \mathbb{F}_5\}$ be a field of characteristics $\neq 2$, and let $n \in \mathbb{N}$. Every matrix $\mathbf{A} \in \mathcal{F}_n$ can be decomposed into circulant and diagonal factors with entries in $\mathcal{F}$, where $\mathcal{F}_n$ denotes the ring of all $n \times n$ matrices with entries in $\mathcal{F}$.*

However, original proof of Theorem 20 is somewhat outside the range of algebraic and numerical algorithms and it opens new unsolved problems. Among many, a fundamental problem in one's mind is that how many circulant and diagonal factors are suffice to realize the factorization? Moreover, computability of the factorization also lies at the borderline in between mathematics and computer science. A recent research paper [26] on linear algebra and Fourier analysis shows that factoring a matrix $\mathbf{A} \in \mathbb{C}^{n \times n}$ into a product of diagonal and circulant matrices requires generically at most $2n - 1$ circulant and diagonal factors, i.e.,

$$\mathbf{A} = \mathbf{D}_1 \mathbf{C}_2 \mathbf{D}_3 \cdots \mathbf{D}_{2n-3} \mathbf{C}_{2n-2} \mathbf{D}_{2n-1}. \tag{5}$$

This factorization is constructive: First suppose that $\mathbf{S}$ is a circulant matrix defined as $\mathbf{S} = \mathbf{circ}(0, \ldots, 1) \in \mathbb{C}^{n \times n}$. Then, given any $\mathbf{A} \in \mathbb{C}^{n \times n}$ can be expressed as a unique polynomial p over the diagonal matrices of degree at most $n - 1$ such that $p(\mathbf{S}) = \mathbf{A}$, i.e., $p(\mathbf{S}) = \sum_{i=0}^{n-1} \mathbf{D}_i \mathbf{S}^i = \mathbf{A}$, where each $\mathbf{D}_i$ is a diagonal matrix. In addition, authors in [26] factor $\mathbf{D}_0 + \mathbf{D}_1 \mathbf{S}$ into a product $\hat{\mathbf{D}}_1 (\mathbf{I} + \alpha \mathbf{S}) \hat{\mathbf{D}}_2$, where $\hat{\mathbf{D}}_1$ and $\hat{\mathbf{D}}_2$ are diagonal matrices. Of course, $\hat{\mathbf{D}}_1 (\mathbf{I} + \alpha \mathbf{S}) \hat{\mathbf{D}}_2$ is a $\mathbf{DCD}$ matrix. Factoring a matrix $\mathbf{A}$ constructively into a product of $2n - 1$ diagonal and circulant matrices can be outlined as follow:

1. Factor p into linear terms, i.e., $p(\mathbf{S}) = (\mathbf{S} - \mathbf{D}_0)(\mathbf{S} - \mathbf{D}_1) \cdots (\mathbf{S} - \mathbf{D}_{n-1})$. This can be achieved by consecutively solving structured systems of polynomial equations.

2. Convert each factor $(\mathbf{S} - \mathbf{D}_i) = -(\mathbf{D}_i + \mathbf{D}_I \mathbf{S})$ into a $\mathbf{DCD}$ matrix as in aforementioned discussion, where $\mathbf{D}_I = -\mathbf{I}$.

3. With this setting, $2n - 2$ diagonal factors and a circulant factor are indispensable to compute the desired factorization (5).

Another interesting factorization can be found with applications in signal processing. For more information, see [27]. In this factorization, authors first decomposed any square matrix $\mathbf{A}$ into a following structured factorization using Jordan canonical form: $\mathbf{A} = \mathbf{X}\mathbf{J}\mathbf{X}^{-1} \implies \alpha\mathbf{I} - (\alpha\mathbf{I} - \mathbf{A}) = \alpha\mathbf{X}(\mathbf{I} - (\mathbf{I} - \mathbf{J}/\alpha))\mathbf{X}^{-1}$ yields

$$\mathbf{A} = \alpha\mathbf{X} \underbrace{\left(\mathbf{I} - \begin{bmatrix} \mathbf{I} - \mathbf{J}_1/\alpha & \mathbf{0} & \cdots \\ \mathbf{0} & \mathbf{0} & \cdots \\ \vdots & & \ddots \end{bmatrix}\right)}_{\mathbf{B}_1} \cdots \underbrace{\left(\mathbf{I} - \begin{bmatrix} \ddots & & \vdots \\ \cdots & \mathbf{0} & \mathbf{0} \\ \cdots & \mathbf{0} & \mathbf{I} - \mathbf{J}_l/\alpha \end{bmatrix}\right)}_{\mathbf{B}_l} \mathbf{X}^{-1},$$

where $\mathbf{J}_1, \ldots, \mathbf{J}_l$ are Jordan blocks of $\mathbf{A}$. See Subsection 2.0.3 for more information about Jordan form and Jordan blocks. By analysing carefully sparsity structures of each block $\mathbf{B}_i$, one can decompose each $\mathbf{B}_i$ into the identity plus rank-one factors, i.e, $\mathbf{B}_i = (\mathbf{I} + \mathbf{X}_1)(\mathbf{I} + \mathbf{X}_2)\cdots(\mathbf{I} + \mathbf{X}_{l_i})$, where each $\mathbf{X}_i$ is a rank-one matrix. Thus, one can substitute these identities with implications of Example 8 to yield a factorization:

$$\mathbf{A} = \alpha(\mathbf{I} + \mathbf{D}_1\mathbf{C}_1\mathbf{D}_2)(\mathbf{I} + \mathbf{D}_3\mathbf{C}_2\mathbf{D}_4)\cdots(\mathbf{I} + \mathbf{D}_{2l-1}\mathbf{C}_l\mathbf{D}_{2l}). \tag{6}$$

Thus, another view point of product of circulant and diagonal matrices is that we can use $\mathbf{DCD}$ matrices as elements in a multiplicative approximation of $\mathbf{A}$. Moreover, it is interesting task to study and develop links between the factorizations (5) and (6). For instance, if the product ${\mathbf{D_1}}^{-1}{\mathbf{D_2}}^{-1}$ is the identity matrix, then one can decompose each factor in (6) into a product of circulant and diagonal factors, since $\mathbf{I}+\mathbf{D}_1\mathbf{C}_1\mathbf{D}_2 = \mathbf{D}_1(\mathbf{D}_1^{-1}\mathbf{D}_2^{-1}+\mathbf{C})\mathbf{D}_2$. At this point, we do not know how to devise such factorization in general settings. Therefore, let's formulate the following more general open problem.

Note: An Open Problem

Factoring $\mathbf{I} + \mathbf{D}_1\mathbf{C}_1\mathbf{D}_2$ into a product of circulant and diagonal factors, where $\mathbf{D}_1, \mathbf{D}_2 \in \mathcal{D}$ and $\mathbf{C}_1 \in \mathcal{C}$.

Inspired by these decompositions, it remains to be seen which way is computationally, or, from the point of view of applications, the best approach to work with product of circulants and diagonals. However, structured matrix factorizations are in general computationally hard problems with appealing mathematical properties. For example, as discussed in Section 2 nonnegative matrix factorization is NP hard. Thus a natural and interesting question towards computation is developing a matrix expansion that can approximate a given $\mathbf{A}$ rapidly. To this end, a fundamental question is choosing elements or basis with proper computational properties. In our context, we develop a matrix expansion based on $\mathbf{DCD}$ matrices. For instance, if $\mathbf{A} \in \mathbb{C}^{2^n \times 2^n}$ is a sum of r-$\mathbf{DCD}$ matrices, then $\mathbf{A}x = \sum_{i=1}^{r} \mathbf{D}_{2i-1}\mathbf{F}^*\mathbf{\Lambda}_i\mathbf{F}\mathbf{D}_{2i}x$ can be computed in $\mathcal{O}(rnlog(n))$ operations by invoking the FFT. Remarkably, $\mathbf{A} = \sum_{i=1}^{r} \mathbf{D}_{2i-1}\mathbf{C}_i\mathbf{D}_{2i}$ also promises that one can develop peculiar parallel and distributed algorithm hierarchies to perform elementary operations like matrix-vector multiplication. Moreover, following example will demonstrate another motivation.

Example 8. *Any rank-one matrix $\mathbf{A} = uv^*$ with $u, v \in \mathbb{C}^n$ is a $\mathbf{DCD}$ matrix.*

Proof: Set $\mathbf{D_1} = \mathbf{diag}(u_1, u_2, \ldots, u_n), \mathbf{D_2} = \mathbf{diag}(\overline{v}_1, \overline{v}_2, \ldots, \overline{v}_n)$, and $\mathbf{C}$ as a circulant matrix having all 1 in first row.

Example 8 seems very artificial to restrict $\mathbf{C}$ to be all ones. However, we recall that one can write any rank-r square matrix $\mathbf{A}$ as an expansion of rank-one matrices with r factors, i.e., $\mathbf{A} = \sum_{i=1}^{k} \sigma_i u_i v_i^*$. See Corollary 13 for more details. From Example 8, any square matrix $\mathbf{A}$ is thus a sum of at most k-$\mathbf{DCD}$ matrices. With this in mind, $\mathbf{DCD}$ matrices can be regarded as playing an important role in additive and multiplicative matrix approximations which leads to have following definition.

Definition 21. $\mathbf{DCD}$*-rank of a matrix* A *is the smallest* r *which gives zero in (1).*

Next, we will illustrate the computations of the $\mathbf{DCD}$-rank of some important matrices. Of course, trivial examples include diagonal or circulant matrices, then the $\mathbf{DCD}$-rank of these matrices are 1.

Example 9. *If* $\mathbf{A}$ *is a rank-*k *matrix, then the* $\mathbf{DCD}$*-rank of* $\mathbf{A}$ *is at most* k*.*

Example 10. *If* $\mathbf{A}$ *is a skew circulant matrix, then the* $\mathbf{DCD}$*-rank of* $\mathbf{A}$ *is* 1*, where a skew circulant matrix is a circulant matrix with a change in sign to all elements below its diagonal.*

Proof: Refer [19, p. 85].

Example 11. *If* $\mathbf{A}$ *has the* $\mathbf{DCD}$*-rank* k *and* $\mathbf{B}$ *has the* $\mathbf{DCD}$*-rank* l*, then* $a\mathbf{A} + b\mathbf{B}$ *has the* $\mathbf{DCD}$*-rank at most* $l + k$ *for some scalars* a *and* b*.*

Proof: Suppose $\mathbf{C} = a\mathbf{A} + b\mathbf{B}$. Notice that $\mathbf{A}$ can be decompose into at most sum of k-$\mathbf{DCD}$ matrices and $\mathbf{B}$ is a sum of at most l-$\mathbf{DCD}$ matrices. Therefore, the $\mathbf{DCD}$-rank of $\mathbf{C}$ is at most $k + l$.

Example 12. *If* $\mathbf{A}$ *is a* $\mathbf{CD}$ *matrix, where* $\mathbf{C}$ *is a* $\{e^{-i\theta}\}$*-circulant matrix, then the* $\mathbf{DCD}$*-rank of* $\mathbf{A}$ *is 1.*

Proof: Note that $\{e^{-i\theta}\}$-circulant matrix can be decomposed into $\mathbf{C} = \mathbf{\Omega}^*\mathbf{C_1}\mathbf{\Omega}$, where $\mathbf{C_1}$ is a circulant and $\mathbf{\Omega}$ is a diagonal matrix. Therefore, $\mathbf{A}$ is a $\mathbf{DCD}$ matrix.

Example 13. *If* $\mathbf{A}$ *is a monomial matrix, i.e.,* $\mathbf{PD}$ *matrix, then the* $\mathbf{DCD}$*-rank of* $\mathbf{A}$ *is at most* n*, where* $\mathbf{P}$ *is a permutation matrix.*

Proof: It follows from the Example 11.

Example 14. *If* $\mathbf{A}$ *is a monomial matrix, i.e.,* $\mathbf{PD}$ *matrix, and* $\mathbf{D}_1$ *is a diagonal matrix, then the* $\mathbf{DCD}$*-rank of* $\mathbf{AD}_1$ *and* $\mathbf{D}_1\mathbf{A}$ *is at most* n*.*

Proof: Note that $\mathbf{AD_1} = \mathbf{PDD}_1$. Thus, $\mathbf{AD}_1$ is a monomial matrix. To prove other, set $\mathbf{D}_1\mathbf{A} = \mathbf{D}_1\mathbf{PD} = \mathbf{P}(\mathbf{P}^*\mathbf{D}_1\mathbf{P})\mathbf{D}$ and observe that it is also a monomial matrix.

