Exploiting Kronecker Structures

With applications to optimization problems arising in the field of adaptive optics

P. Varnai

Master of Science Thesis

Delft University of Technology

Delft Center for Systems and Control

Exploiting Kronecker Structures

With applications to optimization problems arising in the field of adaptive optics

Master of Science Thesis

For the degree of Master of Science in Systems and Control at Delft University of Technology

P. Varnai

August 21, 2017

Faculty of Mechanical, Maritime and Materials Engineering (3mE) · Delft University of Technology

Abstract

We study the important mathematical problem of approximating the inverse of low Kronecker-rank matrices in this same form. A traditional alternating least squares (ALS) scheme for solving such problems is presented, and we discuss two efficient solutions to the subproblems arising in the corresponding iterations. The first relies on a least-squares formulation while the second on a gradient-based solution from the literature. The former new approach is slightly less efficient but more robust as it does not involve forming the normal equations. We also advocate for employing a Nesterov-type acceleration in the higher level ALS scheme. Usage of the resulting algorithm is evaluated in the context of approximating inverses in low Kronecker-rank form in order to preserve this structure for its continued exploitation in applications such as the matrix sign iterations and preconditioning linear systems.

Our theoretical study is motivated by two real-life practical applications in the field of adaptive optics (AO). We address each of these in terms of exploiting the Kronecker products featured within their problem formulations.

A minimum variance control scheme of wavefront control for atmospheric turbulence correction leads to a large-scale Kronecker structured constrained least-squares problem whose efficient solution is crucial for real-time implementation. Potential approaches based on the alternating direction method of multipliers (ADMM), projected alternating Barzilai-Borwein (PABB), and active sets (AS) methods are analyzed and compared in the context of exploiting the Kronecker structure of the system matrices using data from a partially realistic numerical study. The PABB approach is shown to be the most competitive, also with respective to alternative solutions exploiting only the sparsity of the system matrices instead of their Kronecker form as well.

The optimal design of a novel wavefront sensor called a sparse aperture mask (SAM) is also addressed in our work. Here Kronecker products appear when propagating wavefronts using the matrix-form of evaluating two dimensional Fourier transforms. The design problem is formulated in terms of a trade-off between the achievable light throughput of the mask and its ability to distinguish so-called Zernike modes that form a basis for the reconstructed wavefront. Two nonlinear optimization approaches for the solution are presented and discussed in terms of exploiting the Kronecker products by evaluating matrix-vector multiplications with them using a large but sparse philosophy. A final framework is proposed to combine the strengths of these in order to tackle the design problem.

Table of Contents

List of Figures

List of Tables

Acknowledgements

I would like to thank my supervisors prof.dr.ir. M. Verhaegen and B. Sinquin for offering the opportunity to tackle exciting challenges in the field of adaptive optics under the course of this thesis. Our discussions have led to many interesting ideas that overall raised the quality of my thesis work.

I would also like to thank the researchers at the High Contrast Imaging Laboratory at Princeton University under Prof. J. Kasdin for quickly accepting me into their group during my short 3-month visit, as well as for the meaningful insights they offered with respect to my work. I am especially grateful for the Justus and Louise van Effen Research Grant which made the research trip to Princeton possible in the first place.

Finally, I am most thankful to all my family and friends who have supported me during the course of writing this thesis work.

Delft, University of Technology P. Varnai
August 21, 2017

Chapter 1

Introduction

Kronecker products serve as matrix compression technique and allow the reduction of system dimensions, which is an important strategy for coping with the large-scale nature of many practical systems. An especially important class of Kronecker structures, referred to as low Kronecker-rank matrices, are composed of a few sums of Kronecker products. They allow representations of increased accuracy; however, the sums destroy many of the favorable properties Kronecker products possess, and the efficient exploitation of this structure presents a major challenge to the field of scientific computing.

Here we consider the problem of approximating the inverse of a low Kronecker-rank matrix with another of such form. Preserving such structures has important applications in the continued exploitation of Kronecker matrix-vector products during many iterative algorithms. In particular, the inverse approximation is useful for preconditioning linear systems and within structure preserving algorithms such as the matrix sign iterations. We conduct a detailed analysis of two possible solutions to this problem, one based on forming the normal equations, and our own least-squares formulation. This latter is shown to be slightly less efficient in exchange for being more numerically robust. We also compare how fully optimizing for the summands of the inverse approximation at once compares to a term-by-term progressive scheme from the literature in the context of both the preconditioning and structure preserving applications.

Our focus on Kronecker products and their exploitation is motivated by two practical problems, namely

(i) *wavefront control for atmospheric turbulence correction*, and

(ii) *the design of a sparse aperture mask for wavefront reconstruction.*

These problems encompass current developments and challenges faced in the field of adaptive optics for ground and space based telescopes, respectively. We present a brief background review, problem formulation, and our solution proposals for each of these problems in the context of exploiting the Kronecker structures within their description. The solution approaches proposed are motivated by the results and observations from the theoretical sections of our work regarding Kronecker products.

1-1 Exploiting Kronecker structures

Kronecker products are generalizations of the tensor (or outer) products of vectors to matrices. They are becoming increasingly prominent in scientific computing and have attracted considerable attention within the past decade due to their potential use in accelerating many practical algorithms in a wide range of application areas, as advocated for in the excellent review by Van Loan [2]. A Kronecker product of the form

$$\mathbf{A} = \mathbf{B} \otimes \mathbf{C} = \begin{bmatrix} b_{11}\mathbf{C} & b_{12}\mathbf{C} & \cdots & b_{1n}\mathbf{C} \\ b_{21}\mathbf{C} & b_{22}\mathbf{C} & \cdots & b_{2n}\mathbf{C} \\ \vdots & \vdots & \ddots & \vdots \\ b_{m1}\mathbf{C} & b_{m2}\mathbf{C} & \cdots & b_{mn}\mathbf{C} \end{bmatrix}$$

allows a compact representation of the large dimensional $\mathbf{A} \in \mathbb{R}^{m^2 \times n^2}$ matrix using the much smaller $\mathbf{B},\ \mathbf{C} \in \mathbb{R}^{m \times n}$ terms. In the context of exploiting Kronecker structures, we mean to operate with these smaller matrices during solution algorithms for gains in computational efficiency.

There are many favorable algebraic properties that Kronecker products possess which allow them to be manipulated in order to develop accelerated solution algorithms for certain structured problems. For example, in the form above, quantities such as the inverse, transpose, and various factorizations of $\mathbf{A}$ will remain Kronecker products and can be expressed by the corresponding inverses, transposes, and factorizations of the terms $\mathbf{B}$ and $\mathbf{C}$. A brief overview of the main properties we will need for our discussions in this thesis work is presented in the Appendix A-1. Many additional useful features of the smaller matrices, such as symmetry, bandedness, orthogonality, and so on, are also preserved for the larger $\mathbf{A}$ matrix; these relations and their proofs, along with methods for approximating matrices by Kronecker products, are summarized in Van Loan and Pitsianis [3].

For simple problems involving Kronecker products, such as the least squares problem

$$\min_{\boldsymbol{x}} \|\boldsymbol{t} - (\mathbf{B} \otimes \mathbf{C})\boldsymbol{x}\|_2 ,$$

the structure of the coefficient matrix can be readily utilized to provide efficient direct solutions using common factorization methods. For this specific example, Fulton and Wu [4] compare the speed and robustness for solutions obtained using the QR, LU, and SVD decompositions of the Kronecker terms. The main lesson from their analysis is that significant gains can be made by factorizing the small matrices instead of treating the problem in its unstructured form. The class of least squares problems for which efficient direct solutions are available has been extended to accommodate the case of an additional Kronecker product equality constraint by Barrlund [5] using the null-space method. Cases where the coefficient matrix is composed of two rows of Kronecker products can be handled with the use of the so-called generalized SVD, as shown by Bardsley et al. [6] in the context of a wavefront reconstruction problem. Unfortunately, the class of problems where such exploitation is possible through direct methods is rather limited. For efficiently handling more complex structures of Kronecker products in general, iterative solution techniques are required.

Low Kronecker-rank matrices In the context of this thesis work, a structure of considerable interest is the *sums-of-Kronecker* form:

$$\mathbf{A} = \sum_{j=1}^{M} \mathbf{B}_j \otimes \mathbf{C}_j,$$

where the number M is referred to as the *Kronecker* or *separation rank* of the matrix $\mathbf{A}$. In particular, we are most interested in this form in case $\mathbf{A}$ is of *low Kronecker-rank*, i.e. $M \ll n$; the favorable information compression Kronecker products allow is most prominent and effective in this case. Low Kronecker-rank matrices often arise when modeling realistic systems where theoretical ranks of one are ruined by imperfections or other sources of noise, forcing us to include additional correction terms for a satisfactorily accurate description. In other cases, the low Kronecker structures may not appear as an intrinsic property of the system, but can be used as sufficient approximations and present a trade-off between model accuracy and complexity. For example, low Kronecker-rank matrices are good approximations for blurring operators in two dimensional image restoration [7] and for the covariance matrices of spatio-temporal processes such as video streams [8] or atmospheric turbulence [9]. The latter come into play in the context of adaptive optics systems, which have also been shown to admit a low Kronecker-rank representation [10].

Unfortunately, low Kronecker-rank matrices do not possess the convenient algebraic properties of simple Kronecker products, nor do they retain a low separation rank in their inverse or factorizations. This severely hampers the possibility to exploit their structure in algorithms and presents a challenge for efficiently solving problems in which they appear. Attempts at solving simple matrix equations with these form have been based on mostly iterative methods, such as the gradient and least-squares iterative approaches to least squares problems with a coefficient matrix in sums-of-Kronecker form [11]. This class of methods has also been extended to handle general coupled matrix equations involving multiple unknowns, i.e. with a block coefficient matrix composed of rows and columns of (single) Kronecker products [12], [13]. However, these are all based on the Gauss-Seidel iterations and show worse convergence rates than the more advanced Krylov-subspace methods, e.g. the conjugate gradient method which has also been adapted for solving these types of equations [14].

In this work, we are mainly interested in using such Krylov-subspace methods in order to tackle the low Kronecker-rank linear equations encountered during our study of the adaptive optics wavefront control problem. The convergence rate of these methods essentially depends on the conditioning of the system matrices, thus preconditioning plays a vital role in developing efficient solution algorithms which converge in the minimal number of iterations. For sums or rows of Kronecker product matrices, authors have proposed an SVD approximation based [15] and a generalized SVD based preconditioner [6], both of the form

$$(\mathbf{K} \otimes \mathbf{K})\mathbf{D}(\mathbf{K} \otimes \mathbf{K})^{\mathrm{T}}$$

where $\mathbf{K}$ is a single Kronecker product and $\mathbf{D}$ is a diagonal matrix. Simpler preconditioners include the nearest, inverse, or mean-based Kronecker product, all of which have separation rank one. These have been successfully employed in wide range applications such as stochastic automata networks [16], [17], convection–diffusion model problems [18], and finite element discretization [19]. A recent survey of these techniques in the context of stochastic problems can be found in the thesis of Zander [20].

Low Kronecker-rank inverse approximation The idea of using a low Kronecker-rank preconditioner as opposed to single pairs has appeared in the literature in e.g. Giraldi et al. [21] or Oseledets and Dolgov [22] in the context of higher dimensional tensors[1]. Their advantage over simpler forms is that the increased quality of preconditioning does not necessarily require a substantial amount of additional computational time, because matrix-vector multiplications of Kronecker sums can be evaluated in parallel and the results from each pair can be merged together in a negligible number of operations. By using low Kronecker-rank preconditioners, we thus aim to decrease the number of iterations needed for a Krylov solver to converge by increasing the cost of the preconditioner. In essence, we are exchanging operations which must be evaluated in series with ones which can be evaluated in parallel, effectively reducing the overall computation time.