Example 15. *If* $\mathbf{A}$ *is a* $\mathbf{CDC}$ *matrix, then the* $\mathbf{DCD}$*-rank of* $\mathbf{FAF}^*$ *is* 1*.*

Proof: Note that $\mathbf{FAF}^* = \mathbf{FCDCF}^*$ is a $\mathbf{DCD}$ matrix.

Example 16. *If* $\mathbf{A}$ *is a sum of* $\mathbf{DCD}$ *matrices, then* $\mathbf{A}^T$ *is also a sum* $\mathbf{DCD}$ *matrices. In particular, if* $\mathbf{A}$ *is a* $\mathbf{DCD}$ *matrix, then* $\mathbf{A}^T$ *is a* $\mathbf{DCD}$ *matrix.*

Example 17. *If* $\mathbf{A} = \mathbf{D}_1 + \mathbf{D}_2\mathbf{S}$*, then the* $\mathbf{DCD}$*-rank of* $\mathbf{A}$ *is* 1*, where we define* $\mathbf{S} = \mathbf{circ}(0, \ldots, 1)$ *and* $\mathbf{D}_1$ *and* $\mathbf{D}_2$ *are diagonal matrices.*

Proof: Refer [26, Theorem 2.10].

It is seemingly useful to study the $\mathbf{DCD}$ expansions of structured matrices, since their $\mathbf{DCD}$-ranks give key information about optimal number of factors and related compressions. Of course, this information is also crucial for designing algorithms towards applications. Therefore, we will study the $\mathbf{DCD}$-rank of some structured matrices which are appeared in modern computational sciences and engineering.

First, we study the Toeplitz matrices that inherit some of the most attractive computational properties. Hence, Toeplitz matrices have being used in a wide range of stable and fast algorithms. For example, multiplication, inversion, and $\mathbf{LU}$ decompositon of a $n \times n$ Toeplitz matrix can be computed in $\mathcal{O}(n^2)$ operations and remarkably, Toeplitz systems can be solved in $\mathcal{O}(nlog^2(n))$ operations. See Subsection 2.0.3 and references therein for more details. Therefore, our next move is to investigate the relation between Toeplitiz matrices and $\mathbf{DCD}$ expansion.

Example 18. *If* $\mathbf{A}$ *is a Toeplitz matrix, then the* $\mathbf{DCD}$*-rank of* $\mathbf{A}$ *is at most* 2*, where Toeplitz matrix* $\mathbf{T}$ *is a constant matrix along its diagonals, i.e.,*

$$\mathbf{T} = \begin{bmatrix} t_1 & t_2 & t_3 & \cdots & t_n \\ t_{n+1} & t_1 & t_2 & \cdots & t_{n-1} \\ t_{n+2} & t_{n+1} & t_1 & \cdots & t_{n-2} \\ \vdots & \vdots & & \ddots & \vdots \\ t_{2n-1} & t_{2n-1} & t_{2n-2} & \cdots & t_1 \end{bmatrix}.$$

Proof: One important property of Toeplitz matrix $\mathbf{A}$ is that we can decompose $\mathbf{A}$ into a circulant matrix $\mathbf{C}$, and a skew circulant matrix $\mathbf{S}$, i.e., $\mathbf{A} = \mathbf{C} + \mathbf{S}$. Note that $\mathbf{S}$ and $\mathbf{C}$ are $\mathbf{DCD}$ matrices. Therefore, the $\mathbf{DCD}$-rank of $\mathbf{A}$ is at most 2.

In the light of research paper [12], almost any matrix $\mathbf{A} \in \mathbb{C}^{n\times n}$ can be decomposed into a product of at most $2n + 5$ Toeplitz matrices, i.e.,

$$\mathbf{A} = \mathbf{T}_1\mathbf{T}_2 \cdots \mathbf{T}_{2n+5}, \tag{7}$$

where each $\mathbf{T}_i$ is a Toeplitz matrix. See Subsection 2.0.3 and [12] for details. To the best of our knowledge, there is no efficient numerical algorithm to compute such a decomposition (7). On the other hand, given the decomposition (7) of a square matrix $\mathbf{A}$, we can solve linear system $\mathbf{A}x$ recursively in $\mathcal{O}((2n + 5)nlog^2(n))$ operations using standard Toeplitz solvers. Moreover, one can naturally combine aforementioned results to decompose a generic square matrix $\mathbf{A} \in \mathbb{C}^{n\times n}$ into a sum of products of $\mathbf{DCD}$ matrices:

$$\mathbf{A} = \mathbf{Q}_1 + \mathbf{Q}_2 + \ldots + \mathbf{Q}_r,$$

where each $\mathbf{Q}_i$ is a product of $2n + 5$ $\mathbf{DCD}$ matrices and $r = 2^{2n+5}$.

Example 19. *If* $\mathbf{A} = \mathbf{D}_1\mathbf{T}\mathbf{D}_2$ *for some diagonal matrices* $\mathbf{D}_1$ *and* $\mathbf{D}_2$*, then the* $\mathbf{DCD}$*-rank of* $\mathbf{A}$ *is* 2*, where* $\mathbf{T}$ *is a Toeplitz matrix.*

Proof: Note that the $\mathbf{DCD}$-rank is invariant under diagonal scaling from the left and right of a Toeplitz matrix.

Example 20. *If* $\mathbf{A}$ *is an upper bidiagonal matrix, then the* $\mathbf{DCD}$*-rank is at most* 2*.*

Proof: Consider a decomposition: $\mathbf{A} = \mathbf{D_A} + \mathbf{U_A}$, where $\mathbf{D_A}$ is a diagonal matrix whose diagonal elements are equal to the diagonal elements of $\mathbf{A}$ and $\mathbf{U_A}$ is a super diagonal matrix whose super diagonal elements are equal to the super diagonal elements of $\mathbf{A}$. Then $\mathbf{U_A}$ is a $\mathbf{DCD}$ matrix. To see this, consider the following cirulant and diagonal matrices:

- $\mathbf{D}_1$ is a diagonal matrix whose first $n-1$ diagonal elements are super diagonal elements of $\mathbf{U_A}$ and the last element is zero.
- $\mathbf{C}$ is a circulant matrix such that $\mathbf{C} = \mathbf{circ}(0, 1, 0, \ldots, 0)$.
- $\mathbf{D}_2$ is a diagonal matrix whose first diagonal element is zero and other diagonal elements are ones.

Note that $\mathbf{D}_1\mathbf{C}\mathbf{D}_2 = \mathbf{U_A}$ and $\mathbf{D_A}$ is a $\mathbf{DCD}$. It follows that the $\mathbf{DCD}$-rank of $\mathbf{A}$ is 2.

The next example, a special case of Example 20, illustrates another way of computing the $\mathbf{DCD}$-rank of an upper bidiagonal matrix with a constant diagonal matrix.

Example 21. *If* $\mathbf{A}$ *is a constant diagonal with a super diagonal matrix, then the* $\mathbf{DCD}$*-rank of* $\mathbf{A}$ *is* 2*.*

Proof: We demonstrate the computation by taking $\mathbf{A}$ as a 3×3 matrix:

$$\mathbf{A} = \begin{bmatrix} 1 & x_1 & 0 \\ 0 & 1 & x_2 \\ 0 & 0 & 1 \end{bmatrix}.$$

Consider $\mathbf{A}_1 = (\mathbf{D}_1 + \mathbf{D}_2\mathbf{S})^T$, where $\mathbf{D}_1 = \mathbf{diag}(1,1,1)$, $\mathbf{D}_2 = \mathbf{diag}((x_1x_2)^{-1}, x_1, x_2)$, and $\mathbf{S} = \mathbf{circ}(0, \ldots, 1)$. Example 17 and Example 16 guarantee that $\mathbf{A}_1$ is a $\mathbf{DCD}$ matrix. Assume following $\mathbf{DCD}$ matrix:

$$\mathbf{A}_2 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ (x_1x_2)^{-1} & 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & (x_1x_2)^{-1} & 0 \\ 0 & 0 & (x_1x_2)^{-1} \\ (x_1x_2)^{-1} & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}.$$

Note that $\mathbf{A} = \mathbf{A}_1 + \mathbf{A}_2$ and $\mathbf{A}_2$ is a $\mathbf{DCD}$ matrix. Thus, the $\mathbf{DCD}$-rank of $\mathbf{A}$ is 2. In addition, one can generalize above computations for an arbitrary n.

Example 22. *Let* $f_1 = a_px^p + a_{p-1}x^{p-1} \cdots + a_0$ *and* $f_2 = b_qx^q + b_{q-1}x^{q-1} \cdots + b_0$ *be polynomials in* $\mathbb{C}[x]$*. Then, Sylvester matrix* $\mathbf{S} \in \mathbb{C}^{(p+q)\times(p+q)}$ *of* f_1 *and* f_2 *is defined as follows:*

$$\mathbf{S} = \begin{bmatrix} a_p & \cdots & a_0 & 0 & \cdots & 0 \\ 0 & \ddots & & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & & \ddots & 0 \\ 0 & \cdots & 0 & a_p & \cdots & a_0 \\ b_q & \cdots & b_0 & 0 & \cdots & 0 \\ 0 & \ddots & & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & & \ddots & 0 \\ 0 & \cdots & 0 & b_q & \cdots & b_0 \end{bmatrix}.$$

If $\mathbf{A}$ *is a Sylvester matrix, then the* $\mathbf{DCD}$*-rank of* $\mathbf{A}$ *is* 2*.*

Proof: Set two circulant matrices $\mathbf{C}_1 \in \mathbb{C}^{(p+q)\times(p+q)}$ and $\mathbf{C}_2 \in \mathbb{C}^{(p+q)\times(p+q)}$ such that $\mathbf{C}_1 = \mathbf{circ}(a_p, \ldots, a_0, 0, \ldots, 0)$ and $\mathbf{C}_2 = \mathbf{circ}(b_0, 0, \ldots, 0, b_q, \ldots, b_1)$. Next consider diagonal matrices $\mathbf{D}_{11} = \mathbf{diag}(\underbrace{1, \ldots, 1}_{\text{q}}, \underbrace{0 \ldots, 0}_{\text{p}})$ and $\mathbf{D}_{12} = \mathbf{diag}(\underbrace{0, \ldots, 0}_{\text{q}}, \underbrace{1 \ldots, 1}_{\text{p}})$. Note that we have $\mathbf{A} = \mathbf{D}_{11}\mathbf{C}_1\mathbf{D}_{21} + \mathbf{D}_{12}\mathbf{C}_2\mathbf{D}_{22}$, where $\mathbf{D}_{21}$ and $\mathbf{D}_{22}$ are identity matrices. It follows that the $\mathbf{DCD}$-rank of $\mathbf{A}$ is 2.

Example 23. *If* $\mathbf{A} \in \mathbb{C}^{2n\times 2n}$ *is a structured matrix with all nonzero* x_i*'s as below, then* $\mathbf{A}$ *is a* $\mathbf{DCD}$ *matrix.*

$$\mathbf{A} = \begin{bmatrix} 0 & x_1 & 0 & \cdots & 0 & x_2 \\ x_3 & 0 & x_4 & 0 & \cdots & 0 \\ 0 & x_5 & 0 & x_6 & 0 & \cdots \\ \vdots & \cdots & \ddots & \ddots & \ddots & \ddots \\ 0 & \cdots & 0 & x_{4n-3} & 0 & x_{4n-2} \\ x_{4n-1} & 0 & \cdots & 0 & x_{4n} & 0 \end{bmatrix}.$$

Proof: Consider a circulant matrix $\mathbf{C}_1 = \mathbf{circ}(0, 1, 0, \cdots, 0, 1)$ and two diagonal matrices: $\mathbf{D}_1 = \mathbf{diag}(d_1, \cdots, d_{2n})$ and $\mathbf{D}_2 = \mathbf{diag}(\alpha_1, \cdots, \alpha_{2n})$. Note that $\mathbf{D}_1\mathbf{C}_1\mathbf{D}_2$ yields following two system of linear equations:

$$\begin{cases} d_1\alpha_2 = x_1 \\ d_2\alpha_3 = x_4 \\ \vdots \\ d_{2n-1}\alpha_{2n} = x_{4n-2} \\ d_{2n}\alpha_1 = x_{4n-1} \end{cases} \text{and} \begin{cases} d_1\alpha_{2n} = x_2 \\ d_2\alpha_1 = x_3 \\ d_3\alpha_2 = x_5 \\ \vdots \\ d_{2n}\alpha_{2n-1} = x_{4n} \end{cases}.$$

With this information, we can compute unknown d values in terms of α values using first system of equations and insert them into the second system of equations to solve α values. This yields a structured sparse matrix which can be solved very fast to determine α values. For instance, we have following matrix for $n = 4$,

$$\begin{bmatrix} \times & 0 & 0 & \times & 0 & 0 & 0 & 0 \\ 0 & \times & 0 & 0 & \times & 0 & 0 & 0 \\ 0 & 0 & \times & 0 & 0 & \times & 0 & 0 \\ 0 & 0 & 0 & \times & 0 & 0 & \times & 0 \\ 0 & 0 & 0 & 0 & \times & 0 & 0 & \times \\ \times & 0 & 0 & 0 & 0 & \times & 0 & 0 \\ 0 & \times & 0 & 0 & 0 & 0 & \times & 0 \\ 0 & 0 & \times & 0 & 0 & 0 & 0 & \times \end{bmatrix}.$$

Once solve α values, we can substitute them into the first system of equations to determine unknown d values.