Finding low Kronecker-rank preconditioners can be formulated as an inverse approximation problem and will form the basis of our solutions presented for the wavefront control in adaptive optics. However, our interest in inverse approximation is also motived by their use in general structure preserving algorithms, such as the matrix sign iterations [23]. Structure preservation for e.g. so-called sequentially semi-separable matrices has been examined in the PhD dissertation of Rice [24], with potential applications to important areas in systems & control such as determining matrix stability, solving Lyapunov or Riccati equations, and H_2 or H_∞ controller synthesis. This motivates the study of when these algorithms can be carried efficiently with low Kronecker-rank matrices by retaining the low separation rank throughout the calculations.

The main mathematical contributions of this thesis relate to approximating the inverse of low Kronecker-rank matrices using this same form. We present the existing approaches and propose a slightly less efficient, but more robust solution. Robustness may be important for structure preserving algorithms with multiple iterations [24]. We also conduct a numerical study of when the full optimization procedure is most effective compared to a progressive inverse approximation scheme presented by Giraldi et al. [21]. This study is carried out within the context of both retaining accurate low Kronecker-rank inverse approximations and computing high-quality preconditioners for the certain types of equations.

Exploiting Kronecker matrix-vector multiplications For our practical studies of adaptive optics wavefront control and sparse aperture mask design, exploiting Kronecker structures will essentially boil down to efficiently computing the corresponding matrix-vector products; this seems necessitated by the complexity of the problems. In this regard, we note that there seem to be two notions of efficiently computing Kronecker products of the form $\boldsymbol{y} = (\mathbf{B} \otimes \mathbf{C})\,\boldsymbol{x}$:

$$\mathbf{Y} = \mathbf{C} \cdot \mathbf{X} \cdot \mathbf{B}^{\mathrm{T}} \qquad \text{and} \qquad \boldsymbol{y} = (\mathbf{B} \otimes \mathbf{I}_m) \cdot (\mathbf{I}_n \otimes \mathbf{C})\,\boldsymbol{x}.$$

We refer to the former as the *dense and small*, while to the latter as the *sparse and large form* of this computation after the form of the matrices in each case. The dense case is much more efficient in terms of memory and thus more suitable for real-time implementations in embedded systems such as for wavefront control. In the case study of an offline optimization problem encountered during the sparse aperture mask design, we argue that the sparse form can still be useful if memory is not an issue. For more on this topic, see the Appendix A-2.

[1]Higher dimension tensors of rank d take the form $\mathbf{B}_1 \otimes \mathbf{B}_2 \otimes \cdots \otimes \mathbf{B}_d$ or other compressive representations such as the tensor train format under consideration in [22]. Here we limit the scope of our discussions to the two dimensional Kronecker products appearing in the motivating wavefront control and mask design problems.

1-2 Outline

The body of this thesis is divided into four main chapter. The first two constitute our mathematical contributions and analysis, while the latter two consider the wavefront control and sparse aperture mask design problems highlighted in the beginning of the introduction.

In Chapter 2, the inverse low-Kronecker rank approximation problem is introduced as a more general Kronecker structured least squares problem. We present the nominal alternating least squares (ALS) scheme traditionally employed to solve problems of its form, then review two possible methods for solving the corresponding subproblems in each iteration of the algorithm. The first approach is based on dividing the objective into separate least-squares problems, column-wise for the unknown Kronecker matrices. The second approach, based on methods from the literature, forms the normal equations from the stationarity conditions for optimality. The progressive solution scheme for a suboptimal solution is also presented in this context. In both cases, imposing additional structures of symmetry and sparsity are also considered. The chapter is concluded by advocating for a Nesterov-based accelerated version of ALS, which is shown to be very effective yet does not appear often in the literature.

In Chapter 3, we review two main applications of the inverse approximation problem, namely its potential for preserving low Kronecker-rank structures and for providing good preconditioners for Kronecker structured linear equations and least-squares problems. The former serves as a preliminary study in the outlined direction, while the latter discussions give the necessary background for handling the types of problems that will encountered in the wavefront control problem.

Chapter 4 provides a background review and a detailed study of our wavefront control problem within the context of exploiting Kronecker structures in a minimum variance control framework. Potential solutions to the corresponding bound-constrained least-squares problem are proposed for three methods with good real-time capabilities, namely alternating direction method of multipliers (ADMM), projected alternating Barzilai-Borwein (PABB), and active sets (AS). The performance of these methods is analyzed and compared using a partially realistic numerical study.

Chapter 5 treats the sparse aperture mask design problem for a novel wavefront sensor under development for future space missions. The problem is introduced as a trade-off between achieving greatly needed light throughput for the mask and its ability to distinguish entities within a certain set of so-called Zernike modes, which form the basis for wavefront reconstruction. Two solution approaches are presented along with a framework for combining their advantages.

Finally, a brief conclusion is given at the end of the thesis, summarizing the main observations presented in detail in the individual chapters.

The algorithms and solution approaches used throughout the thesis work are based on implementations in the MATLAB programming environment [25].

1-3 Nomenclature

Scalars are denoted by lower or uppercase letters or symbols. We aim to reserve l for iteration numbers in iterative algorithms, i, j, k for indexing vectors and matrices, and capital letters M, N to denote the separation rank of Kronecker matrices. The functions $F(\cdot)$, $f(\cdot)$ $g(\cdot)$ are reserved to denote objective functions and constraints. Values which parameterize given algorithms are often denoted by Greek letters such as μ, ρ.

Vectors are written as boldface lower-case (sometimes Greek) letters such as $\boldsymbol{x}$, $\boldsymbol{\lambda}$. The boldface is used to make a distinction between indexing a set of vectors, such as $\boldsymbol{x}_1, \boldsymbol{x}_2$, and referring to the elements of a single vector $\boldsymbol{x} \in \mathbb{R}^n$, such as $x_1, x_2, \dots \boldsymbol{x}_n$. The scalar product of two vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ is using the dot-product notation $\langle \boldsymbol{x},\ \boldsymbol{y} \rangle = \sum_{i=1}^{n} x_i y_i$. The Euclidean-norm of a vector $\boldsymbol{x}$ is given as $\|\boldsymbol{x}\| = \sqrt{x_1^2 + x_2^2 + \cdots + x_n} = \langle \boldsymbol{x},\ \boldsymbol{x} \rangle$. The null-vector and the vector of ones is denoted by $\mathbf{0}$ and $\mathbf{1}$, respectively, where an index can be used to explicitly show its size e.g. $\mathbf{1}_n \in \mathbb{R}^n$. Relations such as $=$ and $\leq$ between vectors are to be interpreted element-wise. The 'diag' operator constructs a diagonal matrix from a vector whose entries on the diagonal are the elements of the vector.

Matrices are represented by boldface uppercase letters such as $\mathbf{A}$ and $\mathbf{X}$. Indexing is done similarly to as in the case of vectors, though when using the index k we also mean the kth column of a matrix, e.g. $\mathbf{F}_k \in \mathbb{R}^n$; this will be evident from the context. The inverse and transpose are written as $\mathbf{A}^{-1}$, and $\mathbf{A}^{\mathrm{T}}$. The ith largest singular value of a matrix is represented σ_i; the ratio of the largest and the smallest of these is given by the condition number κ, often used in the form of a function e.g. $\kappa\mathbf{A}$. The Frobenius-norm for a matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$ is denoted by $\|\mathbf{A}\|_F = \sqrt{\sum_{i=1}^{q} \sigma_i^2}$, where $q = \min(m, n)$. The nuclear norm is defined as $\|\mathbf{A}\|_* = \sum_{i=1}^{q} \sigma_i$ The condition number $\kappa(\mathbf{A})$ is defined as the ratio of its largest and smallest singular values. The null and identity matrix are denoted by $\mathbf{0}$ and $\mathbf{I}$, where a subscript may be used to denote their respective dimension. The 'vec' operation constructs a vector from a matrix by stacking its columns below each other.

The Kronecker product of two matrices is represented by the symbol $\otimes$ such as in $\mathbf{A} = \mathbf{B} \otimes \mathbf{C}$. The element-wise (Hadamard) product is denoted by $\odot$. When using the same letter for the two terms, the letters L and R are used to denote the left and right pairs, respectively, as in $\mathbf{A} = \mathbf{A}_L \otimes \mathbf{A}_R$. Calligraphic symbols are used to denote sets, such as $\mathcal{B}$ or $\mathcal{M}$. For a set of matrices $\{\mathbf{X}_1,\ \mathbf{X}_2\ \dots\ \mathbf{X}_n\}$, we often use the shorthand expression $\{\mathbf{X}_i\}$ where the range $i = 1,\ \dots,\ n$ of the index will be evident from the context. Elements of matrices are often referred to using MATLAB notation, e.g. $\mathbf{X}$`:,k` denotes the kth column of $\mathbf{X}$.

For describing computational complexities, we use the big-O notation. An operation costing $\mathcal{O}(n)$ floating-point operations (flops) is guaranteed to finish in at most $c \cdot n$ flops, for some constant c. These operations can be any of the four elemental $+$, $-$, $*$, or $/$.

While many of the topics addressed within this thesis entail their own specific nomenclature, we aimed to employ a unified notation. For example, coefficient matrices are often denoted by $\mathbf{A}$ or $\mathbf{H}$, while left hand side of linear equations by t, b, or T in case of matrix equations. Known Kronecker pairs within system structures are denoted by $\mathbf{B}$, $\mathbf{C}$; the unknown ones by $\mathbf{X}$, $\mathbf{Y}$. Preconditioners are usually denoted by $\mathbf{P}$ or $\mathbf{M}$ depending on whether they approximate the inverse of the coefficient matrix $\mathbf{A}$ or $\mathbf{A}^{\mathrm{T}}\mathbf{A}$ of a linear system. Other chapter-specific notations which may arise in our work are introduced in the respective sections.

Chapter 2

Low Kronecker-rank inverse approximation

In the following, we discuss two solutions to a low Kronecker-rank structured least squares problem, whose performance in the context of preconditioning and structure preservation through inverse approximation will be extensively studied as part of the next chapter.

2-1 Problem formulation

Let us begin by introducing the mathematical formulation of the problem under consideration:

Problem 2.1 (Kronecker-LS).

$$\underset{\mathcal{X}}{\text{minimize}}\ F(\mathcal{X}) = \left\| \mathbf{T} - \left(\sum_{j=1}^{M} \mathbf{B}_j \otimes \mathbf{C}_j \right) \left(\sum_{i=1}^{N} \mathbf{X}_i \otimes \mathbf{Y}_i \right) \right\|_F^2, \tag{2-1}$$

where $\mathbf{T} \in \mathbb{R}^{n^2 \times n^2}$, $\mathbf{B}_j, \mathbf{C}_j, \mathbf{X}_i, \mathbf{Y}_i \in \mathbb{R}^{n \times n}$ for $j = 1, \ldots, M$ and $i = 1, \ldots, N$, and the symbol $\mathcal{X} = \{\mathbf{X}_i\} \cup \{\mathbf{Y}_i\}$ represents the set of unknowns. The goal is to exploit the Kronecker structure to achieve an $\mathcal{O}(n^4)$ solution which scales down to $\mathcal{O}(n^3)$ in case $\mathbf{T}$ is also structured.

Assumption. The Kronecker matrices within the objective are of low separation rank, i.e. we have $M,\ N \ll n$.

For the specific case of $\mathbf{T} = \mathbf{I}_n \otimes \mathbf{I}_n$, this formulation serves to find an approximate inverse of a low separation rank matrix in a similar form. In this thesis, we are primarily interested in its application for just this purpose. Preserving the low Kronecker-rank form when taking the inverse allows continued exploitation of the structure and will be important for an efficient solution to the adaptive optics wavefront control problem examined in Chapter 4. Retaining additional properties in the approximation, such as symmetry or sparsity, are also important as they can be further exploited for efficiency. Hence we will also look at Problem 2.1 under these constraints on the solution.

Tensor structured least squares problems have been examined in the literature in the more general context of d-dimensional tensors, such as:

$$\underset{\{\mathbf{X}_i^1\},\{\mathbf{X}_i^2\},\ldots,\{\mathbf{X}_i^d\}}{\text{minimize}} \left\| \left(\sum_{p=1}^{P} \mathbf{T}_p^1 \otimes \cdots \otimes \mathbf{T}_p^d \right) - \left(\sum_{j=1}^{M} \mathbf{A}_j^1 \otimes \cdots \otimes \mathbf{A}_j^d \right) \left(\sum_{i=1}^{N} \mathbf{X}_i^1 \otimes \cdots \otimes \mathbf{X}_i^d \right) \right\|_F^2$$

In Beylkin and Mohlenkamp [26], a solution to this problem is presented for the case when the unknowns are vectors, not matrices. In a later work by Giraldi et al. [21], such a problem is encountered for the case $N = 1$ when searching for the inverse approximation in a progressive manner. The approximation sum is assembled term by term, not in its entirety, which is why only one tensor product appears as the unknown in each step. They also discuss methods to impose sparsity or symmetry on the solution. Finally, Oseledets and Dolgov [22] examines linear systems in the so-called *tensor-train format*, a different high dimensional tensor representation for which the tensor forms in the above problem are a special case. In their work, the focus is on tensors with high ranks (large d) and low modes (small core matrix size n). The inverse approximation problem is only looked at in the sense of rearranging $\mathbf{AX} = \mathbf{I}$ as $(\mathbf{I} \otimes \mathbf{A})\,\text{vec}(\mathbf{X}) = \text{vec}(\mathbf{I})$, leading to a solution which is not efficient in terms of the dimension n, though this is not problematic in their context.

The goal of this chapter is to provide a comprehensive overview of two main solutions to Problem 2.1, both of which are based on an underlying alternating least squares scheme for the set of unknown $\{\mathbf{X}_i\}$ and $\{\mathbf{Y}_i\}$ matrices. The first is based on a further decomposition of the subproblems by formulating a least squares problem for each column of the unknown matrices. The second is based on assembling the normal equations for their determination using the stationarity conditions for optimality and follows the ideas outlined in previous works [26], [21]. In both cases, we review how the solution can be constrained by the additional structures of sparsity and/or symmetry.

2-2 ALS solution scheme

The objective function $F(\mathcal{X})$ of Problem 2.1 is nonlinear due to the Kronecker products between the unknown $\mathbf{X}_i$ and $\mathbf{Y}_i$ pairs. Nevertheless, it possesses a key property, namely biconvexity, which will allow its tractable solution in an iterative manner. This is demonstrated by the following theorem.

Theorem 2.1. *The objective function defined in Problem 2.1 is biconvex in the set of unknowns* $\{\mathbf{X}_i\}$ *and* $\{\mathbf{Y}_i\}$*.*

Proof. By definition, a function is biconvex if fixing one of its unknowns renders it a convex function of the other. Suppose we fix each $\mathbf{Y}_i$, i.e. the set $\{\mathbf{Y}_i\}$. The term within the norm of (2-1) is then a linear function of the remaining unknown $\{\mathbf{X}_i\}$. Since the composition of a convex and a linear function remains convex, this implies that the objective is indeed convex in $\{\mathbf{X}_i\}$. A similar argument can be applied when fixing each $\mathbf{X}_i$ for the convexity in the remaining unknown $\{\mathbf{Y}_i\}$, and the proof is complete. $\square$

A common method for solving biconvex problems is known as alternating least squares (ALS). With ALS, the solution is found by alternatingly fixing one of the unknowns and solving for the other. This gives rise to a series of convex problems until the solution is approximated to an acceptable degree. Note that the ALS scheme is, in general, not guaranteed to yield the global optimum as a solution. In the context of Kronecker structured inverse approximations, ALS was employed in all the previously cited works [21], [22], and [26]. It has also been used for identifying Kronecker structured vector autoregressive with exogenous inputs (VARX) models [27].

The advantage of ALS is that the convex subproblems are typically much simpler to solve than the original one, allowing the resulting algorithm to be quite efficient if it converges reasonably fast. In the context of our Problem 2.1, the respective ALS solution scheme is summarized as Algorithm 2.1 below. Note that we extended the general framework to incorporate normalization within the unknown Kronecker pairs. This is to avoid possible numerical issues by preventing the elements of each $\mathbf{X}_i$ and $\mathbf{Y}_i$ pair from differing by several orders of magnitude. Also note that the ALS iterations could be terminated by a suitable criteria, such as a threshold reached in the relative change of the objective or solution.

Algorithm 2.1 (Kron-ALS) ALS solution scheme for Problem 2.1

```
   Input:
      Problem description $\mathbf{T},\ \{\mathbf{B}_j\},\ \{\mathbf{C}_j\}$
      Initial values $\{\mathbf{X}_i^{(0)}\},\ \{\mathbf{Y}_i^{(0)}\}$
1: for $l = 0, 1, 2, \dots$ do
2:    Set
         $$\{\mathbf{X}_i^{(l+1)}\} = \arg\min_{\{\mathbf{X}_i\}} \left\| \mathbf{T} - \left(\sum_{j=1}^{M} \mathbf{B}_j \otimes \mathbf{C}_j\right) \left(\sum_{i=1}^{N} \mathbf{X}_i \otimes \mathbf{Y}_i^{(l)}\right) \right\|_F^2 \quad (2\text{-}2a)$$
3:    Set
         $$\{\mathbf{Y}_i^{(l+1)}\} = \arg\min_{\{\mathbf{Y}_i\}} \left\| \mathbf{T} - \left(\sum_{j=1}^{M} \mathbf{B}_j \otimes \mathbf{C}_j\right) \left(\sum_{i=1}^{N} \mathbf{X}_i^{(l+1)} \otimes \mathbf{Y}_i\right) \right\|_F^2 \quad (2\text{-}2b)$$
4:    Normalize each $\mathbf{X}_i^{(l+1)}, \mathbf{Y}_i^{(l+1)}$ pair such that $\left\|\mathbf{X}_i^{(l+1)}\right\|_F = \left\|\mathbf{Y}_i^{(l+1)}\right\|_F$ for $i = 1, \dots, N$
5:    Check stopping criteria
6: end for
```

We now briefly examine the naïve solution to the respective ALS subproblems. Suppose we are at the first step of the lth iteration where each $\mathbf{Y}_i^{(l)}$ is fixed. Let us assemble the unknown set $\{\mathbf{X}_i\}$ into the vector $\boldsymbol{z} \in \mathbb{R}^{Nn^2}$ in the following manner:

$$\boldsymbol{z} = \begin{bmatrix} \operatorname{vec}(\mathbf{X}_1)^{\mathrm{T}} & \operatorname{vec}(\mathbf{X}_2)^{\mathrm{T}} & \dots & \operatorname{vec}(\mathbf{X}_N)^{\mathrm{T}} \end{bmatrix}^{\mathrm{T}}. \quad (2\text{-}3)$$

The corresponding linear least squares subproblem (2-2a) can then be written as:

$$\min_{\boldsymbol{z}} \left\| \boldsymbol{t} - \mathbf{H}^{(l)} \boldsymbol{z} \right\|_F^2 \quad (2\text{-}4)$$

for the $\boldsymbol{t} = \operatorname{vec}(\mathbf{T}) \in \mathbb{R}^{n^4}$ vectorization of $\mathbf{T}$ and some $\mathbf{H}^{(l)} \in \mathbb{R}^{n^4 \times Nn^2}$ coefficient matrix. In this form, such a problem can be solved in $\mathcal{O}(N^2 n^8)$ operations [28].

A substantial improvement can be achieved by noting that the objective (2-2a) can actually be separated into n further subproblems, because the kth column of each $\mathbf{X}_i$ only influences the kth block of n columns of the matrix $\mathbf{T}$. Let us denote these values by:

$$\boldsymbol{x}_{i,k} := \mathbf{X}_i(\texttt{:,k}) \qquad \text{and} \qquad \mathbf{T}_k := \mathbf{T}(\texttt{:, (k-1)n+1:kn}), \tag{2-5}$$

where $\boldsymbol{x}_{i,k} \in \mathbb{R}^n$ and $\mathbf{T}_k \in \mathbb{R}^{n^2 \times n}$, respectively. Assembling the kth columns of the unknown matrices into the vector $\boldsymbol{z}_k \in \mathbb{R}^{Nn}$ as:

$$\boldsymbol{z}_k = \left[\operatorname{vec}(\boldsymbol{x}_{1,k})^{\mathrm{T}} \quad \operatorname{vec}(\boldsymbol{x}_{2,k})^{\mathrm{T}} \quad \ldots \quad \operatorname{vec}(\boldsymbol{x}_{N,k})^{\mathrm{T}}\right]^{\mathrm{T}}, \tag{2-6}$$

the subproblems for each $k = 1, \ldots, n$ now take the form:

$$\min_{\boldsymbol{z}_k} \left\| \boldsymbol{t}_k - \mathbf{H}_k^{(l)} \boldsymbol{z}_k \right\|_F^2 \tag{2-7}$$

for the $\boldsymbol{t}_k = \operatorname{vec}(\mathbf{T}_k) \in \mathbb{R}^{n^3}$ rearrangement of $\mathbf{T}_k$ and the corresponding $\mathbf{H}_k^{(l)} \in \mathbb{R}^{n^3 \times Nn}$ coefficient matrix. The computational complexity of solving all n subproblems is considerably reduced and amounts to a total cost of $\mathcal{O}(N^2 n^6)$ operations.

In the following, we will show that by making proper use of the Kronecker structures appearing in the ALS subproblems, the naïve solution can be further improved to achieve the target solution with $\mathcal{O}(n^4)$ dependency on the matrix sizes.

2-2-1 Progressive approximation scheme

In Giraldi et al. [21], Problem 2.1 is solved for $\mathbf{T} = \mathbf{I}$ using a progressive approximation of the unknown Kronecker pairs. However, a discussion on how this approach compares to the full solution was not given. As we will conduct such a comparison in the next chapter, we briefly review their method for low Kronecker-rank inverse approximation, referred to as the Progressive-ALS algorithm in this thesis.