Example 24. *Any* $\mathbf{A} \in \mathbb{C}^{2\times 2}$ *with nonzero entries is a* $\mathbf{DCD}$ *matrix.*

Example 25. *If* $\mathbf{A} \in \mathbb{C}^{2n\times 2n}$ *is a structured matrix whose sparsity pattern agrees with that of* $\mathbf{C} = \mathbf{circ}(0,\ldots,\underbrace{1}_{i},0,\ldots,\underbrace{1}_{j},0,\ldots,0)$*, then* $\mathbf{A}$ *is a* $\mathbf{DCD}$ *matrix.*

Proof: Suppose that we have diagonal matrices $\mathbf{D}_1 = \mathbf{diag}(d_1,\ldots,d_{2n})$ and $\mathbf{D}_2 = \mathbf{diag}(\alpha_1,\ldots,\alpha_{2n})$ and a circulant matrix: $\mathbf{C}_1 = \mathbf{circ}(0,\ldots,0,1,0,\ldots,0,1,0,\ldots,0) \in \mathbb{C}^{2n\times 2n}$ which agrees with the sparsity pattern of a given matrix $\mathbf{A} \in \mathbb{C}^{2n\times 2n}$. Looking at nonzero subdiagonals of $\mathbf{D}_1\mathbf{C}_1\mathbf{D}_2$, one can compute two systems of linear equations in a similar way as we discussed in Example 23.

Theorem 22. *If* $\mathbf{A}$ *is a square matrix such that* $\mathbf{A} \in \mathbb{C}^{2n\times 2n}$ *with nonzero entries, then the* $\mathbf{DCD}$*-rank of* $\mathbf{A}$ *is at most* n*. In addition, we have* $\mathbf{DCD}$*-rank*$(\mathbf{A}) <$ *rank*$(\mathbf{A}) \leq 2n$*.*

Proof: For $n = 1$, see Example 24. For $n > 1$, observe that

$$\mathbf{A} = \sum_{i=1}^{n-1} \mathbf{A}_i + \mathbf{A}_j,$$

where each $\mathbf{A}_i$ has same sparsity pattern as $\mathbf{C}_i = \mathbf{circ}(0,\ldots,\underbrace{1}_{\text{i+1}},0,\ldots,\underbrace{1}_{\text{2n-(i-1)}},0,\ldots,0) \in \mathbb{C}^{2n\times 2n}$ and if $\mathbf{A}_i(m,n)$ is a nonzero element then, $\mathbf{A}_i(m,n) = \mathbf{A}(m,n)$. $\mathbf{A}_j$ has same sparsity pattern as $\mathbf{C}_j = \mathbf{circ}(\underbrace{1}_{\text{1}},0,\ldots,\underbrace{1}_{\text{n+1}},0,\ldots,0) \in \mathbb{C}^{2n\times 2n}$ and if $\mathbf{A}_j(m,n)$ is a nonzero element then, $\mathbf{A}_j(m,n) = \mathbf{A}(m,n)$. Example 25 implies that every $\mathbf{A}_i$ and $\mathbf{A}_j$ are $\mathbf{DCD}$ matrices. Thus, the $\mathbf{DCD}$-rank of $\mathbf{A}$ is at most n. Inequalities can be proved by using the rank theorem in linear algebra.

Next, we would like to propose a natural generalization of $\mathbf{DCD}$ expansion by adapting an extension of diagonal matrices. We follow the definition of the generalized diagonal of a matrix proposed by Bhatia in [28] and define a generalized diagonal matrix as follows:

Definition 23. *A generalized diagonal matrix* $\hat{\mathbf{D}}$ *is a matrix in which each row and each column contain exactly one nonzero element.*

Inspired with this definition, we further study a generalization of $\mathbf{DCD}$ expansion consisting circulant matrices and generalized diagonal matrices. In other words, we study a matrix expansion whose elements are $\hat{\mathbf{D}}\mathbf{C}\hat{\mathbf{D}}$ matrices, i.e., a product of generalized diagonal matrix, circulant matrix, and another generalized diagonal matrix. Nevertheless, our report below is extremely preliminary and it is a future research to explore more properties of $\hat{\mathbf{D}}\mathbf{C}\hat{\mathbf{D}}$ expansion. Remarkably, if we fix $\hat{\mathbf{D}}\mathbf{C}$, then a set of all $\hat{\mathbf{D}}\mathbf{C}\mathbf{D}$ forms a linear space under diagonal matrix addition and scalar multiplication. Further, $(\hat{\mathbf{D}}\mathbf{C}\hat{\mathbf{D}})^T$ and $(\hat{\mathbf{D}}\mathbf{C}\hat{\mathbf{D}})^*$ are also $\hat{\mathbf{D}}\mathbf{C}\hat{\mathbf{D}}$ matrices as well.

Definition 24. $\hat{\mathbf{D}}\mathbf{C}\hat{\mathbf{D}}$*-rank of a given matrix* $\mathbf{A} \in \mathbb{C}^{n\times n}$ *is the smallest* r *which gives zero in the problem defined as follows:*

$$\min_{\hat{\mathbf{D}}_i,\hat{\mathbf{D}}_j\in\hat{\mathcal{D}},\mathbf{C}_k\in\mathcal{C}} \left\| \mathbf{A} - \sum_{i,j,k}^{r} \hat{\mathbf{D}}_i \mathbf{C}_k \hat{\mathbf{D}}_j \right\|_F, \tag{8}$$

where $\hat{\mathcal{D}}$ and $\mathcal{C}$ denote a set of all generalized diagonal matrices and circulant matrix subspace, respectively.

One can easily observe that the Examples that we described the computations of the $\mathbf{DCD}$-ranks can be extended for computing the $\hat{\mathbf{D}}\mathbf{C}\hat{\mathbf{D}}$-rank of these matrices.

Example 26. *If $\mathbf{A}$ is a monomial matrix, i.e., $\mathbf{PD}$ matrix, then the $\hat{\mathbf{D}}\mathbf{C}\hat{\mathbf{D}}$-rank of $\mathbf{A}$ is 1, where $\mathbf{P}$ is a permutation matrix.*

Proof: $\mathbf{PD}$ is a $\hat{\mathbf{D}}\mathbf{C}\hat{\mathbf{D}}$ matrix.

Example 27. *If $\mathbf{A}$ is a Hankel matrix, then $\mathbf{A}$ is at most a sum of two $\hat{\mathbf{D}}\mathbf{C}\hat{\mathbf{D}}$ matrices, where Hankel matrix $\mathbf{H}$ is a constant matrix along its anti-diagonals, i.e.,*

$$\mathbf{H} = \begin{bmatrix} h_1 & h_2 & h_3 & \cdots & h_n \\ h_2 & h_3 & h_4 & \cdots & h_{n+1} \\ h_3 & h_4 & h_5 & \cdots & h_{n+2} \\ \vdots & \vdots & & \ddots & \vdots \\ h_n & h_{n+1} & h_{n+2} & \cdots & h_{2n-1} \end{bmatrix}.$$

Proof: Consider the decomposition $\mathbf{A} = \mathbf{TJ}$, where $\mathbf{T}$ and $\mathbf{J}$ are Toeplitz and reflection matrices, respectively. Together with Example 18, we can conclude that $\mathbf{A}$ is sum of two $\mathbf{DC}\hat{\mathbf{D}}$ matrices, since $\mathbf{A} = \mathbf{D}_1\mathbf{C}_1\mathbf{D}_2\mathbf{J} + \mathbf{D}_3\mathbf{C}_2\mathbf{D}_4\mathbf{J}$ for some two $\mathbf{DCD}$ matrices. Therefore, the $\hat{\mathbf{D}}\mathbf{C}\hat{\mathbf{D}}$-rank is 2. Note that reflection matrix $\mathbf{J}$ is a anti-diagonal matrix:

$$\mathbf{J} = \begin{bmatrix} 0 & \cdots & 0 & 0 & 1 \\ 0 & \cdots & 0 & 1 & 0 \\ 0 & \cdots & 1 & 0 & 0 \\ \vdots & \vdots & & \vdots & \vdots \\ 1 & 0 & 0 & \cdots & 0 \end{bmatrix}.$$

Computing the $\hat{\mathbf{D}}\mathbf{C}\hat{\mathbf{D}}$-rank of a Hankel matrix is a key to study complicated matrix structures and we outline an application as follows: One of the ingredients in numerical computing is studying the products of structured matrix classes. The products of structured matrices do not generally inherit explicit structures of their constituent elements. However, products of Toeplitz $\mathcal{T}$, Hankel $\mathcal{H}$, and Vandermonde $\mathcal{V}$ retain some specific structures. The tight connection for studying the products of the matrix classes is that multiplications of these structured matrices transform the three structures into each other. For example, $\mathcal{TH} = \mathcal{H}$, $\mathcal{HH} = \mathcal{T}$, and $\mathcal{V}^T\mathcal{V} = \mathcal{H}$. The paper [29] proposed that such transformations enable us to extend any efficient algorithm, which computes the inversion of the matrices in any of the classes, into inverting the matrices of other classes. This approach is widely used and recognized to develop

numerically stable Toeplitz solvers. The underlying trick is so called displacement rank (another approach is described in Subsection 2.0.3.). The idea is to transform original structured matrices by their displacements. Of course, displacement operators can be defined by small number of parameters and one can recover the original rapidly. For example, one can improve Toeplitz computations by reducing them into a Cauchy like computations [29]. Thus, studying the $\hat{\mathbf{D}}\mathbf{C}\hat{\mathbf{D}}$-rank of products of the matrix classes are also important.

Example 28. *If* $\mathbf{A}$ *is a element of* $\mathcal{TH}$*,* $\mathcal{HH}$ *or* $\mathcal{V}^T\mathcal{V}$*, then the* $\hat{\mathbf{D}}\mathbf{C}\hat{\mathbf{D}}$*-rank is at most* 2*.*

Proof: Note that $\mathbf{A} = \mathbf{V}^T\mathbf{V} \in \mathcal{V}^T\mathcal{V}$ is a Hankel matrix, where $\mathbf{V}$ is a Vandermonde matrix, i.e.,

$$\mathbf{V} = \begin{bmatrix} 1 & v_1 & v_1^2 & \cdots & v_1^{n-1} \\ 1 & v_2 & v_2^2 & \cdots & v_2^{n-1} \\ 1 & v_3 & v_3^2 & \cdots & v_3^{n-1} \\ \vdots & \vdots & & \ddots & \vdots \\ 1 & v_n & v_n^2 & \cdots & v_n^{n-1} \end{bmatrix}.$$

It follows from Example 27 is that the $\hat{\mathbf{D}}\mathbf{C}\hat{\mathbf{D}}$-rank of $\mathbf{A}$ is at most 2. Similarly, the $\hat{\mathbf{D}}\mathbf{C}\hat{\mathbf{D}}$-ranks of other products of matrix classes can be computed by recalling Example 18.