Suppose a Kronecker rank $(\hat{N} - 1)$ approximation of the unknowns has been calculated. The $\hat{N}$th term is then determined by solving:

$$\underset{\mathbf{X}_{\hat{N}}, \mathbf{Y}_{\hat{N}}}{\text{minimize}} \quad \left\| \left(\mathbf{I}_n \otimes \mathbf{I}_n - \sum_{i=1}^{\hat{N}-1} \mathbf{X}_i \otimes \mathbf{Y}_i \right) - \left(\sum_{j=1}^{M} \mathbf{B}_j \otimes \mathbf{C}_j \right) (\mathbf{X}_{\hat{N}} \otimes \mathbf{Y}_{\hat{N}}) \right\|_F \tag{2-8}$$

using ALS. Note that this has the same form as Problem 2.1 with $N = 1$ and $\mathbf{T}$ as a sum of Kronecker products. The progressive approximation scheme involves iteratively solving the above optimization problem for the unknown $\mathbf{X}_{\hat{N}}, \mathbf{Y}_{\hat{N}}$ pairs from $\hat{N} = 1, \ldots, N$. After each step, a correction is applied to improve upon the current solution. Let $\mathcal{U}_{\hat{N}}^X = \operatorname{span}\{\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_{\hat{N}}\}$ and $\mathcal{U}_{\hat{N}}^Y = \operatorname{span}\{\mathbf{Y}_1, \mathbf{Y}_2, \ldots, \mathbf{Y}_{\hat{N}}\}$. A better solution is then searched for within the subspace $\mathcal{U}_{\hat{N}} = \mathcal{U}_{\hat{N}}^X \otimes \mathcal{U}_{\hat{N}}^Y$. For example, with $\hat{N} = 2$, the improvement is sought as a linear combination of the basis obtained from $\mathbf{X}_1 \otimes \mathbf{Y}_1$, $\mathbf{X}_1 \otimes \mathbf{Y}_2$, $\mathbf{X}_2 \otimes \mathbf{Y}_1$, and $\mathbf{X}_2 \otimes \mathbf{Y}_2$. As we will see, this heuristic yields a suboptimal solution which may perform considerably worse than the one obtained by optimizing for all N unknown Kronecker pairs as a whole.

2-3 Distributed least squares approach

From this point onwards we will concentrate on solving one step of the Kron-ALS algorithm outlined in the previous section. Specifically, we assume that at the lth iteration each $\mathbf{Y}_i^{(l-1)}$ is fixed and we determine the solutions $\mathbf{X}_i^{(l)}$ for $i = 1, \ldots, N$. The updates for the $\mathbf{Y}_i$ matrices could be obtained following the presented derivations and will not be reviewed separately.

With the first proposed approach, the subproblems are decomposed and we solve a least squares problem for each column of the unknown matrices. After describing the algorithm and analyzing its computational complexity, we also discuss how it can be modified to recover a symmetric or a sparse solution.

2-3-1 Algorithm

To start the derivation, we restate the optimization subproblem (2-2a) to be solved for the $\mathbf{X}_i^{(l+1)}$ updates for simplicity:

$$\min_{\{\mathbf{X}_i\}} \left\| \mathbf{T} - \left(\sum_{j=1}^{M} \mathbf{B}_j \otimes \mathbf{C}_j \right) \left(\sum_{i=1}^{N} \mathbf{X}_i \otimes \mathbf{Y}_i^{(l)} \right) \right\|_F^2 . \tag{2-9}$$

Using property (A-5), the matrix multiplication of Kronecker products can be merged together, yielding:

$$\min_{\{\mathbf{X}_i\}} \left\| \mathbf{T} - \sum_{j=1}^{M} \sum_{i=1}^{N} \left(\mathbf{B}_j \mathbf{X}_i \right) \otimes \left(\mathbf{C}_j \mathbf{Y}_i^{(l)} \right) \right\|_F^2 . \tag{2-10}$$

Applying the Kronecker rearrangement operator to the matrix whose Frobenius norm we are taking does not change the value of the norm itself. Thus, with property (A-10), we have:

$$\min_{\{\mathbf{X}_i\}} \left\| \mathcal{R}\left(\mathbf{T}\right) - \sum_{j=1}^{M} \sum_{i=1}^{N} \operatorname{vec}\left(\mathbf{B}_j \mathbf{X}_i \right) \cdot \operatorname{vec}\left(\mathbf{C}_j \mathbf{Y}_i^{(l)} \right)^{\mathrm{T}} \right\|_F^2 . \tag{2-11}$$

Looking at the objective, it is clear that the kth column of the products $\mathbf{B}_j \mathbf{X}_i$, and thus of $\mathbf{X}_i$, only influences the kth block of n rows of the rearrangement $\mathcal{R}(\mathbf{T})$. Let us denote these latter two quantities by $\boldsymbol{x}_{i,k} \in \mathbb{R}^n$ and $\tilde{\mathbf{T}}_k \in \mathbb{R}^{n \times n^2}$, respectively, defined similarly as in (2-5). We can thus decompose (2-11) into n separate subproblems, one for each set of the N unknown matrix columns $\{\boldsymbol{x}_{i,k}\}_{i=1}^{N}$ for $k = 1, \ldots, n$. We will use the shorthand notation $\{\boldsymbol{x}_{i,k}\}$ for such a set with fixed column k. The kth subproblem can thus be written as:

$$\min_{\{\boldsymbol{x}_{i,k}\}} \left\| \tilde{\mathbf{T}}_k - \sum_{i=1}^{N} \sum_{j=1}^{M} \mathbf{B}_j \boldsymbol{x}_{i,k} \cdot \operatorname{vec}\left(\mathbf{C}_j \mathbf{Y}_i^{(l)} \right)^{\mathrm{T}} \right\|_F^2 . \tag{2-12}$$

Note that the number of unknowns has become n instead of the original n^2, while the number of elements in $\tilde{\mathbf{T}}_k$ has also decreased by a factor of n compared to the number in $\mathbf{T}$.

Let us now rewrite the summations in the subproblem (2-12) using matrix notation. To this end, we denote the kth column of each $\mathbf{Y}_i^{(l)}$ by $\boldsymbol{y}_{i,k}^{(l)} \in \mathbb{R}^n$ and introduce the following matrices:

$$\mathbf{B} := \begin{bmatrix} \mathbf{B}_1 & \mathbf{B}_2 & \dots & \mathbf{B}_M \end{bmatrix}, \tag{2-13a}$$

$$\mathbf{C}^{(l)\,\mathrm{T}} := \begin{bmatrix} (\mathbf{C}_1\boldsymbol{y}_{1,1}^{(l)})^{\mathrm{T}} & (\mathbf{C}_1\boldsymbol{y}_{1,2}^{(l)})^{\mathrm{T}} & \dots & (\mathbf{C}_1\boldsymbol{y}_{1,n}^{(l)})^{\mathrm{T}} \\ \vdots & \vdots & \ddots & \vdots \\ (\mathbf{C}_M\boldsymbol{y}_{1,1}^{(l)})^{\mathrm{T}} & (\mathbf{C}_M\boldsymbol{y}_{1,2}^{(l)})^{\mathrm{T}} & \dots & (\mathbf{C}_M\boldsymbol{y}_{1,n}^{(l)})^{\mathrm{T}} \\ \vdots & \vdots & \ddots & \vdots \\ (\mathbf{C}_1\boldsymbol{y}_{N,1}^{(l)})^{\mathrm{T}} & (\mathbf{C}_1\boldsymbol{y}_{N,2}^{(l)})^{\mathrm{T}} & \dots & (\mathbf{C}_1\boldsymbol{y}_{N,n}^{(l)})^{\mathrm{T}} \\ \vdots & \vdots & \ddots & \vdots \\ (\mathbf{C}_M\boldsymbol{y}_{N,1}^{(l)})^{\mathrm{T}} & (\mathbf{C}_M\boldsymbol{y}_{N,2}^{(l)})^{\mathrm{T}} & \dots & (\mathbf{C}_M\boldsymbol{y}_{N,n}^{(l)})^{\mathrm{T}} \end{bmatrix}, \tag{2-13b}$$

and assemble the set of unknown columns $\{\boldsymbol{x}_{i,k}\}$ as:

$$\mathbf{D}_k := \operatorname{diag}\left(\mathbf{I}_M \otimes \boldsymbol{x}_{1,k}, \dots \mathbf{I}_M \otimes \boldsymbol{x}_{N,k}\right) = \begin{bmatrix} \mathbf{I}_M \otimes \boldsymbol{x}_{1,k} & & \\ & \ddots & \\ & & \mathbf{I}_M \otimes \boldsymbol{x}_{N,k} \end{bmatrix}. \tag{2-14}$$

Here $\mathbf{D}_k \in \mathbb{R}^{(Nn)\times N}$, $\mathbf{B} \in \mathbb{R}^{n\times(Mn)}$, while the larger $\mathbf{C}^{(l)\,\mathrm{T}} \in \mathbb{R}^{(MN)\times n^2}$. Furthermore, let us denote the set of matrices which have the above form of $\mathbf{D}_k$ by the symbol $\mathcal{D}$. The kth subproblem (2-12) can now be compactly expressed as:

$$\min_{\mathbf{D}_k\in\mathcal{D}} \left\| \tilde{\mathbf{T}}_k - \left(\mathbf{1}_N^{\mathrm{T}} \otimes \mathbf{B}\right) \mathbf{D}_k \; \mathbf{C}^{(l)\,\mathrm{T}} \right\|_F^2 . \tag{2-15}$$

To continue, we denote the economy-sized QR and LQ factorizations of $\mathbf{B}$ and $\mathbf{C}^{(l)\,\mathrm{T}}$ by:

$$\mathbf{B} = \mathbf{Q}_B\mathbf{R}, \qquad\qquad \mathbf{C}^{(l)\,\mathrm{T}} = \mathbf{L}^{(l)}\;\mathbf{Q}_C^{(l)\,\mathrm{T}}, \tag{2-16}$$

where $\mathbf{Q}_B \in \mathbb{R}^{n\times n}$, $\mathbf{R} \in \mathbb{R}^{n\times(Mn)}$, $\mathbf{Q}_C^{(l)\,\mathrm{T}} \in \mathbb{R}^{(MN)\times n^2}$, $\mathbf{L}^{(l)} \in \mathbb{R}^{(MN)\times(MN)}$, and we assumed that the matrices $\mathbf{B}$ and $\mathbf{C}^{(l)\,\mathrm{T}}$ have full row ranks. Substituting into (2-15), we have:

$$\min_{\mathbf{D}_k\in\mathcal{X}} \left\| \tilde{\mathbf{T}}_k - \left(\mathbf{1}_N^{\mathrm{T}} \otimes \mathbf{Q}_B\mathbf{R}\right) \mathbf{D}_k(\mathbf{L}^{(l)}\;\mathbf{Q}_C^{(l)\,\mathrm{T}}) \right\|_F^2 . \tag{2-17}$$

Noting that $\left(\mathbf{1}_N^{\mathrm{T}} \otimes \mathbf{Q}_B\mathbf{R}\right) = \mathbf{Q}_B\left(\mathbf{1}_N^{\mathrm{T}} \otimes \mathbf{R}\right)$ and introducing the transformation:

$$\hat{\mathbf{T}}_k^{(l)} := \mathbf{Q}_B^{\mathrm{T}}\tilde{\mathbf{T}}_k\mathbf{Q}_C^{(l)}, \tag{2-18}$$

we finally arrive at the following form:

$$\min_{\mathbf{D}_k\in\mathcal{X}} \left\| \hat{\mathbf{T}}_k^{(l)} - \left(\mathbf{1}_N^{\mathrm{T}} \otimes \mathbf{R}\right) \mathbf{D}_k\mathbf{L}^{(l)} \right\|_F^2 . \tag{2-19}$$