Example 29. *Computing the* $\hat{\mathbf{D}}\mathbf{C}\hat{\mathbf{D}}$ *ranks of the systems described in Example 7.*

$$4\begin{bmatrix} \mathbf{C} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} = \left[\begin{bmatrix} \mathbf{0} & \mathbf{I} \\ \mathbf{I} & \mathbf{0} \end{bmatrix} + \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & -\mathbf{I} \end{bmatrix}\right]\begin{bmatrix} \mathbf{C}+\mathbf{I} & \mathbf{C}-\mathbf{I} \\ \mathbf{C}-\mathbf{I} & \mathbf{C}+\mathbf{I} \end{bmatrix}\left[\begin{bmatrix} \mathbf{0} & \mathbf{I} \\ \mathbf{I} & \mathbf{0} \end{bmatrix} + \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & -\mathbf{I} \end{bmatrix}\right].$$

It follows that the $\hat{\mathbf{D}}\mathbf{C}\hat{\mathbf{D}}$-rank is 4. In a similar fashion, we can decompose $\begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{C} \end{bmatrix}$ matrix into $\hat{\mathbf{D}}\mathbf{C}\hat{\mathbf{D}}$ factors as well.

4.0.3. Approximating with CD Matrices

There has been lots of attention on approximating a matrix using a product of a circulant matrix and a diagonal matrix, i.e., approximating $\mathbf{A}$ using a product of $\mathbf{CD}$ or $\mathbf{DC}$ matrices. We will illustrate an application arouse in signal processing and one can find some other interesting applications in Section 1. A diffractive optical system or equivalently $\mathbf{A} \in \mathbb{C}^{n\times n}$ can be decomposed into a product of circulant and diagonal matrices. See Subsection 4.0.2 for more details. However, it is not physically feasible to synthesize very large systems in practice, since such factorizations in general require large number of terms as a product, i.e., $\mathbf{A} = \mathbf{D}_1\mathbf{C}_2\mathbf{D}_3\cdots\mathbf{D}_k$ with a large k. Therefore, number of factors k should be compatible with modern technology. Thus, researchers are seeking approximated factorizations which can meet modern data processing requirements as well as technological limitations. Certainly, approximating

with one factor, probably with a circulant matrix, is the simplest one, i.e., $\mathbf{A} \approx \mathbf{C}$. However, the paper [27] proposed to compute a matrix expansion with $\mathbf{CD}$ matrices for approximating a given system or matrix $\mathbf{A} \in \mathbb{C}^{n\times n}$. In this subsection, we will outline some computational methods as well as some applications involving $\mathbf{CD}$ matrices and its approximations.

Approximating $\mathbf{A}$ with a sum of $\mathbf{CD}$ matrices was proposed in [27]:

$$\min_{\mathbf{D}_k,\in\mathcal{D},\mathbf{C}_k\in\mathcal{C}} \left\| \mathbf{A} - \sum_{k=1}^{r} \mathbf{C}_k\mathbf{D}_k \right\|_F, \tag{9}$$

where $\mathcal{D}$ and $\mathcal{C}$ denote diagonal and circulant matrix subspaces respectively.

One important application on approximating with $\mathbf{CD}$ matrices is that it opens a door for designing fast preconditioners for dense Toeplitz related linear systems [30]. For a given $\mathbf{A} \in \mathbb{C}^{n\times n}$, *Algorithm 2* was introduced to find a nearly optimal product $\mathbf{C}_\theta\mathbf{D}$, where $\mathbf{C}_\theta$ is a $\{e^{-i\theta}\}$-circulant matrix.

Algorithm 2: *Computing Optimal CD*

Input:

- $\mathbf{A} \in \mathbb{C}^{n\times n}$.
- Anti-diagonal matrix $\mathbf{P}$.
- $\theta \in [-\pi, \pi]$.

Steps:

1. Compute $\mathbf{P}_\theta = \mathbf{P}(\mathbf{L}_A + e^{-i\theta}\mathbf{U}_A)$.
   - $\mathbf{U}_A-$ Strictly upper triangular matrix whose j^{th} row is j^{th} diagonal of $\mathbf{A}$.
   - $\mathbf{L}_A-$ Lower triangular matrix whose j^{th} row is $(j-n)^{th}$ diagonal of $\mathbf{A}$.
2. Find the best rank-1 approximation $\sigma_1 u_1 v_1^*$ of $\mathbf{P}_\theta$.
3. Compute a diagonal matrix $\mathbf{D}$ by setting diagonal $\sqrt{\sigma_1}\overline{v}_1$.
4. Compute a $\{e^{-i\theta}\}$-circulant matrix $\mathbf{C}_\theta$ having first column $\sqrt{\sigma_1}u_1$.

Definition 25. *$\mathbf{C}_\theta$ matrix is a $\{e^{-i\theta}\}$-circulant matrix if it has a spectral decomposition:*

$$\mathbf{C}_\theta = \mathbf{\Omega}\mathbf{F}^*\mathbf{\Lambda}\mathbf{F}\mathbf{\Omega}^* = \mathbf{\Omega}\mathbf{C}\mathbf{\Omega}^*, \tag{10}$$

where $\mathbf{\Lambda}$ is a diagonal matrix containing eigenvalues of $\mathbf{C}$ and $i^2 = -1$. Further, $\mathbf{\Omega} = \mathbf{diag}(1, e^{-i\theta/n}, \cdots, e^{-i(n-1)\theta/n})$ and $\mathbf{C}$ denotes a circulant matrix.

Application: Approximating Johnson-Lindenstrauss Transforms

Given a higher dimensional finite point set, it is an interesting task to study a low dimensional representation of the set without suffering great distortions, i.e., pairwise distances between the points in a low dimensional space do not change dramatically with respect to their original representation. This distance preserving property often guarantees that a solution computed in the low dimensional space is a good approximation to the problems arouse in the higher dimensional space. Therefore, such a low dimensional representation of the points can be used to speed up algorithms whose running time depend on the dimension of the working space.

To study this problem, the Johnson-Lindenstrauss lemma (JL lemma) is a key. Roughly speaking, the Johnson-Lindenstrauss lemma describes that n-points in a higher dimension can be projected onto a lower dimension size of $k = \mathcal{O}(\epsilon^{-2}log(n))$, while incurring distortions at most $(1 \pm \epsilon)$ in their pairwise distances:

Lemma 26. *Let $x_1, x_2, \cdots, x_n \in \mathbb{R}^d$ be arbitarary points and $k = \mathcal{O}(\epsilon^{-2}log(n))$ be a natural number with $\epsilon \in (0, \frac{1}{2})$. Then there exists a linear map $f : \mathbb{R}^d \to \mathbb{R}^k$ such that*

$$(1-\epsilon)\left\|x_i - x_j\right\|_2^2 \leq \left\|f(x_i) - f(x_j)\right\|_2^2 \leq (1+\epsilon)\left\|x_i - x_j\right\|_2^2 \ \forall\, i, j.$$

Proof: See [4] and references therein.

A computationally challenging problem is developing fast algorithms to evaluate $f(x)$, since the JL lemma requires a dense $k \times d$ random matrix $\mathbf{A}$ and each projection costs $\mathcal{O}(dlog(n))$ operations. An improvement in computational time was achieved by Fast Johnson-Lindenstrauss Transforms (FJLT) proposed by [31]. The idea was based on replacing the dense matrix $\mathbf{A}$ with a preconditioned sparse matrix which has same distance preserving property. Roughly speaking, FJLT is defined as $f(x) = \mathbf{PHD}x$, where $\mathbf{P}$ is generated from independently at random, $\mathbf{H}$ is a Hadamard matrix, and $\mathbf{D}$ is a random diagonal matrix. With this setting, FJLT computes the projections in $\mathcal{O}(dlog(d) + qd\epsilon^{-2}log(n))$ operations with high probability [31].

A notable running time improvement with high probability $\mathcal{O}(dlog(d))$ can be achieved by setting $f(x) = \mathbf{CD}x$, where $\mathbf{C}$ is a partial circulant matrix with independent Bernoulli random variables and $\mathbf{D}$ is a randomly generated diagonal matrix [4].

5. ALGORITHMS FOR APPROXIMATING WITH DCD MATRICES

In this section, we will introduce a greedy algorithmic framework for approximating a given matrix $\mathbf{A} \in \mathbb{R}^{n\times n}$ with sums of $\mathbf{DCD}$ matrices. The greedy method is quite powerful and it may serve as an invitation to develop more numerically reliable algebraic algorithms. To this end, we recall that greedy algorithms in general make the choice that looks optimal at each iteration. In other words, greedy algorithms make a sequence of locally optimal choices in an effort to determine a globally optimal solution. Moreover, these algorithms efficiently find local optimal (or suboptimal) solutions which can be coarser approximations to the global optimal solutions.

With this in mind, suppose that we are equipped with an oracle $\mathcal{P}$. An oracle is a computational unit which can answer the successive questions of the iterative algorithm. Specifically, the oracle $\mathcal{P}$ we suggest, can compute an optimal $\mathbf{DCD}$ matrix which minimizes objective function $g : \mathbb{R}^{3n} \to \mathbb{R}$ defined on parameterized circulant and diagonal matrix subspaces:

$$g(\alpha, c, \lambda) = \left\|\mathbf{A} - \mathbf{D}_1\mathbf{C}\mathbf{D}_2\right\|_F^2 = \sum_{i,j=1}^{n}\sum_{k}(a_{i,j} - \alpha_i c_k \lambda_j)^2, \tag{11}$$

where $\mathbf{D}_1 = \mathbf{diag}(\alpha)$ and $\mathbf{D}_2 = \mathbf{diag}(\lambda) \in \mathcal{D}$, and $\mathbf{C} = \mathbf{circ}(c) \in \mathcal{C}$ with $\alpha, \lambda, c \in \mathbb{R}^{1\times n}$. We denote an optimal $\mathbf{DCD}$ matrix which minimizes problem (11) as $\mathbf{DCD}^\star$. Equivalently, oracle $\mathcal{P}$ will return matrix $\mathbf{DCD}^\star$ for input $\mathbf{A}$, i.e., $\mathcal{P}(\mathbf{A}) = \mathbf{DCD}^\star$. As written in equation (11), we use following notations to represent the elements of parameterized $\mathbf{A} - \mathbf{D}_1\mathbf{C}\mathbf{D}_2$ matrix as below:

$$\begin{bmatrix} a_{11} - \alpha_1 c_1 \lambda_1 & a_{12} - \alpha_1 c_2 \lambda_2 & a_{13} - \alpha_1 c_3 \lambda_3 & \cdots & a_{1n} - \alpha_1 c_n \lambda_n \\ a_{21} - \alpha_2 c_n \lambda_1 & a_{22} - \alpha_2 c_1 \lambda_2 & a_{23} - \alpha_2 c_2 \lambda_3 & \cdots & a_{2n} - \alpha_2 c_{n-1} \lambda_n \\ \vdots & \vdots & & \ddots & \vdots \\ a_{n1} - \alpha_n c_2 \lambda_1 & a_{n2} - \alpha_n c_3 \lambda_2 & a_{n3} - \alpha_n c_4 \lambda_3 & \cdots & a_{nn} - \alpha_n c_1 \lambda_n \end{bmatrix}.$$

Together with the oracle $\mathcal{P}$, we propose *Algorithm 4* which iteratively increases the $\mathbf{DCD}$-rank by one, while improving the approximation quality. Such an incremental $\mathbf{DCD}$-rank one update algorithm is inspired from the optimal approximation property of the SVD for computing rank-r approximation.

One can observe that numerically computing the oracle $\mathcal{P}$ in *Algorithm 4* is a challenging problem. Therefore, our main purpose in this section is developing efficient algorithms for approximating the oracle $\mathcal{P}$. To this end, we seek optimization techniques in the following subsections. Before deriving algorithms, we will explore some properties of the function g.

Let us try to compute an estimate of function g, when diagonal matrices of $\mathbf{DCD}$ are orthogonal matrices.

Proposition 27. *Suppose that* $\mathbf{D}_1$ *and* $\mathbf{D}_2$ *are orthogonal diagonal matrices, then* $\|\mathbf{A} - \mathbf{D}_1\mathbf{C}\mathbf{D}_2\|_F \geq \|\mathbf{A}\|_F - \|\mathbf{\Lambda}\|_F$.

Proof: $\|\mathbf{A} - \mathbf{D}_1\mathbf{C}\mathbf{D}_2\|_F \geq \|\mathbf{A}\|_F - \|\mathbf{D}_1\mathbf{C}\mathbf{D}_2\|_F = \|\mathbf{A}\|_F - \|\mathbf{C}\|_F = \|\mathbf{A}\|_F - \|\mathbf{F}^*\mathbf{\Lambda}\mathbf{F}\|_F$. It follows the proposition, since $\mathbf{F}$ is a unitary matrix and Frobenius norm is unitarily invariant.

Algorithm 4: *Computing DCD expansion*

Input:

- $\mathbf{A} \in \mathbb{R}^{n\times n}$.
- $r \in \mathbb{R}$.

Steps:

1. Set $\xi_0 = \mathbf{A}$.
2. **For** $i = 1 : r$
   - Compute $\mathbf{DCD}^\star = \mathcal{P}(\xi_{i-1})$.
   - Compute $\xi_i = \xi_{i-1} - \mathbf{D}_1\mathbf{C}\mathbf{D}_2^*$ and store $\mathbf{DCD}^\star$.

   end

Next we will analyze the convexity of the function g, since the set of convex function is an important class of functions in optimization. For instance, the global optimal condition of a convex function is guaranteed by Theorem 29. Before proceeding, we first recall the definition of the convex function as below:

Definition 28. *A function $f : \mathbb{R}^n \to \mathbb{R}$ is called convex if $\mathbf{dom}(f)$ is convex and for all $x, y \in \mathbf{dom}(f)$ and $\theta \in [0, 1]$, the following inequality holds:*

$$f(\theta x + (1-\theta)y) \leq \theta f(x) + (1-\theta)f(y),$$

where $\mathbf{dom}(f) = \{x \in \mathbb{R}^n : |f(x)| < \infty\}$.

Theorem 29. *If f is a convex function defined on a convex set $\mathcal{S}$ and $\nabla f(x^*) = 0$, then $x^* \in \mathcal{S}$ is a global minimal.*

Proof: See [32].

Example 30. *A function $g_2 : \mathbb{R}^{3n\times n} \to \mathbb{R}, (\mathbf{D}_1, \mathbf{C}, \mathbf{D}_2) \mapsto \|\mathbf{A} - \mathbf{D}_1\mathbf{C}\mathbf{D}_2\|_F$ is a non-convex function, where $\mathbf{D}_1, \mathbf{D}_1 \in \mathcal{D}$ and $\mathbf{C} \in \mathcal{C}$.*

Proof: Note that $\mathbf{dom}(g_2)$ is not a convex set, since $\mathbf{dom}(g_2)$ is a composite of circulant and diagonal matrix subspaces. From Definition 28, we can conclude that g_2 is not a convex function.

Last example illustrates that minimizing the function g over parameterized circulant and diagonal subspaces is a non-convex optimization problem. Thus, we could not take the advantages of rich theories in convex analysis and computing a global minimum of g over all variables is now computationally challenging. For more information about convex optimization, see [32]. To circumvent the problem, we seek another very powerful but simple algorithmic technique so-called greedy for approximating oracle $\mathcal{P}$. We remark that greedy algorithms provide a natural way of computing better solutions in practice, sometimes with mathematical guarantees. For example, author

in [33] showed that greedy algorithms can yield the better performance in sparse signal processing than other frameworks.

Inspired from them, we explore some information of the function g. Since g is a differentiable function, we can compute partial derivatives of the function g as follows:

$$\frac{\partial g}{\partial \alpha_t} = \sum_{k=1}^{n} 2(a_{tk} - \alpha_t c_k^{'} \lambda_k)(-c_k^{'} \lambda_k), \tag{12}$$

$$\frac{\partial g}{\partial \lambda_t} = \sum_{k=1}^{n} 2(a_{kt} - \alpha_k c_k^{''} \lambda_t)(-c_k^{''} \alpha_k), \tag{13}$$

$$\frac{\partial g}{\partial c_t} = \sum_{k=1}^{n-t+1} -2(\mathbf{Diag}(\mathbf{X}_1)_k \alpha_k \lambda_{k+t-1}) + \sum_{l=1}^{t-1} -2(\mathbf{Diag}(\mathbf{X}_2)_l \alpha_{t+n-l-1} \lambda_l), \tag{14}$$

where $c^{'}$ is $(t-1)$ cyclic shift of first row of $\mathbf{C}$, $c^{''}$ is $(t-1)$ cyclic shift of first column of $\mathbf{C}$, $\mathbf{X}_1$ is submatrix $(\mathbf{A} - \mathbf{D}_1\mathbf{C}\mathbf{D}_2)(1 : n+1-t, t : n)$, and $\mathbf{X}_2$ is submatrix $(\mathbf{A} - \mathbf{D}_1\mathbf{C}\mathbf{D}_2)(n-t+2 : n, 1 : t-1)$.[4]

Moreover, the formulae (12)-(14) can be easily implemented in a matrix computation environment like Matlab. With this information, we will next study gradient based greedy algorithms for approximating oracle $\mathcal{P}$ in the following subsections.

5.0.1. Gradient Descent Approach

Let us start with a simplest first order optimization scheme: gradient descent method. It is well known that negative gradient is a direction of the locally steepest descent of a differentiable function. This is the idea behind gradient descent method whose search direction is negative gradient at a position x. Gradient decent has been extensively used in matrix approximation problems. For example, gradient descent has been used for computing approximative SVD in large scaled systems (e.g., collaborative filtering [34]). Moreover, paper [35] provides a proof of global convergence of gradient search for low-rank matrix approximation: To prove the convergence, authors considered an equivalent optimization problem to low rank approximation problem, defined on the Grassmann manifold equipped with the so-called Fubini–Study distance. In this setting, all suboptimal points are antipodal to global optimal [35]. Thus, gradient descent paths for most of the starting points on the manifold are monotonically decreasing. For sparse matrix approximation using gradient descent like algorithms, see [36].

In our context, we propose *Algorithm 5* which returns a suboptimal solution $(\alpha, c, \lambda)^{\star}$. For stopping criteria, we check a condition $\|\nabla g(x)\|_2 \leq \epsilon$ in each iteration.

[4] We denote $\mathbf{Diag}(\mathbf{A})$ as the diagonal of matrix $\mathbf{A}$.

Computational Cost

When computing gradient of g, we require totally $9n^2$ multiplications at a single iteration, since each computation of $\frac{\partial g}{\partial \alpha_t}$, $\frac{\partial g}{\partial \lambda_t}$, and $\frac{\partial g}{\partial c_t}$ needs $3n$ multiplications for $t = 1, \ldots, n$. Similarly, we require $3(2n-1)n$ total additions, since each computation of $\frac{\partial g}{\partial \alpha_t}$, $\frac{\partial g}{\partial \lambda_t}$, and $\frac{\partial g}{\partial c_t}$ needs $(2n-1)$ additions for $t = 1, \ldots, n$.

Algorithm 5: *Gradient Descent Method*

Input:

- $\mathbf{A} \in \mathbb{R}^{n \times n}$.
- Given initial point $x = (\alpha, c, \lambda) \in \mathbb{R}^{3n}$.
- Parameters t and ϵ.

Steps:

1. Compute $\nabla g(x)$.
2. $x = x - t\nabla g(x)$.
3. Go to Step 1 if $\|\nabla g(x)\|_2 \geq \epsilon$, otherwise exit.

5.0.2. Alternating Minimization Approach

A main drawback of the gradient descent approach is that it could be slow in convergence and it equivalently costs lots of iterations. To find a suboptimal point efficiently, we study an alternating minimization algorithm (AM) to approximate oracle $\mathcal{P}$. Simplicity and iterative nature of such algorithms allow to its applications in bioinformatics, information theory, and finance.

Roughly speaking, the alternating minimization framework can be explained as follows: Given the sets $\mathcal{A}, \mathcal{B}, \mathcal{C}$ and the function $g : \mathcal{A} \times \mathcal{B} \times \mathcal{C} \to \mathbb{R}$, our problem is minimizing g over $\mathcal{A} \times \mathcal{B} \times \mathcal{C}$, i.e.,

$$\min_{\alpha,c,\lambda \in \mathcal{A} \times \mathcal{B} \times \mathcal{C}} g(\alpha, c, \lambda).$$

Often, it is computationally hard to solve such a problem simultaneously over all variables; see Example 30. However, one can sometimes alternate the minimization over the variables for computing a suboptimal solution fast. That is, one can minimize g over one variables, say $\mathcal{A}$, while freezing other variables and etc. Of course, one can follow this procedure until, the algorithm converges to a some suboptimal point. Thus, alternating algorithms solve a sequence of probably simple optimization problems over one variable. Some elementary results for alternating algoritms were discussed in seminal work [37]. Another advantage of such a computational framework is that individual variable update is computationally much cheaper than batch updates.

Remarkably, the alternate minimization algorithm provides that each subproblem is a convex problem as follows:

Proposition 30. *Consider a function* $g_2 : \mathbb{R}^{3n\times n} \to \mathbb{R}$ *defined as follows:* $g_2(\mathbf{D_1}, \mathbf{C}, \mathbf{D_2}) = \|\mathbf{A} - \mathbf{D}_1\mathbf{C}\mathbf{D}_2\|_F$. *Then, alternatively minimizing* g_2 *over diagonal and circulant matrix subspaces, is a convex optimization problem.*

Proof: Suppose $\mathbf{D}_1$ and $\mathbf{D}_2$ are freezed variables. From Definition 28, g_2 is a convex function over $\mathcal{C}$:

$$\begin{aligned}\|\mathbf{A} - \mathbf{D}_1(\alpha\mathbf{C}_1 + (1-\alpha)\mathbf{C}_2)\mathbf{D}_2\|_F &= \|\alpha\mathbf{A} - \alpha\mathbf{D}_1\mathbf{C}_1\mathbf{D}_2 + (1-\alpha)\mathbf{A} - (1-\alpha)\mathbf{D}_1\mathbf{C}_2\mathbf{D}_2\|_F \\ &\leq \alpha\|\mathbf{A} - \mathbf{D}_1\mathbf{C}_1\mathbf{D}_2\|_F + (1-\alpha)\|\mathbf{A} - \mathbf{D}_1\mathbf{C}_2\mathbf{D}_2\|_F .\end{aligned}$$

Note that $\mathcal{C}$ is a subspace. Thus, minimizing g_2 over $\mathcal{C}$ is a convex problem. Similar steps can be followed for other variables $\mathbf{D}_1$ and $\mathbf{D}_2$ over $\mathcal{D}$ to prove the result.

We remark that $g_1 : \mathbb{R}^{3n} \to \mathbb{R}^{3n\times n}, (\alpha, c, \lambda) \mapsto (\mathbf{D}_1, \mathbf{C}, \mathbf{D}_2)$ is a convex function over alternating the variables α, c, and λ. In this setting, it follows that minimizing $g = g_3 \circ g_2 \circ g_1$ alternatively is a convex optimization problem, where $g_3 : \mathbb{R} \to \mathbb{R}, x \mapsto x^2$. In this framework, a sequence of convex optimization subproblems needs to be solved and thus may yield the convergence to a suboptimal solution. Remarkably, one can solve a convex problem efficiently and reliably with modern mathematical subroutines.

Using Theorem 29, we can derive following alternating minimization algorithm as depicted in *Algorithm 6*. For convergence of alternating algorithms for non-convex optimization problems, see [38].

Algorithm 6: *Alternating Minimization*

Input:

- $\mathbf{A} \in \mathbb{R}^{n\times n}$.
- Given initial point $x = (\alpha, c, \lambda) \in \mathbb{R}^{3n}$.

Steps:

Repeat

1. **For** $t = 1 : n$
   $\alpha_t = \sum_{k=1}^{n} a_{tk}c_k^{'}\lambda_k / \sum_{k=1}^{n}(c_k^{'}\lambda_k)^2$
   end
2. **For** $t = 1 : n$
   $\mathbf{A}_1 = \mathbf{A}(1 : n+1-t, t : n)$ and $\mathbf{A}_2 = \mathbf{A}(n-t+2 : n, 1 : t-1)$
   $N_1 = \sum_{k=1}^{n-t+1}(\alpha_k\lambda_{k+t-1})^2 + \sum_{l=1}^{t-1}(\alpha_{t+n-l-1}\lambda_l)^2$
   $c_t = \big(\sum_{k=1}^{n-t+1}\mathbf{Diag}(\mathbf{A}_1)_k\alpha_k\lambda_{k+t-1} + \sum_{l=1}^{t-1}\mathbf{Diag}(\mathbf{A}_2)_l\alpha_{t+n-l-1}\lambda_l\big)/N_1$
   end
3. **For** $t = 1 : n$
   $\lambda_t = \sum_{k=1}^{n} a_{tk}c_k^{''}\alpha_k / \sum_{k=1}^{n}(c_k^{''}\alpha_k)^2$
   end

Until converge

Computational Cost

When computing optimal parameters α, c and λ, we require totally $9n^2$ multiplications at a single iteration, since each computation of α_t, c_t and λ_t requires $3n$ multiplications for $t = 1, \ldots, n$. Similarly, we require $6(n-1)n$ total additions, since for each computation of α_t, c_t and λ_t needs $2(n-1)$ additions for $t = 1, \ldots, n$.