A key observation is that $\hat{\mathbf{T}}_k^{(l)} \in \mathbb{R}^{n\times(MN)}$ has much less elements than $\tilde{\mathbf{T}}_k \in \mathbb{R}^{n\times n^2}$. It is now worthwhile to express how these relate to the unknown components of each $\boldsymbol{x}_{i,k}$. Denoting the matrix $\left(\mathbf{1}_N^{\mathrm{T}} \otimes \mathbf{R}\right) \mathbf{D}_k \mathbf{L}^{(l)}$ by $\hat{\mathbf{X}}^k \in \mathbb{R}^{n\times(MN)}$, we can write the pth column of this quantity using `MATLAB` notation as:

$$
\begin{aligned}
\hat{\mathbf{X}}^k(\texttt{:, p}) &= \sum_{j=0}^{M-1}\sum_{i=0}^{N-1} \mathbf{R}(\texttt{:,j*n+1:(j+1)*n}) \cdot \boldsymbol{x}_{i+1,k} \cdot \mathbf{L}^{(l)}(\texttt{i*M+u+1,p}) \\
&= \sum_{i=0}^{N-1}\left(\sum_{j=0}^{M-1} \mathbf{R}(\texttt{:,j*n+1:(j+1)*n}) \cdot \mathbf{L}^{(l)}(\texttt{i*M+j+1,p})\right) \cdot \boldsymbol{x}_{i+1,k} \\
&:= \sum_{i=1}^{N} \mathbf{H}_{p,i}^{(l)} \boldsymbol{x}_{i,k}
\end{aligned} \tag{2-20}
$$

where the coefficient matrices $\mathbf{H}_{p,i}^{(l)} \in \mathbb{R}^{n\times n}$ show how the columns of $\hat{\mathbf{X}}^k$ depend linearly on each $\boldsymbol{x}_{i,k}$; note, however, that it is independent of the unknown column index k. Figure 2-1 provides a graphical explanation of how its first block row is assembled for the case $N = 3$ and $M = 2$ to ease understanding. The uncovered relation allows us to rewrite (2-19) as the simple least squares problem:

$$
\min_{\{\boldsymbol{x}_{i,k}\}} \left\| \operatorname{vec}(\hat{\mathbf{T}}_k^{(l)}) - \begin{bmatrix} \mathbf{H}_{1,1}^{(l)} & \cdots & \mathbf{H}_{1,N}^{(l)} \\ \mathbf{H}_{2,1}^{(l)} & \cdots & \mathbf{H}_{2,N}^{(l)} \\ \vdots & \ddots & \vdots \\ \mathbf{H}_{MN,1}^{(l)} & \cdots & \mathbf{H}_{MN,N}^{(l)} \end{bmatrix} \cdot \begin{bmatrix} \boldsymbol{x}_{1,k} \\ \boldsymbol{x}_{2,k} \\ \vdots \\ \boldsymbol{x}_{N,k} \end{bmatrix} \right\|^2 \tag{2-21}
$$

or

$$
\min_{\boldsymbol{x}_k^{\text{cols}}} \left\| \operatorname{vec}(\hat{\mathbf{T}}_k^{(l)}) - \mathbf{H}^{(l)} \boldsymbol{x}_k^{\text{cols}} \right\|^2 \tag{2-22}
$$

for short, where $\mathbf{H}^{(l)} \in \mathbb{R}^{(MNn)\times(Nn)}$ and $\boldsymbol{x}_k^{\text{cols}} \in \mathbb{R}^{Nn}$. The solutions for each column k of the unknown $\mathbf{X}_i$ matrices can now be obtained by solving this equation. The derived algorithm is summarized in the box on the next page.

Remark. *Taking the transpose of the expression within the norm of (2-11) makes clear how the solution for $\{\mathbf{Y}_i\}$ would follow a similar derivation due to the same form of the equations.*

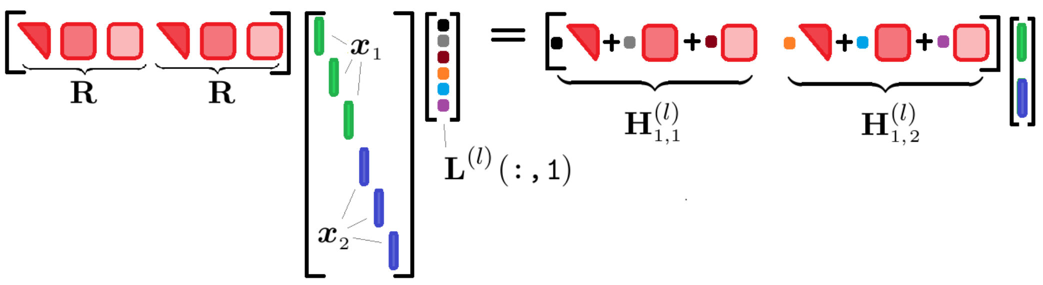


Figure 2-1: Assembling the first block row of the coefficient matrix $\mathbf{H}^{(l)}$ from its definition (2-20) for the case $N = 2$, $M = 3$.

Remark. *It is interesting to note that the matrix $\mathbf{H}^{(l)}$ has a Kronecker rank of M. This is due to the fact that each $\mathbf{H}_{p,i}^{(l)}$ is assembled as a linear combination of the same M matrices stemming from a partitioning of $\mathbf{R}$. However, the entire reason of looking at the inverse approximation problem is that iterative solutions for such problems, e.g. as in [11], are generally slow to converge and need to be tuned. Each iteration would involve $\mathcal{O}(n^2)$ computations, and we would solve* n *problems, which amounts to much work due to the high number of iterations. Instead, we propose solving the obtained final equation in the traditional manner, e.g. using the QR decomposition, especially since the matrix $\mathbf{H}^{(l)}$ remains the same for all columns k.*

Algorithm 2.2 Distributed least squares solution to the ALS subproblems

```
   Input:
      Problem description $\mathbf{T}, \{\mathbf{B}_j\}, \{\mathbf{C}_j\}, \{\mathbf{Y}_i^{(l)}\}$ of the $l$th iteration.
   Initialize:
      Assemble $\mathbf{B}$ using (2-13a) and determine $\mathbf{B} = \mathbf{Q}_B\mathbf{R}$.
1: Assemble ${\mathbf{C}^{(l)}}^{\mathrm{T}}$ using (2-13b) and determine ${\mathbf{C}^{(l)}}^{\mathrm{T}} = \mathbf{L}\,{\mathbf{Q}_C^{(l)}}^{\mathrm{T}}$.
2: Calculate each $\mathbf{H}_{p,i}^{(l)}$ for $p = 1, \dots, MN$ and $i = 1, \dots, N$ using the derivation (2-20).
3: Assemble $\mathbf{H}^{(l)}$ according to (2-21) and determine $\mathbf{H}^{(l)} = \mathbf{Q}_H^{(l)}\mathbf{R}_H^{(l)}$.
4: for $k = 1$ to $n$ do
5:    $\hat{\mathbf{T}}_k^{(l)} := \mathbf{Q}_B^{\mathrm{T}}\tilde{\mathbf{T}}_k\mathbf{Q}_C^{(l)}$
6:    Solve ${\mathbf{Q}_H^{(l)}}^{\mathrm{T}}\operatorname{vec}(\hat{\mathbf{T}}_k^{(l)}) = \mathbf{R}_H^{(l)}\boldsymbol{x}_k^{\mathrm{cols}}$.
7:    Recover the $k$th columns of the unknown matrices $\mathbf{X}_i$ from $\boldsymbol{x}_k^{\mathrm{cols}}$.
8: end for
```

2-3-2 Computational complexity

In this section we will examine the computational complexity of the previously derived distributed least squares solution to the ALS subproblems.

Theorem 2.2. *Assuming $M, N \ll n$, the computational complexity of the algorithm outlined in Algorithm 2.2 is of the order $\mathcal{O}(4MNn^4)$.*

Proof. We give the (approximate) computational cost of each of the steps given in the outlined algorithm. For certain matrix operations, such as computing factorizations, the cost is taken from the book by Golub and Van Loan [28], with the corresponding page number referred to in parenthesis. For clarity, the steps are given in a more detailed manner than before.

- (1) Determining the QR decomposition of $\mathbf{B} \in \mathbb{R}^{n\times(Mn)}$ amounts to a total of $4n^3/3 + 2(M-1)n^3 = 2(M-1/3)n^3$ flops (pg. 213, 225). This can be precomputed before starting the ALS iterations.

- (2) The MNn matrix-vector multiplications needed to assemble ${\mathbf{C}^{(l)}}^{\mathrm{T}}$ according to Equation (2-13b) take a total of $2MNn \cdot n^2 = 2MNn^3$ operations.

- (3) Determining the LQ decomposition of ${\mathbf{C}^{(l)}}^{\mathrm{T}} \in \mathbb{R}^{(MN)\times n^2}$ requires $2(MN)^2\left(n^2 - MN/3\right)$ flops (pg. 225).

- (4) The multiplication $\tilde{\mathbf{T}}_k \cdot \mathbf{Q}_C^{(l)}$ for each $k = 1, \ldots, n$ can be done by applying the previously obtained ${\mathbf{Q}_C^{(l)}}^{\mathrm{T}}$ to $\tilde{\mathbf{T}}_k^{\mathrm{T}} \in \mathbb{R}^{n \times n^2}$ in factored form, which requires a total number of $n \cdot 2MNn \left(2n^2 - MN\right) = 4MNn^4 - 2(MN)^2 n^2$ operations (pg. 213).
- (5) Calculating $\hat{\mathbf{T}}_k^{(l)} = \mathbf{Q}_B^{\mathrm{T}} \cdot \tilde{\mathbf{T}}_k \cdot \mathbf{Q}_C^{(l)}$ for each $k = 1, \ldots, n$ by applying $\mathbf{Q}_B^{\mathrm{T}}$ to $\tilde{\mathbf{T}}_k \cdot \mathbf{Q}_C^{(l)} \in \mathbb{R}^{n \times MN}$ can be done in $n \cdot 2MNn\,(2n - n) = 2MNn^3$ flops (pg. 213)[1].
- (6) Calculating all the $\mathbf{H}_{p,i}^{(l)}$ matrices for $p = 1, \ldots, MN$ and $i = 1, \ldots, N$ using (2-20) takes $MN^2 \cdot 2Mn^2/2 = (MN)^2 n^2$ operations, as $\mathbf{L}^{(l)}$ is lower triangular.
- (7) The QR decomposition of $\mathbf{H}^{(l)} \in \mathbb{R}^{(MNn) \times (Nn)}$ can be found in $2(Nn)^2\,(MNn - Nn/3) = 2N^3(M - 1/3)n^3$ flops (pg. 225).
- (8) Calculating ${\mathbf{Q}_H^{(l)}}^{\mathrm{T}} \operatorname{vec}(\hat{\mathbf{T}}_k^{(l)})$ for each $k = 1 \ldots n$ by applying ${\mathbf{Q}_H^{(l)}}^{\mathrm{T}}$ to $\operatorname{vec}(\hat{\mathbf{T}}_k^{(l)}) \in \mathbb{R}^{MNn}$ requires $n \cdot 2Nn\,(2MNn - Nn) = 2(2M - 1)N^2 n^3$ operations (pg. 213).
- (9) Computing the solutions $\boldsymbol{x}_{1,k}, \ldots, \boldsymbol{x}_{N,k}$ for each $k = 1 \ldots n$ by performing backsolves with the factor $\mathbf{R}_H^{(l)} \in \mathbb{R}^{(Nn) \times (Nn)}$ can be done at a cost of $n \cdot (Nn)^2 = N^2 n^3$ operations (pg. 240).

As we can see, the most computationally heavy step (4) and thus the entire algorithm consists of $\mathcal{O}(4MNn^4)$ operations, as was to be shown. □

For a certain case of interest, the complexity of the solution can be considerably reduced. This is highlighted by the following result.