5.0.3. Gauss-Newton Approach

We begin our studies in this subsection by rearranging equation (11) into the following form:

$$g(\alpha, c, \lambda) = \sum_{i,j=1}^{n} \sum_{k} (a_{i,j} - \alpha_i c_k \lambda_j)^2 = \sum_{l=1}^{n^2} f_l(\alpha, c, \lambda)^2 = \|f(x)\|_2^2,$$

where $f_l(\alpha, c, \lambda) = a_{i,j} - \alpha_i c_k \lambda_j$ and $f = (f_1(x), f_2(x), \ldots, f_{n^2}(x))^T$ with $x = (\alpha, c, \lambda) \in \mathbb{R}^{3n}$. Now the challenge is developing an efficient algorithm to minimize $\|f(x)\|_2^2$ or equivalently minimize the sum of squares of f_l's. To this end, this is a variable translation of our problem (11), minimizing g over α, c and λ, into a least square like problem. We can outline the procedure as follows:

First note that f is a nonlinear function and thus developing an algorithm to compute a suboptimal point is computationally hard. To circumvent this problem, we can use Taylor's first order approximation to linearize the function $f : \mathbb{R}^{3n} \to \mathbb{R}^{n^2}$ around a small neighbourhood of $x_k \in \mathbb{R}^{3n}$ as follows:

$$h(x, x_k) \approx f(x_k) + \mathbf{J}(x_k)(x - x_k),$$

where $\mathbf{J}(x_k)$ is the Jacobian of f at x_k.

For $n = 3$, structure of $\mathbf{J}(x)^T$ of $f : \mathbb{R}^{3*3} \to \mathbb{R}^{3^2}$ can be depicted as follows:

$$\left(\begin{array}{ccc|ccc|ccc}
-c_1\lambda_1 & -c_2\lambda_2 & -c_3\lambda_3 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & -c_3\lambda_1 & -c_1\lambda_2 & -c_2\lambda_3 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & -c_2\lambda_1 & -c_3\lambda_2 & -c_1\lambda_3 \\
\hline
-\alpha_1\lambda_1 & 0 & 0 & 0 & -\alpha_2\lambda_2 & 0 & 0 & 0 & -\alpha_3\lambda_3 \\
0 & -\alpha_1\lambda_2 & 0 & 0 & 0 & -\alpha_2\lambda_3 & -\alpha_3\lambda_1 & 0 & 0 \\
0 & 0 & -\alpha_1\lambda_3 & -\alpha_2\lambda_1 & 0 & 0 & 0 & -\alpha_3\lambda_2 & 0 \\
\hline
-c_1\alpha_1 & 0 & 0 & -c_3\alpha_2 & 0 & 0 & -c_2\alpha_3 & 0 & 0 \\
0 & -c_2\alpha_1 & 0 & 0 & -c_1\alpha_2 & 0 & 0 & -c_3\alpha_3 & 0 \\
0 & 0 & -c_3\alpha_1 & 0 & 0 & -c_2\alpha_2 & 0 & 0 & -c_1\alpha_3
\end{array}\right)$$

The idea behind Gauss-Newton approach is basically minimizing $\|h(x, x_k)\|_2^2$ over x. Of course, new problem is much easier to solve, since $h(x, x_k) \approx f(x_k) + \mathbf{J}(x_k)(x - x_k)$ is a approximative system of linear equations. If $\mathbf{J}(x_k)$ is full rank, one can compute following iteration by setting $\nabla h(x) = 0$, i.e., computing the least squares solution at x_k:

$$x_{k+1} = x_k - (\mathbf{J}(x_k)^* \mathbf{J}(x_k))^{-1} \mathbf{J}(x_k)^* f(x_k).$$

Typically, the Gauss-Newton method converges much faster than gradient descent method [32]. A main drawback of the Gauss-Newton approach is that computing the iteration when $\mathbf{J}(x_k)$ is not full rank. In other words, we could not update the iteration when columns of $\mathbf{J}(x)$ are linearly dependent, since $\mathbf{J}(x)^T\mathbf{J}(x)$ is a singular matrix. However, a robust solution can be proposed as a minimization to the following system:

$$\|h(x, x_k)\|_2^2 + \mu_k \|x - x_k\|_2^2, \tag{15}$$

where μ_k is a positive scalar.

Solving equation (15) yields the following update:

$$x_{k+1} = x_k - (\mathbf{J}(x_k)^*\mathbf{J}(x_k) + \mu_k\mathbf{I}_k)^{-1}\mathbf{J}(x_k)^* f(x_k). \tag{16}$$

This update (16) is referred as *Levenberg– Marquardt* (LM) update. For more information about this algorithm, see [32] and references therein. We adapt this idea for computing oracle $\mathcal{P}$ approximately and we highlight its basic steps in *Algorithm 7*.

***Algorithm 7**: Levenberg–Marquardt Method*

Input:

- $\mathbf{A} \in \mathbb{R}^{n\times n}$.
- Given initial point $x_0 \in \mathbb{R}^{3n}$.
- Given scalars $0 < \beta_1 < 1 < \beta_2$ and a positive scalar μ_0.

Steps:

Repeat

1. Compute $\mathbf{J}(x_i)$
2. Compute: $\hat{x} = x_i - (\mathbf{J}(x_i)^*\mathbf{J}(x_i) + \mu_i\mathbf{I})^{-1}\mathbf{J}(x_i)^* f(x_i)$
3. Update x_{i+1} and μ_{i+1} as follows:
   - If $\|f(\hat{x})\|_2^2 < \|f(x_i)\|_2^2$, then $x_{i+1} = \hat{x}$ and $\mu_{i+1} = \beta_1\mu_i$
   - Otherwise $x_{i+1} = x_i$ and $\mu_{i+1} = \beta_2\mu_i$

Until converge

Computational Cost

Computing $\mathbf{J}(x_k)$ costs $3n^2$ multiplications. Note that least squares solution can be numerically computed by QR factorization: Suppose that $\mathbf{J}(x_k) = \mathbf{QR}$. Then, $(\mathbf{J}(x_k)^T\mathbf{J}(x_k))^{-1}\mathbf{J}(x_k)^T = ((\mathbf{QR})^T\mathbf{QR})^{-1}(\mathbf{QR})^T = \mathbf{R}^{-1}\mathbf{Q}^T$. QR factorization requires $\mathcal{O}(9n^4)$ operations.

6. EXPERIMENTS: PROOF OF CONCEPT

In this section, we present an experimental proof of concept on approximating a given matrix with $\mathbf{DCD}$ expansions. In other words, the goal is essentially to present some practical examples that can demonstrate the potentials of $\mathbf{DCD}$ expansions on matrix approximation problems. We perform two experiments: We will first empirically analyze the proposed algorithms in previous section using a synthetic data set. Our main objective in this experiment is comparing errors $\|\mathbf{A}-\mathbf{DCD}^{\star}\|_F$ with different $\mathbf{DCD}^{\star}$ suboptimal matrices returned by proposed algorithms. Next, we will numerically evaluate the $\mathbf{DCD}$ expansion obtained from our main algorithm, *Algorithm 4*, with an application to image reconstruction. As a benchmark, we will compare our simulation results with vanilla SVD. Finally, we will comment on initial point selection for algorithms and orthogonality of numerically computed $\mathbf{DCD}$ elements in the expansion. Let us keep in mind, though, that these experiments are very preliminary and finally we remark that the SVD approximations are optimal whereas the greedy methods in general give an upper bound. So we do not actually know how good the idea proposed is. It might be very good, since it works well already in the greedy settings. In other words, the SVD has had around 50 years head start!

Experiment: $\mathbf{DCD}$ rank-one Approximations for Synthetic Data

In this experiment, we are investigating the $\mathbf{DCD}$-rank one approximations to a given square matrix $\mathbf{A}$. One important question we need to answer is that what is, to our mind, the best proposed algorithm to compute the $\mathbf{DCD}$-rank one approximation for a given matrix $\mathbf{A}$? Or what is best proposed algorithm to approximate oracle $\mathcal{P}$? To answer this question, we design following simple experiment: We prepare a synthetic data set consisting square matrices and compare $\mathbf{DCD}$-rank one approximation errors associated to the proposed algorithms.

Preparing Data set: In Matlab environment, we randomly generate set S_1 consisting 20 matrices with dimensions of 100×100 using the command **randi([1 10],100,100)**. With this input, randi returns a 100×100 matrix whose entries are pseudorandom integers between 1 and 10 drawn from the discrete uniform distribution on $[1, 10]$. Thus, $S_1 = \{\mathbf{A}_i \in \mathbb{R}^{100\times 100}, i = 1, \ldots, 20\}$. We set $\gamma \in \{$Gradient Descent, AM, LM$\}$.

Discussion: For each $\mathbf{A}_i \in S_1$ and for each algorithm γ, we compute the approximation error $e_i^{\gamma} = \|\mathbf{A}_i - \mathbf{DCD}^{\star}\|_F$, where $\mathbf{DCD}^{\star}$ is the approximant returned by γ for a given $\mathbf{A}_i$. Next we compute average error $e^{\gamma} = \frac{1}{20}\sum_{i=1}^{20} e_i^{\gamma}$ for each algorithm and we report them in Figure 4. Moreover, Figure 4 shows average error information of algorithms with respect to their number of iterations, since the performance of the algorithms also depend on their complexities. For example, on average gradient descent method requires more than 30 iterations to compute $\mathbf{DCD}$ approximants which yield better approximations than the SVD rank-one approximations. However, gradient descent method does not converge to a point. Remarkably, AM algorithm converges to a suboptimal point much faster (around 5 iterations) than other proposed algorithms and its $\mathbf{DCD}$ approximation is better than SVD rank-one approximation. In other words, we can employ AM algorithm for

faster matrix approximations with $\mathcal{O}(n^2)$ operations. For more information about the SVD and its computational cost, see Section 3. Next, we turn our attention to LM algorithm. It is the best proposed algorithm with better approximation errors. However, on average more iterations (around 10 iterations) are required than AM algorithm and we can observe that it always converges to a some optimal point. Roughly speaking, all proposed algorithms are able to compute better approximants than the SVD rank-one matrix and these approximants could not be able to perform better than the SVD rank-two approximations. Finally, we remark that initial points for these algorithms were generated randomly by Matlab **randi** command.

Number of Iterations	Average Error of Gradient Descent Method	Average Error of AM Method	Average Error of LM Method
1	619.59	389.30	619.15
5	618.41	283.15	312.86
10	614.45	283.65	282.85
15	602.15	283.35	282.65
20	557.04	283.28	282.45
30	284.10	283.15	282.55
40	283.56	283.05	282.47

Figure 4. Average error information. SVD rank-one and rank-two average approximation errors are 284.24 and 278.85, respectively.

Experiment: Approximating Images using DCD Expansion

In this experiment, we are interested in approximating grayscale natural images using sums of $\mathbf{DCD}$ matrices. To this end, we create a small database S_2 which consists of natural images arbitrarily chosen from freely available standard database [39]. We mean by *natural image* is that a grayscale value of a pixel does not predominate grayscale values of its neighbours, i.e., grayscale values do not change significantly in its neighborhood. This image database [39] has images taken around Groningen, Holland, in town as well as in the surrounding countryside.

Preparing Data set: We arbitrary choose 100 images from the database and we then resize each image into 1000 pixel $\times 1000$ pixel grayscale images, where original images are size of 1536 pixel $\times 1024$ pixel. The resultant database S_2 has roughly 43 images of town and 57 images of countryside. Of course, resized grayscale image $\mathbf{X}_i$ can be considered as 1000×1000 matrix whose entries can take 256 possible values and $S_2 = \{\mathbf{X}_i : 0 \leq \mathbf{X}_i(j,k) \leq 255, i = 1, \ldots, 100\}$.