Corollary 2.2.1. *If $\mathbf{T}$ takes the form $\mathbf{T} = \sum_{p=1}^{P} \mathbf{U}_p \otimes \mathbf{V}_p$ with $\mathbf{U}_p, \mathbf{V}_p \in \mathbb{R}^{n \times n}$, the complexity of Algorithm 2.2 reduces to $\mathcal{O}(f_{kron}(M,N) \cdot n^3)$, where $f_{kron}(M,N) = 2MN + 2(M-1/3)N^3 + (4M-1)N^2$ and the dependency on the number of terms P is negligible after precomputations, assuming $P \ll n$.*

Proof. If $\mathbf{T}$ admits such a Kronecker representation, its rearrangement can be written as a sum of P rank 1 matrices, i.e. $\mathcal{R}(\mathbf{T}) = \sum_{p=1}^{P} \operatorname{vec}(\mathbf{U}_p) \cdot \operatorname{vec}(\mathbf{V}_p)^{\mathrm{T}} := \sum_{p=1}^{P} \boldsymbol{u}_p \cdot \boldsymbol{v}_p^{\mathrm{T}}$. Keeping the kth block of n rows from this matrix, we have $\tilde{\mathbf{T}}_k = \sum_{p=1}^{P} \boldsymbol{u}_{p,k} \cdot \boldsymbol{v}_p^{\mathrm{T}}$. Calculating the matrix $\hat{\mathbf{T}}_k^{(l)} = \mathbf{Q}_B^{\mathrm{T}} \cdot \tilde{\mathbf{T}}_k \cdot \mathbf{Q}_C^{(l)} = \sum_{p=1}^{P} \left(\mathbf{Q}_B^{\mathrm{T}} \boldsymbol{u}_{p,k}\right) \cdot \left({\mathbf{Q}_C^{(l)}}^{\mathrm{T}} \boldsymbol{v}_p\right)^{\mathrm{T}}$ now becomes a much cheaper operation. The terms $\mathbf{Q}_B^{\mathrm{T}} \boldsymbol{u}_{p,k}$ can be precomputed for $k = 1, \ldots, n$ and $p = 1, \ldots, P$ for a total of $2n^3 P$ flops, while the P terms ${\mathbf{Q}_C^{(l)}}^{\mathrm{T}} \boldsymbol{v}_p$ require $4MNPn^2$ operations. The outer product of the resulting vectors of size n and MN for each p and k costs a further $MNPn^2$ flops. The entire algorithm thus consists of maximum $\mathcal{O}(n^3)$ operations. Disregarding the precomputated terms, these can be added together for steps (2) and (7)-(9) to get the dependency $f_{\text{kron}}(M,N) = 2MN + 2(M-1/3)N^3 + (4M-1)N^2$. □

Remark. *Additional structures, such as sparsity of $\mathbf{T}$ or the terms $\mathbf{B}_j, \mathbf{C}_j$, can also be readily exploited through multiplications involving these matrices. Nevertheless, the computationally heavy $\mathcal{O}(n^3)$ steps (7)-(9) will always remain and we do not further discuss these possibilities.*

[1]This step can actually be switched with the previous to allow the precalculation of $\mathbf{Q}_B^{\mathrm{T}} \tilde{\mathbf{T}}_k$, and applying only $\mathbf{Q}_C^{(l)}$ during iterations to the result

2-3-3 Imposing symmetry

We continue by discussing how symmetry can be enforced on the solutions $\mathbf{X}_i$ (and $\mathbf{Y}_i$) in the distributed least squares approach framework. With the current algorithm, this seems problematic because such structure prevents the column-wise decomposition of the ALS subproblems. For example, without symmetry, the first column of $\mathbf{X}_i$ only appears in the first column of the products $\mathbf{B}_j\mathbf{X}_i$. By explicitly enforcing symmetry, the elements of the first column of $\mathbf{X}_i$ are set equal to that of the first row, and now this entire product becomes a function of these values. A similar argument can be made for all columns of $\mathbf{X}_i$, and a least squares formulation encompassing all these dependencies at once would not lead to a computationally feasible solution.

In order to remain efficient, we need to retain the decomposability of the subproblems. To this end, we first introduce the following theorem.

Theorem 2.3. *Assume $f(\mathbf{X})$ is a strongly convex function and $g(\mathbf{X})$ is a linear invertible function of the matrix $\mathbf{X}$ such that $g(g(\mathbf{X})) = \mathbf{X}$ for all $\mathbf{X} \in \mathbb{R}^{n\times n}$. Then $f(g(\mathbf{X}))$ is also strongly convex and the optimal solution $\mathbf{X}^*$ to minimizing $\hat{f}(\mathbf{X}) = f(\mathbf{X}) + f(g(\mathbf{X}))$ satisfies $\mathbf{X}^* = g(\mathbf{X}^*)$.*

Proof. The composition $f(g(\mathbf{X}))$ does not change the convexity properties of this term due to the fact that $g(\mathbf{X})$ is linear and invertible, hence $f(g(\mathbf{X}))$ remains strongly convex. The summation of strongly convex functions is strongly convex, hence $\hat{f}(\mathbf{X})$ also exhibits this property. This guarantees that the optimization problem has a unique solution $\mathbf{X}^*$. Furthermore, since $g(g(\mathbf{X})) = \mathbf{X}$, we have $\hat{f}(\mathbf{X}) = \hat{f}(g(\mathbf{X}))$, thus for any minimum $\mathbf{X}^*$ the matrix $g(\mathbf{X}^*)$ is also a minimum. However, since the solution is unique, the relation $\mathbf{X}^* = g(\mathbf{X}^*)$ must hold, as desired. $\square$

This theorem allows us to enforce certain properties on the unknown matrices of Problem 2.1 by augmenting the objective function of the ALS subproblems with an additive term. For example, choosing $g(\mathbf{X}) = \mathbf{X}^{\mathrm{T}}$ forces the solution to become symmetric, while $g(\mathbf{X}) = -\mathbf{X}^{\mathrm{T}}$ forces it to be antisymmetric. Splitting techniques such as the alternating direction method of multipliers (ADMM)[2] [29] will allow an iterative solution to these augmented subproblems without sacrificing the efficiency gained from the distributed least squares approach.

For simplicity, from this point onwards we will only discuss the case of imposing symmetry on the solution matrices. Let us look at the subproblem (2-2a) that arises during the lth iteration of the ALS algorithm when we are solving for the unknown $\mathbf{X}_i^{(l+1)}$ matrices. As discussed previously, we augment the objective to restate the subproblem in the following manner, i.e. $g(\cdot)$ is interpreted as taking the transpose of all entries in the set $\{\mathbf{X}_i\}$:

$$\min_{\{\mathbf{X}_i\}} \left\| \mathbf{T} - \left(\sum_{j=1}^{M} \mathbf{B}_j \otimes \mathbf{C}_j\right)\left(\sum_{i=1}^{N} \mathbf{X}_i \otimes \mathbf{Y}_i^{(l)}\right)\right\|_F^2 + \left\| \mathbf{T} - \left(\sum_{j=1}^{M} \mathbf{B}_j \otimes \mathbf{C}_j\right)\left(\sum_{i=1}^{N} \mathbf{X}_i^{\mathrm{T}} \otimes \mathbf{Y}_i^{(l)}\right)\right\|_F^2$$

[2]For an overview of ADMM, please refer to a brief summary in Appendix B.

To achieve the decoupling of the two additive terms, we rewrite this subproblem to adhere to the general ADMM framework as follows:

$$\begin{aligned}
\underset{\{\mathbf{X}_i\},\{\mathbf{Z}_i\}}{\text{minimize}} \quad & \left\| \mathbf{T} - \left(\sum_{j=1}^{M} \mathbf{B}_j \otimes \mathbf{C}_j \right) \left(\sum_{i=1}^{N} \mathbf{X}_i \otimes \mathbf{Y}_i^{(l)} \right) \right\|_F^2 + \left\| \mathbf{T} - \left(\sum_{j=1}^{M} \mathbf{B}_j \otimes \mathbf{C}_j \right) \left(\sum_{i=1}^{N} \mathbf{Z}_i^{\mathrm{T}} \otimes \mathbf{Y}_i^{(l)} \right) \right\|_F^2 \\
\text{s.t.} \quad & \mathbf{X}_i = \mathbf{Z}_i, \qquad i = 1, \ldots, N.
\end{aligned}$$

This corresponds to the two variable split in the sets $\{\mathbf{X}_i\}$ and $\{\mathbf{Z}_i\}$. The solution is found iteratively using the ADMM update equations. In the scaled form (B-5) reformulated for matrices, these are written as follows [3,4]:

$$\begin{aligned}
\{\mathbf{X}_i^{(k+1)}\} &= \underset{\{\mathbf{X}_i\}}{\arg\min} \left\| \mathbf{T} - \left(\sum_{j=1}^{M} \mathbf{B}_j \otimes \mathbf{C}_j \right) \left(\sum_{i=1}^{N} \mathbf{X}_i \otimes \mathbf{Y}_i^{(l)} \right) \right\|_F^2 + \sum_{i=1}^{N} \frac{\rho}{2} \left\| \mathbf{X}_i - \mathbf{Z}_i^{(k)} + \mathbf{U}_i^{(k)} \right\|_F^2 \\
\{\mathbf{Z}_i^{(k+1)}\} &= \underset{\{\mathbf{Z}_i\}}{\arg\min} \left\| \mathbf{T} - \left(\sum_{j=1}^{M} \mathbf{B}_j \otimes \mathbf{C}_j \right) \left(\sum_{i=1}^{N} \mathbf{Z}_i^{\mathrm{T}} \otimes \mathbf{Y}_i^{(l)} \right) \right\|_F^2 + \sum_{i=1}^{N} \frac{\rho}{2} \left\| \mathbf{X}_i^{(k+1)} - \mathbf{Z}_i + \mathbf{U}_i^{(k)} \right\|_F^2 \\
\mathbf{U}_i^{(k+1)} &= \mathbf{U}_i^{(k)} + \mathbf{X}_i^{(k+1)} - \mathbf{Z}_i^{(k+1)}, \qquad i = 1, \ldots, N.
\end{aligned}$$

With a few minor modifications, Algorithm 2.2 can be used to solve the updates for both $\{\mathbf{X}_i^{(k+1)}\}$ and $\{\mathbf{Z}_i^{(k+1)}\}$. Notice that these two essentially have the same form in $\mathbf{X}_i$ and $\mathbf{Z}_i^{\mathrm{T}}$, respectively, and we only need to derive how to accommodate the added regularization terms. Focusing on the $\{\mathbf{X}_i^{(k+1)}\}$ updates, the regularization term can be added to each of the distributed least-squares problems by augmenting (2-22). For the kth column of the unknowns, this final least-squares form becomes:

$$\min_{\boldsymbol{x}_k^{\text{cols}}} \left\| \begin{bmatrix} \operatorname{vec}(\hat{\mathbf{T}}_k^{(l)}) \\ \sqrt{\frac{\rho}{2}} \operatorname{vec}\left(\mathbf{Z}_i^{(k)} - \mathbf{U}_i^{(k)} \right) \end{bmatrix} - \begin{bmatrix} \mathbf{H}^{(l)} \\ \sqrt{\frac{\rho}{2}} \mathbf{I}_{Nn} \end{bmatrix} \boldsymbol{x}_k^{\text{cols}} \right\|^2, \tag{2-23}$$

which can be solved in a similar manner as previously using the QR decomposition. The coefficient matrix will now have $(M+1)Nn$ rows instead of MNn, which leads to a slight increase in computational cost when calculating the factorization and applying the orthogonal transform to the left hand side.

The method can be efficient if the ADMM scheme converges in a few number of iterations; however, automatically selecting the penalty parameter that achieves this remains an open issue. In the context of this thesis, we imposed symmetry by tuning ρ manually.