Discussion: Our purpose is numerically reconstructing a given image $\mathbf{X}_i$ using a $\mathbf{DCD}$ expansion. To this end, we utilize the proposed computational framework *Algorithm 4* in last section. An elementary computational unit in *Algorithm 4*, oracle $\mathcal{P}$, can be approximated by AM algorithm as discussed in last experiment. Putting everything together, we can have numerical framework to compute finite $\mathbf{DCD}$ expansions. For our simulation purposes, we set the length of the $\mathbf{DCD}$ expansion to 50. In summary, we are going to fit a $\mathbf{DCD}$ expansion of length 50 which approximates a given image

(a) **DCD**-rank 2 (b) **DCD**-rank 5 (c) **DCD**-rank 10

(d) **DCD**-rank 20 (e) **DCD**-rank 50 (f) Original image

Figure 5. Approximating original images at different **DCD**-ranks.

(a) rank 2 (b) rank 5 (c) rank 10

(d) rank 20 (e) rank 50 (f) Original image

Figure 6. Approximating original images at different ranks.

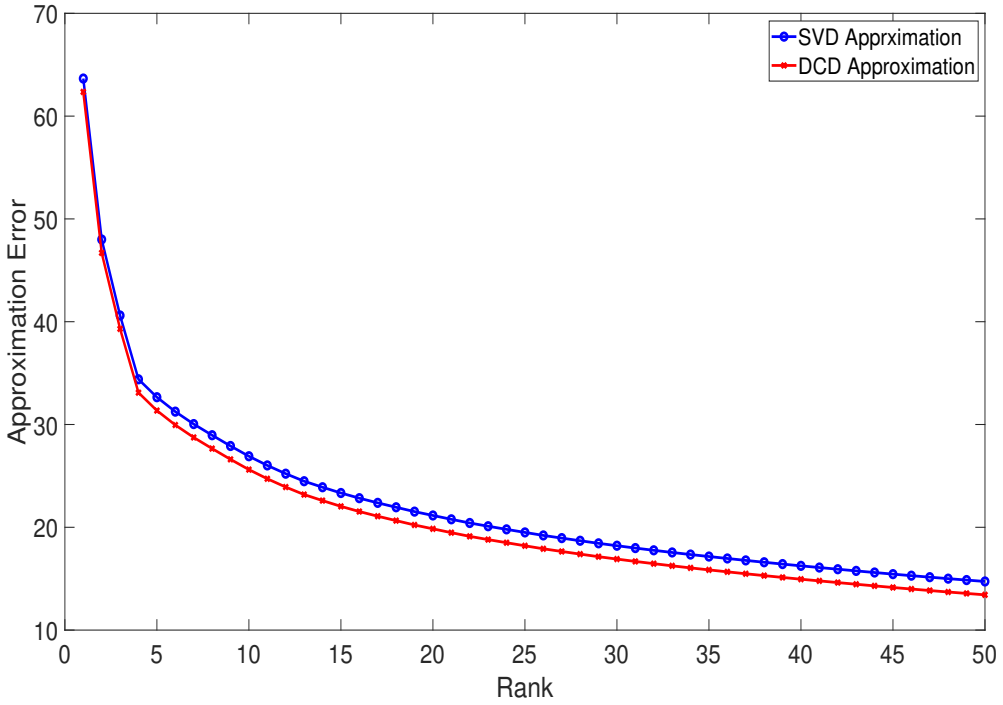

Figure 7. Average approximation errors: $\mathbf{DCD}$ expansion and the SVD.

$\mathbf{X}_i$. With this setting, we have $\mathbf{X}_i \approx \sum_{j=1}^{50}(\mathbf{DCD})_j$, where $(\mathbf{DCD})_j$ matrices are computed by AM algorithm. We then compute the approximation error $e_{ik} = \left\|\mathbf{X}_i - \sum_{j=1}^{k}(\mathbf{DCD})_j\right\|_F$ for each $k = 1, \ldots, 50$ and average approximation error $e_k = \frac{1}{100}\sum_{i=1}^{100} e_{ik}$ for $k = 1, \ldots, 50$.

As a benchmark, we consider the SVD, a rank-one expansion, and its approximations to evaluate our $\mathbf{DCD}$ simulation results. From SVD, we compute the best rank-k approximations for each image and $k = 1, \ldots, 50$. We then compute their average approximation errors. We remark that average rank of $\mathbf{X}_i$ is 998.24, where we use Matlab **rank**$(\mathbf{X}_i)$ command to determine the rank of $\mathbf{X}_i \in S_2$, numerically.

We illustrate an example of reconstructing a original image at different $\mathbf{DCD}$-ranks in Figure 5 and the SVD approximations of the original image are shown in Figure 6. Comparing these Figures, we can notice that low $\mathbf{DCD}$-rank images (see Figure 5(a)-(c)) contain more information than the SVD low rank approximations (see Figure 6(a)-(c)). However, we could not find considerable differences among the approximations with higher ranks (see Figure 5(d)-(e) and Figure 6(d)-(e)). Moreover, we depict average approximation errors of two frameworks in Figure 7. We can observe that $\mathbf{DCD}$ approximations on average yield better approximations comparing with the SVD low rank approximations.

We close this experiment by examining the interplay between initial point selection methods: In this experiment, we set initial points for AM algorithm by computing the SVD rank-one approximation for each residue that we obtain from *Algorithm 4*. We remark that the initial points play a critical role for computing better $\mathbf{DCD}$ approximations. For example, if we run the experiment with random initializations, it

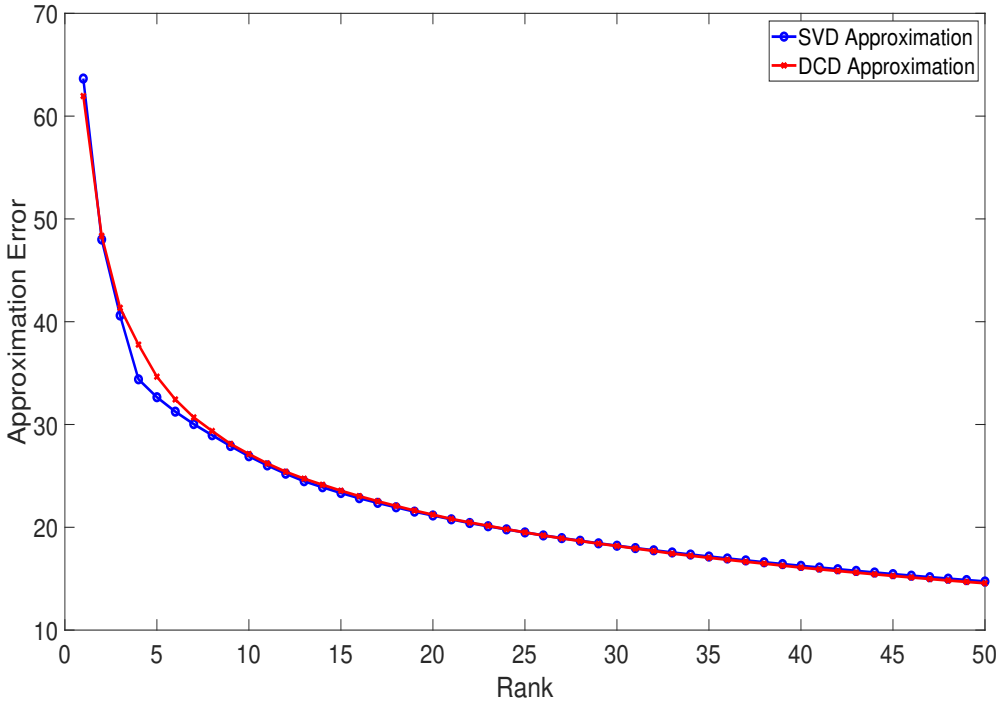

Figure 8. Average approximation errors: $\mathbf{DCD}$ expansion with random initialization and the SVD.

produces average approximation errors with $\mathbf{DCD}$ expansions as shown in Figure 8. Let us discuss this topic in some detail next.

6.0.1. Initial Point Selection Methods

We recall that function $g : \mathbb{R}^{3n} \to \mathbb{R}$ defined in equation (11) is non-convex and our iterative algorithms are based on the gradient information of g. Thus, our algorithms are in general limited to converge to suboptimal points due to non-convexity of the function g and they are therefore highly dependent on their initialization. It follows that selecting initial points for these algorithms are very important in convergence of these algorithm to a better suboptimal points. With this in mind, we next outline some mechanisms to generate an initial $\mathbf{D}_1\mathbf{C}\mathbf{D}_2$ matrix parameterized by α, c and λ:

1. *Selecting Random point*: A simplest approach is generating required parametrization, α_0, c_0, and λ_0, randomly. For example, one can use **randi** command in Matlab environment to generate random vectors and then one can arrange those random vectors to have appropriate matrix structures.

2. *SVD based approach*: Recall that rank-one matrix is a $\mathbf{DCD}$ matrix, see Example 8. Thus, we have $xy^* = \mathbf{D}_1\mathbf{C}\mathbf{D}_2$, where $\mathbf{D}_1, \mathbf{C}$, and $\mathbf{D}_2$ matrices are parametrized by $x, \mathbf{1}$, and y vectors, respectively. We can adapt this idea for generating the initial parameters as follows: It is well known that for a given square matrix $\mathbf{A}$, we can compute best rank-one approximation by invoking the SVD, i.e., $\mathbf{A} \approx \sigma_1 u_1 v_1^*$, where u_1, v_1, and σ_1 are singular vectors and

largest singular value of $\mathbf{A}$, respectively. This representation yields a way of computing required initial parameters via singular values and singular vectors of $\mathbf{A}$ as described earlier.

3. $\mathbf{CD}$ *approximation based approach*: Note that in the absence of $\mathbf{D}_1$, the problem in equation (11) becomes minimizing $\|\mathbf{A} - \mathbf{CD}\|_F^2$ over circulant and diagonal matrix subspaces. This problem can be solved by the shuffling trick, through finding the rank-one approximation to a shuffled $\mathbf{A}$; for more information, see *Algorithm 2* and [27]. We can use this approach to generate an initial point for our algorithms.

   Suppose that we have

   $$\mathbf{D}_1\mathbf{A} \approx \mathbf{CD}_2,$$

   for some nonsingular $\mathbf{D}_1$. Of course, we can compute such a approximation from *SVD based approach*. By utilizing alternating minimization, one can solve $\mathbf{D}_1$ while freezing $\mathbf{CD}_2$ and then one can solve $\mathbf{CD}_2$ while freezing $\mathbf{D}_1$. After some iterations, we may use $\mathbf{D}_1^{-1}\mathbf{CD}_2$ as our initial point for our algorithms.

We report preliminary simulation results based on selecting initial $\mathbf{DCD}$ matrices or equivalently parameters as described above:

- We noticed that gradient descent algorithm, most of the times, did not converge in the case of selecting initial points randomly. *SVD based approach* and $\mathbf{CD}$ *approximation based approach* almost all the times produced the same approximation errors for gradient descent approach. But final approximant of circulant and diagonal matrices differed in those two approaches.

- We noticed that selecting initial points with *SVD based approach* and $\mathbf{CD}$ *approximation based approach* produced approximatively the same errors for AM method. Further, we noticed that the approximants of circulant and diagonal matrices differed in those approaches. AM algorithm with random initialization converged to a point, but with higher error values. Similar observations were hold for LM algorithm as well.

6.0.2. Some Comments on Orthogonality of DCD Approximation

In this subsection, we are going to empirically study the *orthogonality* of the elements in a $\mathbf{DCD}$ expansion. For instance, orthogonal $\mathbf{DCD}$ bases which span whole $\mathbb{C}^{n\times n}$, may provide a natural way of representing any matrix $\mathbf{A} \in \mathbb{C}^{n\times n}$. Another application is that a k-closer approximation to a given $\mathbf{A}$ may be computed by finding k-largest $\mathbf{DCD}$ matrices, where $k \leq n$. Expansions with similar characteristics give the foundations to many applications; see Fourier expansion, the SVD, and their applications.

We recall that the SVD approximation of $\mathbf{A}$ with fixed k, i.e., best rank-k approximation, we consider following:

$$\mathbf{A} \approx \sum_{i=1}^{k} \sigma_i u_i v_i^*,$$

where u_i and v_i for $i = 1, \ldots, k$ can be freely chosen singular vectors of $\mathbf{A}$. For more information about the SVD, see Section 3. Specifically, it is turnout that approximation with rank-one matrices $u_i v_i^*$ are orthogonal, i.e., $\mathbf{Tr}(u_j v_j^* (u_l v_l^*)^*) = 0$, where $\mathbf{Tr}(\mathbf{A}\mathbf{B}^*)$ is an inner product in $\mathbb{C}^{n\times n}$. Inspired from these factors, it is interesting to consider following $\mathbf{DCD}$ approximation problem and check numerically the orthogonality of the $\mathbf{DCD}$ elements:

$$\min_{\mathbf{D}_1,\mathbf{D}_2,\hat{\mathbf{D}}_1,\hat{\mathbf{D}}_2\in\mathcal{D},\mathbf{C}_1,\hat{\mathbf{C}}_1\in\mathcal{C}} \left\|\mathbf{A} - \mathbf{D}_1\mathbf{C}_1\mathbf{D}_2 - \hat{\mathbf{D}}_1\hat{\mathbf{C}}_1\hat{\mathbf{D}}_2\right\|_F^2, \tag{17}$$

where $\mathcal{D}$ and $\mathcal{C}$ are diagonal and circulant matrix subspaces, respectively.

By setting initial points from the SVD, i.e., set initial $\mathbf{D}_1\mathbf{C}_1\mathbf{D}_2 = u_1 v_1^*$ and $\hat{\mathbf{D}}_1\hat{\mathbf{C}}_1\hat{\mathbf{D}}_2 = u_2 v_2^*$ as $\mathbf{DCD}$ matrices like Example 8, we can devise following alternating iteration to minimize (17):

$$\begin{aligned} &\min_{\mathbf{D}_1,\mathbf{D}_2\in\mathcal{D},\mathbf{C}_1\in\mathcal{C}} \left\|\mathbf{A} - \mathbf{D}_1\mathbf{C}_1\mathbf{D}_2 - \hat{\mathbf{D}}_1\hat{\mathbf{C}}_1\hat{\mathbf{D}}_2\right\|_F^2, \\ &\min_{\hat{\mathbf{D}}_1,\hat{\mathbf{D}}_2\in\mathcal{D},\hat{\mathbf{C}}_1\in\mathcal{C}} \left\|\mathbf{A} - \mathbf{D}_1\mathbf{C}_1\mathbf{D}_2 - \hat{\mathbf{D}}_1\hat{\mathbf{C}}_1\hat{\mathbf{D}}_2\right\|_F^2. \end{aligned} \tag{18}$$

To minimize (18), we utilize gradient descent method as discussed in our first experiment. For our simulation purposes, we randomly generate a set $\{\mathbf{A}_i \in \mathbb{R}^{100\times 100}, i = 1, \ldots, 10\}$ in Matlab environment using command **randi([0 10],100,100)** and compute inner products between optimal $\mathbf{DCD}^*$ matrices returned by the gradient descent. On average, it is 0.0074 whereas inner product between the SVD rank-one matrices on average is $2.14 \times e^{-14}$. Hence based on numerical experiments, it appears that the terms in the $\mathbf{DCD}$ expansions are not orthogonal. We will leave a rigorous analysis of the orthogonality of $\mathbf{DCD}$ approximants as a future research.

7. CONCLUSION AND FUTURE DIRECTIONS

The last seven decades, dating back to the World War **II**, have witnessed a number of astonishing developments at the intersection of linear algebra and applications in computer science. Specifically, new developments including matrix factorizations, approximations, and expansions evolve with influx of ideas coming from diverse areas in mathematics. We covered some of these results in Section 2 and we conclude that it is very challenging to develop algorithms for matrix decompositions which yield the factors with proper computational properties.

Inspired from optical signal processing, there has been a lot of attention towards approximating a given matrix using products of circulant and diagonal matrices. With these motivations, our exposition mainly concerned on studying products of circulant and diagonal matrices in matrix factorings, approximatings, and expandings. Computationally, we developed a structured matrix expansion whose elements are $\mathbf{DCD}$ matrices, i.e., a product of diagonal matrix, circulant matrix, and another diagonal matrix. A mathematical notion related to the expansion, denoted as $\mathbf{DCD}$-rank, was introduced and we computed the $\mathbf{DCD}$-rank of some structured matrices. For instance, Toeplitz matrix is a sum of two $\mathbf{DCD}$ matrices. Moreover, we outlined an random projection method called "Johnson-Lindenstrauss Transforms" which can gain speed ups once constructed approximately using a $\mathbf{CD}$ matrix, i.e., a product of a partially circulant matrix and a diagonal matrix.

Next, our challenge was to devise efficient algorithms for approximating given $\mathbf{A}$ with sums of $\mathbf{DCD}$ matrices. Inspired from the optimality of the singular value decomposition (SVD) and the rank-r approximation, we proposed a greedy framework, *Algorithm 4*, which iteratively increases the $\mathbf{DCD}$-rank by one, while increasing the approximation quality. We performed two preliminary experiments to evaluate the framework numerically. As a benchmark, we compared our simulation results with the SVD. Objective of the first experiment was to compare the $\mathbf{DCD}$-rank one approximation errors for a given matrix $\mathbf{A} \in \mathbb{R}^{n\times n}$, i.e., $\|\mathbf{A} - \mathbf{DCD}\|_F$. Our simulation results suggest that the $\mathbf{DCD}$-rank one approximants yield better performance than the SVD rank-one approximants. Our objective of the second experiment was reconstructing a given grayscale image $\mathbf{X} \in \mathbb{R}^{1000\times 1000}$ using sums of $\mathbf{DCD}$ matrices, i.e., $\left\|\mathbf{X} - \sum_{i=1}^{50}(\mathbf{DCD})_i\right\|_F$. Comparing with the SVD, numerical simulations suggest that sums of $\mathbf{DCD}$ matrices yield better approximation quality for every $i = 1, \ldots, 50$. We remarked that the initial point of the algorithm plays a critical role in convergence of the algorithm to a better suboptimal point. To circumvent this problem, we proposed three mechanisms to generate good initial guesses for the algorithm. However, extensive analysis on convergence and numerical simulations of these methods are left as future research directions.

In the spirit, the study of $\mathbf{DCD}$ matrices shows that $\mathbf{DCD}$ matrices possess a rich structure like tensors and therefore a study in tensor analysis can give essential knowledge to develop our understandings and efficient algorithms.

8. REFERENCES

[1] Donoho D. (2018) Blackboard to bedside: High-dimensional geometry is transforming the mri industry. Notices of the American Mathematical Society 65, pp. 40–44.

[2] Gillis N. & Glineur F. (2011) Low-rank matrix approximation with weights or missing data is np-hard. SIAM Journal on Matrix Analysis and Applications 32(4).

[3] Swayambhoo S. & Haupt J. (2017) Convolution approximations to linear dimensionality reduction operators. ICASP , p. 5885–5889.

[4] Vybíra J. (2011) A variant of the johnson lindenstrauss lemma for circulant matrices. Journal of Functional Analysis , pp. 1096–1105.

[5] Cheng Y., Yu F.X., Feris R.S., Kumar S., Choudhary A. & Chang S. (2015) An exploration of parameter redundancy in deep networks with circulant projections. IEEE International Conference on Computer Vision 4, pp. 2857–2865.

[6] Gillis N. & Glineur F. (2012) On the geometric interpretation of the nonnegative rank. Linear Algebra and its Applications , pp. 2685–2712.

[7] Mond D., Smith J. & Straten D.V. (2003) Stochastic factorizations, sandwiched simplices and the topology of the space of explanations. Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences 459, p. 2821–2845.

[8] Gillis N. (2014) The why and how of nonnegative matrix factorization. Regularization, Optimization, Kernels, and Support Vector Machines , pp. 257–291.

[9] Bosch A.J. (1986) The factorization of a square matrix into two symmetric matrices. The American Mathematical Monthly 93, pp. 462–464.

[10] Horn R.A. & Johnson C.R. (1985) Matrix Analysis. Cambridge University Press, first ed.

[11] Chandrasekeran S., Gu M., Sun X., Xia J. & Zhu J. (2007) A superfast algorithm for toeplitz systems of linear equations. SIAM Journal of Matrix Analysis and Applications 29, pp. 1247–1266.

[12] Ye K. & Lim L. (2016) Every matrix is a product of toeplitz matrices. Foundation of Computational Mathematics , pp. 577–598.

[13] Pasupathy J. & Damodar R.A. (1992) The gaussian toeplitz matrices. Linear Algebra and its Applications 171, pp. 133–147.

[14] Tropp J.A. (2015) An introduction to matrix concentration inequalities. Foundations and Trends in Machine Learning .

[15] Eisenbud D. (1980) Homological algebra on a complete intersection, with an application to group representations. Transactions of the American Mathematical Society 260, pp. 35–64.

[16] Eckart C. & Young G. (1936) The approximation of one matrix by another of lower rank. Psychometrika 1.

[17] Golub G.. & Kahan W. (1965) Calculating the singular values and pseudo-inverse of a matrix. SIAM Journal of Numerical Analysis 2, pp. 205–224.

[18] Golub G.H. & Loan C.F.V. (2013) Matrix Computations. The Johns Hopkins University, forth ed.

[19] Davis P. (1994) Circulant Matrices. A Wiley-Interscience Pub., John Wiley and Sons, second ed.

[20] Loan C.V. (1992) Computational Frameworks for the Fast Fourier Transform. SIAM.

[21] Strassen V. (1969) Gaussian elimination is not optimal. Numerische Mathematik 13, pp. 354–356.

[22] Strang G. (1986) A proposal for toeplitz matrix calculation. Studies in Applied Mathematics 74, pp. 171–176.

[23] Chan T.F. (1988) An optimal circulant preconditioner for toeplitz systems. SIAM J. Sci. Statist. Comput. 9, pp. 766–771.

[24] Quade J.M., Aagedel H., Beth T. & Schmid M. (1998) Algorithmic design of diffractive optical systems for information processing. Physica D: Nonlinear Phenomena 120, pp. 196–205.

[25] Schmid M., Steinwandt R., Quade J.M., Rötteler M. & Beth T. (2000) Decomposing a matrix into circulant and diagonal factors. Linear Algebra and its Applications 306, pp. 131–143.

[26] Huhtanen M. (2015) Factoring matrices into the product of circulant and diagonal matrices. Journal of Fourier Analysis and Applications 21, pp. 1018–1033.

[27] Huhtanen M. (2008) Approximating ideal diffractive optical systems. Journal of Mathematical Analysis and Applications 345, pp. 53–62.

[28] Bhatia R. (2000) Pinching, trimming, truncating, and averaging of matrices. The American Mathematical Monthly 107, pp. 602–608.

[29] Pan V.Y. (1989) On computations with dense structured matrices. Proc. Intern. Symposium on Symbolic and Algebraic Computation , p. 34–42.

[30] Huhtanen M. (2007) How real is your matrix? Linear Algebra and its Applications 424, pp. 304–319.

[31] Ailon N. & Chazelle B. (2009) The fast johnson-lindenstrauss transform and approximate nearest neighbors. SIAM Journal of Computing 39, pp. 302–322.

[32] Bertsekas D.P. (1999) Nonlinear Programming. Athena Scientific, second ed.

[33] Tropp J.A. (2004) Greed is good: Algorithmic results for sparse approximation. IEEE Transactions on Information Theory 50, pp. 2232–2242.

[34] Paterek A. (2007) Improving regularized singular value decomposition for collaborative filtering. Proc. KDD Cup and Workshop , pp. 5–8.

[35] Pitava R.A., Dai W. & Tirkkonen O. (2015) Convergence of gradient descent for low-rank matrix approximation. IEEE Transactions on Information Theory , pp. 4451–4457.

[36] Journée M., Nesterov Y., Richtárik P. & Sepulchre R. (2010) Generalized power method for sparse principal component analysis. J. Mach. Learn. Res 11, pp. 517–553.

[37] Csiszár I. & Tusnády G. (1984) Information geometry and alternating minimization procedures. Statistics and Decisions: Supplement Issue , pp. 205–237.

[38] Xu Y. & Yin W. (2017) A globally convergent algorithm for nonconvex optimization based on block coordinate update. Journal of Scientific Computing 72, pp. 700–734.

[39] Hateren J.H.V. & Schaaf A.V.D. (1998) Independent component filters of natural images compared with simple cells in primary visual cortex. Proceedings: Biological Sciences 265, pp. 359–366.