Remark. *When solving for $\{\mathbf{Z}_i^{\mathrm{T}}\}$, the coefficient matrix will remain the same as the two updates have the same form. Its factorization only needs to be computed once for each ADMM iteration, or just once in the beginning of the ALS scheme if the penalty parameter ρ is kept constant, though achieving good convergence might be difficult in this case.*

[3] Choosing the penalty parameter ρ for good convergence remains an issue subject to future research; a simple adaptive scheme is given in e.g. [29].

[4] Note that the *upper* index (k) is used to represent that ADMM iteration number and is not related to the *lower* index k denoting column numbers or the (l) index of the ALS iterations

Remark. *Symmetry or antisymmetry could also be enforced in the context of ADMM by augmenting the ALS subproblems with an indicator function for these constraints. The updates for $\{\mathbf{Z}_i\}$ would then involve a projection of each $(\mathbf{X}_i^{(k+1)} + \mathbf{U}_i^{(k)})$ onto the set of symmetric matrices, an $\mathcal{O}(Nn^2)$ operation of adding this matrix to its transpose and dividing by two. However, this formulation does not preserve the strong convexity of the local ADMM objective functions. Accelerated ADMM methods do not guarantee a $\mathcal{O}(1/k^2)$ convergence rate for such a case, only $\mathcal{O}(1/k)$ [30]. Even with the cost of each iteration roughly halved, the solution may be found slower, which is why we opted to present the discussed alternative formulation of imposing symmetry. Further study is required to determine which solution works faster in practice.*

2-3-4 Imposing sparsity

In this section, we focus on how the solution to Problem 2.1 can be obtained given separate sparsity patterns on the unknown $\{\mathbf{X}_i\}$ and $\{\mathbf{Y}_i\}$ set of matrices. We again only consider the first update (2-2a) for the set $\{\mathbf{X}_i^{(l+1)}\}$ during the lth ALS iteration. Enforcing sparsity is important if we want to preserve such structure in our inverse approximations for efficiency. Although the solution presented can be readily extended to accommodate the case of having different sparsity patterns for the unknown matrices individually, for simplicity we will limit the scope of our discussion by the following assumption.

Assumption. The sparsity pattern imposed on each $\mathbf{X}_i$ and $\mathbf{Y}_i$ is identical for all $i = 1, \ldots, N$ and is such that the matrices have p nonzero elements in each column, where $Mp \ll n$.

To begin, we continue the derivation of Section 2-3-1 and restate the least squares problem (2-12) for determining the kth columns of the unknown $\mathbf{X}_i$ matrices for convenience:

$$\min_{\{\boldsymbol{x}_{i,k}\}} \left\| \tilde{\mathbf{T}}_k - \sum_{i=1}^{N} \sum_{j=1}^{M} \mathbf{B}_j \boldsymbol{x}_{i,k} \cdot \operatorname{vec}\left(\mathbf{C}_j \mathbf{Y}_i^{(l)} \right)^{\mathrm{T}} \right\|_F^2 . \tag{2-24}$$

The necessity of imposing a given sparsity pattern on the unknowns leads to three fundamental observations. First, we only need to solve for the nonzero entries of each column $\boldsymbol{x}_{i,k}$, greatly reducing the number of unknowns. Second, the sparsity of the matrices $\mathbf{Y}_i^{(l)}$ can and should be exploited when computing their products with each $\mathbf{C}_j$.

Let us extract the p nonzero elements from the kth columns of each $\mathbf{X}_i$ into the vectors $\boldsymbol{x}_{i,k}^s \in \mathbb{R}^p$. The corresponding columns of each $\mathbf{B}_j$ assembled as a separate matrix will be denoted by $\mathbf{B}_{j,k}^s \in \mathbb{R}^{n \times p}$; these new quantities thus satisfy $\mathbf{B}_j \boldsymbol{x}_{i,k} = \mathbf{B}_{j,k}^s \boldsymbol{x}_{i,k}^s$. A similar notation is introduced to extract the elements of each $\mathbf{C}_j$ and $\mathbf{Y}_i^{(l)}$ such that $\mathbf{C}_j \boldsymbol{y}_{i,k}^{(l)} = \mathbf{C}_{j,k}^s \boldsymbol{y}_{i,k}^{s(l)}$ with each $\mathbf{C}_{j,k}^s \in \mathbb{R}^{n \times p}$ and $\boldsymbol{y}_{i,k}^{s(l)} \in \mathbb{R}^p$.

The new notation allows us to rewrite our previous matrix notation of expressing the summation in the kth least squares subproblem (2-24). Instead of the matrix (2-13a), we introduce:

$$\mathbf{B}_k^s = \begin{bmatrix} \mathbf{B}_{1,k}^s & \mathbf{B}_{2,k}^s & \ldots & \mathbf{B}_{M,k}^s \end{bmatrix}, \tag{2-25a}$$

and note that the matrix $\mathbf{C}^{(l)\,\mathrm{T}}$ from (2-13b) can be compactly expressed as:

$$\mathbf{C}^{(l)\,\mathrm{T}} = \begin{bmatrix} (\mathbf{C}^s_{1,1}\boldsymbol{y}^{s(l)}_{1,1})^\mathrm{T} & (\mathbf{C}^s_{1,2}\boldsymbol{y}^{s(l)}_{1,2})^\mathrm{T} & \dots & (\mathbf{C}^s_{1,n}\boldsymbol{y}^{s(l)}_{1,n})^\mathrm{T} \\ \vdots & \vdots & \ddots & \vdots \\ (\mathbf{C}^s_{M,1}\boldsymbol{y}^{s(l)}_{1,1})^\mathrm{T} & (\mathbf{C}^s_{M,2}\boldsymbol{y}^{s(l)}_{1,2})^\mathrm{T} & \dots & (\mathbf{C}^s_{M,n}\boldsymbol{y}^{s(l)}_{1,n})^\mathrm{T} \\ \vdots & \vdots & \ddots & \vdots \\ (\mathbf{C}^s_{1,1}\boldsymbol{y}^{s(l)}_{N,1})^\mathrm{T} & (\mathbf{C}^s_{1,2}\boldsymbol{y}^{s(l)}_{N,2})^\mathrm{T} & \dots & (\mathbf{C}^s_{1,n}\boldsymbol{y}^{s(l)}_{N,n})^\mathrm{T} \\ \vdots & \vdots & \ddots & \vdots \\ (\mathbf{C}^s_{M,1}\boldsymbol{y}^{s(l)}_{N,1})^\mathrm{T} & (\mathbf{C}^s_{M,2}\boldsymbol{y}^{s(l)}_{N,2})^\mathrm{T} & \dots & (\mathbf{C}^s_{M,n}\boldsymbol{y}^{s(l)}_{N,n})^\mathrm{T} \end{bmatrix}. \tag{2-25b}$$

The unknown extracted columns $\{\boldsymbol{x}^s_{i,k}\}$ are assembled in the following manner:

$$\mathbf{D}^s_k = \operatorname{diag}\left(\mathbf{I}_M \otimes \boldsymbol{x}^s_{1,k}, \dots \mathbf{I}_M \otimes \boldsymbol{x}^s_{N,k}\right) = \begin{bmatrix} \mathbf{I}_M \otimes \boldsymbol{x}^s_{1,k} & & \\ & \ddots & \\ & & \mathbf{I}_M \otimes \boldsymbol{x}^s_{N,k} \end{bmatrix}. \tag{2-26}$$

In this formulation, $\mathbf{C}^{(l)\,\mathrm{T}}$ is much cheaper to assemble, while the matrices $\mathbf{B}^s_k \in \mathbb{R}^{n \times Mp}$ and $\mathbf{D}^s_k \in \mathbb{R}^{MNp \times MN}$ are much smaller than before. Denoting the set of matrices with such structure by $\mathcal{D}^s$, the kth subproblem (2-24) can thus be written as:

$$\min_{\mathbf{D}^s_k \in \mathcal{D}^s} \left\| \tilde{\mathbf{T}}_k - \left(\mathbf{1}^\mathrm{T}_N \otimes \mathbf{B}^s_k\right) \cdot \mathbf{D}^s_k \cdot \mathbf{C}^{(l)\,\mathrm{T}} \right\|^2_F, \tag{2-27}$$

which has the same form as (2-15). The solution concludes in a similar manner as before, with the QR decomposition of the matrices $\mathbf{B}^s_k$ and $\mathbf{C}^{(l)\,\mathrm{T}}$, followed by assembling a more compact least squares problem for the unknown extracted columns, leading to the final form:

$$\min_{\boldsymbol{x}^{s,\mathrm{cols}}_k} \left\| \operatorname{vec}(\hat{\mathbf{T}}^{(l)}_k) - \mathbf{H}^{s(l)}_k \boldsymbol{x}^{s,\mathrm{cols}}_k \right\|^2, \tag{2-28}$$

where the unknown $\boldsymbol{x}^{s,\mathrm{cols}}_k \in \mathbb{R}^{Np}$ and the coefficient matrix $\mathbf{H}^{s(l)}_k \in \mathbb{R}^{(M^2Np)\times(Np)}$. Due to $p \ll n$, this can be solved much more efficiently than in the dense case, i.e. in $\mathcal{O}(M^2N^3np^3)$ operations for all n columns. A detailed computational complexity analysis would follow similar suit as before.

Remark. *It is straightforward to combine the structures of sparsity and symmetry using this solution and the ADMM framework presented in the previous section.*

Remark. *Due to the assumption $Mp \ll n$, the matrix $\mathbf{B}^s_k$ is now tall instead of wide, which has to be taken into account during the derivations and why M^2 appears in the coefficient matrix dimension instead of just M as in the dense case. Solving for the unknown extracted columns will involve less expensive operations than before. On the other hand, these coefficient matrices of the subproblems will now differ, as the columns can have different sparsity patterns, and thus their factorizations have to be evaluated separately for each k.*

2-4 Gradient-based approach

In this section, we present an alternative gradient-based solution to the ALS subproblems which is applicable in case the matrix $\mathbf{T}$ is composed of Kronecker products. Using the notation introduced in Section 2-3-2, we thus have the form:

$$\mathbf{T} = \sum_{p=1}^{P} \mathbf{U}_p \otimes \mathbf{V}_p, \tag{2-29}$$

where each $\mathbf{U}_p, \mathbf{V}_p \in \mathbb{R}^{n \times n}$ and we assume a low separation rank representation, i.e. $P \ll n$.

As with the distributed least squares approach, we examine both the algorithm, its complexity, and possibilities to impose symmetry or sparsity on the solution matrices. The approach is based on the stationarity condition of setting the gradient with respect to the unknowns equal to zero, which is guaranteed to yield the optimal solution as we are working with convex functions. The derivations closely follow the ideas and tricks presented in Beylkin and Mohlenkamp [26], where the unknowns are vectors, not matrices. We also refer to Giraldi et al. [21], where the problem is solved for a single unknown and enforcing symmetry and sparsity are also discussed.

2-4-1 Algorithm

For simplicity, we again start by restating the ALS optimization subproblem (2-2a) to be solved for the $\mathbf{X}_i^{(l+1)}$ updates:

$$\min_{\{\mathbf{X}_i\}} \left\| \mathbf{T} - \left(\sum_{j=1}^{M} \mathbf{B}_j \otimes \mathbf{C}_j \right) \left(\sum_{i=1}^{N} \mathbf{X}_i \otimes \mathbf{Y}_i^{(l)} \right) \right\|_F^2 . \tag{2-30}$$

To lighten the equations, let us introduce the matrix $\mathbf{A} \in \mathbb{R}^{n^2 \times n^2}$ as:

$$\mathbf{A} = \sum_{j=1}^{M} \mathbf{B}_j \otimes \mathbf{C}_j . \tag{2-31}$$

The objective of the optimization subproblem can then be written in the compact form:

$$\left\| \mathbf{T} - \mathbf{A} \left(\sum_{i=1}^{N} \mathbf{X}_i \otimes \mathbf{Y}_i^{(l)} \right) \right\|_F^2 , \tag{2-32}$$

or, using the inner product defined on $\mathbb{R}^{n^2 \times n^2}$ matrices, as:

$$\left\langle \mathbf{T} - \mathbf{A} \left(\sum_{i=1}^{N} \mathbf{X}_i \otimes \mathbf{Y}_i^{(l)} \right) ,\ \mathbf{T} - \mathbf{A} \left(\sum_{i=1}^{N} \mathbf{X}_i \otimes \mathbf{Y}_i^{(l)} \right) \right\rangle . \tag{2-33}$$

By the distributive property of inner products, this becomes:

$$\langle \mathbf{T},\ \mathbf{T} \rangle - 2 \left\langle \mathbf{T},\ \mathbf{A} \left(\sum_{i=1}^{N} \mathbf{X}_i \otimes \mathbf{Y}_i^{(l)} \right) \right\rangle + \left\langle \mathbf{A} \left(\sum_{i=1}^{N} \mathbf{X}_i \otimes \mathbf{Y}_i^{(l)} \right) ,\ \mathbf{A} \left(\sum_{i=1}^{N} \mathbf{X}_i \otimes \mathbf{Y}_i^{(l)} \right) \right\rangle , \tag{2-34}$$

which can be rewritten by noting that the adjoint operator of a real matrix is its transpose[5]:

$$\langle \mathbf{T},\ \mathbf{T}\rangle - 2\left\langle \mathbf{A}^{\mathrm{T}}\mathbf{T},\ \left(\sum_{i=1}^{N}\mathbf{X}_i \otimes \mathbf{Y}_i^{(l)}\right)\right\rangle + \left\langle \mathbf{A}^{\mathrm{T}}\mathbf{A}\left(\sum_{i=1}^{N}\mathbf{X}_i \otimes \mathbf{Y}_i^{(l)}\right),\ \left(\sum_{i=1}^{N}\mathbf{X}_i \otimes \mathbf{Y}_i^{(l)}\right)\right\rangle. \tag{2-35}$$

Taking the total differential in the direction of some $\delta\mathbf{X}_t$, for the stationarity condition to hold we must have:

$$-2\left\langle \mathbf{A}^{\mathrm{T}}\mathbf{T},\ \delta\mathbf{X}_t \otimes \mathbf{Y}_t^{(l)}\right\rangle + 2\left\langle \mathbf{A}^{\mathrm{T}}\mathbf{A}\left(\sum_{i=1}^{N}\mathbf{X}_i \otimes \mathbf{Y}_i^{(l)}\right),\ \delta\mathbf{X}_t \otimes \mathbf{Y}_t^{(l)}\right\rangle = 0 \tag{2-36}$$

for any $\delta\mathbf{X}_t \in \mathbb{R}^{n\times n}$ and for each $t = 1, \ldots, N$. Substituting in the low Kronecker-rank forms (2-29) and (2-31) of $\mathbf{T}$ and $\mathbf{A}$, the first term in this expression can be rewritten as:

$$\begin{aligned}
\left\langle \mathbf{A}^{\mathrm{T}}\mathbf{T},\ \delta\mathbf{X}_t \otimes \mathbf{Y}_t^{(l)}\right\rangle &= \left\langle \left(\sum_{j=1}^{M}\mathbf{B}_j^{\mathrm{T}} \otimes \mathbf{C}_j^{\mathrm{T}}\right)\cdot\left(\sum_{p=1}^{P}\mathbf{U}_p \otimes \mathbf{V}_p\right),\ \delta\mathbf{X}_t \otimes \mathbf{Y}_t^{(l)}\right\rangle \\
&= \left\langle \sum_{j=1}^{M}\sum_{p=1}^{P}\mathbf{B}_j^{\mathrm{T}}\mathbf{U}_p \otimes \mathbf{C}_j^{\mathrm{T}}\mathbf{V}_p,\ \delta\mathbf{X}_t \otimes \mathbf{Y}_t^{(l)}\right\rangle \\
&= \sum_{j=1}^{M}\sum_{p=1}^{P}\left\langle \mathbf{B}_j^{\mathrm{T}}\mathbf{U}_p,\ \delta\mathbf{X}_t\right\rangle \cdot \left\langle \mathbf{C}_j^{\mathrm{T}}\mathbf{V}_p,\ \mathbf{Y}_t^{(l)}\right\rangle \\
&= \left\langle \sum_{j=1}^{M}\sum_{p=1}^{P}\mathbf{B}_j^{\mathrm{T}}\mathbf{U}_p\left\langle \mathbf{C}_j^{\mathrm{T}}\mathbf{V}_p,\ \mathbf{Y}_t^{(l)}\right\rangle,\ \delta\mathbf{X}_t\right\rangle \\
&:= \left\langle \hat{\mathbf{T}}_t^{(l)},\ \delta\mathbf{X}_t\right\rangle,
\end{aligned} \tag{2-37}$$

where we used properties (A-5) and (A-7) of Kronecker products and the introduced matrix $\hat{\mathbf{T}}_t^{(l)} \in \mathbb{R}^{n\times n}$. With a similar derivation, the second term becomes:

$$\begin{aligned}
\left\langle \mathbf{A}^{\mathrm{T}}\mathbf{A}\left(\sum_{i=1}^{N}\mathbf{X}_i \otimes \mathbf{Y}_i^{(l)}\right),\ \delta\mathbf{X}_t \otimes \mathbf{Y}_t^{(l)}\right\rangle &= \left\langle \sum_{j_1=1}^{M}\sum_{j_2=1}^{M}\sum_{i=1}^{N}\left(\mathbf{B}_{j_1}^{\mathrm{T}}\mathbf{B}_{j_2}\mathbf{X}_i \otimes \mathbf{C}_{j_1}^{\mathrm{T}}\mathbf{C}_{j_2}\mathbf{Y}_i^{(l)}\right),\ \delta\mathbf{X}_t \otimes \mathbf{Y}_t^{(l)}\right\rangle \\
&= \left\langle \sum_{i=1}^{N}\sum_{j_1=1}^{M}\sum_{j_2=1}^{M}\mathbf{B}_{j_1}^{\mathrm{T}}\mathbf{B}_{j_2}\left\langle \mathbf{C}_{j_2}\mathbf{Y}_i^{(l)},\ \mathbf{C}_{j_1}\mathbf{Y}_t^{(l)}\right\rangle\mathbf{X}_i,\ \delta\mathbf{X}_t\right\rangle \\
&:= \left\langle \sum_{i=1}^{N}\mathbf{H}_{t,i}^{(l)}\mathbf{X}_i,\ \delta\mathbf{X}_t\right\rangle,
\end{aligned} \tag{2-38}$$

where each $\mathbf{H}_{t,i}^{(l)} \in \mathbb{R}^{n\times n}$. Note that the expression for $\mathbf{H}_{t,i}^{(l)}$ can be obtained from that of $\mathbf{H}_{i,t}^{(l)}$ by swapping the indices of the matrices $\mathbf{B}_{j_1}^{\mathrm{T}}$ and $\mathbf{B}_{j_2}$; this implies that $\mathbf{H}_{t,i}^{(l)} = \mathbf{H}_{i,t}^{(l)\,\mathrm{T}}$. The derived expressions for these two terms allow the stationarity condition (2-36) to be written in the following compact form:

$$\left\langle \hat{\mathbf{T}}_t^{(l)} - \sum_{i=1}^{N}\mathbf{H}_{t,i}^{(l)}\mathbf{X}_i,\ \delta\mathbf{X}_t\right\rangle = 0. \tag{2-39}$$

[5]The adjoint $\mathcal{A}^*(\cdot)$ of an operator $\mathcal{A}(\cdot)$ satisfies the relation $\langle \mathcal{A}(\mathbf{X}),\ \mathbf{Y}\rangle = \langle \mathbf{X},\ \mathcal{A}^*(\mathbf{Y})\rangle$.

Since the stationarity condition must be satisfied for a step in any $\delta\mathbf{X}_t$ direction, (2-39) must hold independently of its value. Assuming the term on the left side of the dot product is non-zero, we thus have the condition:

$$\hat{\mathbf{T}}_t^{(l)} = \sum_{i=1}^{N} \mathbf{H}_{t,i}^{(l)} \mathbf{X}_i \tag{2-40}$$

for all $t = 1, \ldots, N$. Assembling these equations together, we arrive at the final form:

$$\begin{bmatrix} \hat{\mathbf{T}}_1^{(l)} \\ \hat{\mathbf{T}}_2^{(l)} \\ \vdots \\ \hat{\mathbf{T}}_N^{(l)} \end{bmatrix} = \begin{bmatrix} \mathbf{H}_{1,1}^{(l)} & \mathbf{H}_{1,2}^{(l)} & \ldots & \mathbf{H}_{1,N}^{(l)} \\ \mathbf{H}_{2,1}^{(l)} & \mathbf{H}_{2,2}^{(l)} & \ldots & \mathbf{H}_{2,N}^{(l)} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{H}_{N,1}^{(l)} & \mathbf{H}_{N,2}^{(l)} & \ldots & \mathbf{H}_{N,N}^{(l)} \end{bmatrix} \begin{bmatrix} \mathbf{X}_1 \\ \mathbf{X}_2 \\ \vdots \\ \mathbf{X}_N \end{bmatrix}, \tag{2-41}$$

or, for brevity:

$$\hat{\mathbf{T}}^{(l)} = \mathbf{H}^{(l)} \hat{\mathbf{X}}, \tag{2-42}$$

where $\hat{\mathbf{T}}^{(l)} \in \mathbb{R}^{Nn \times n}$, $\mathbf{H}^{(l)} \in \mathbb{R}^{Nn \times Nn}$, and $\hat{\mathbf{X}} \in \mathbb{R}^{Nn \times n}$. Note that the derived relation $\mathbf{H}_{t,i}^{(l)} = {\mathbf{H}_{i,t}^{(l)}}^{\mathrm{T}}$ implies that $\mathbf{H}^{(l)}$ is a symmetric matrix. The next iterate $\{\mathbf{X}_i^{(l+1)}\}$ of the ALS algorithm can be directly obtained by solving this linear equation for $\hat{\mathbf{X}}$. A summary of the derived algorithm is summarized in Algorithm 2.3 below.

Remark. *The solution to the subproblems for updating each $\mathbf{Y}_i^{(l+1)}$ can be obtained following a similar derivation and the stationarity condition corresponding to a step in the direction of each $\delta\mathbf{Y}_t$.*

Remark. *Each $\mathbf{H}_{t,i}^{(l)}$ is assembled as a linear combination of matrices of the form $\mathbf{B}_{j_1}^{\mathrm{T}}\mathbf{B}_{j_2}$ for $j_1 = 1, \ldots, M$ and $j_2 = 1, \ldots, M$. Therefore, the matrix $\mathbf{H}^{(l)}$ will have a Kronecker rank of M^2. Compare this to the $\mathbf{H}^{(l)}$ coefficient matrix derived in the distributed least squares approach, which had a Kronecker rank of M; this is expected, as here the normal equations were derived, increasing the separation rank and the conditioning of the coefficient matrix.*

Algorithm 2.3 Gradient-based solution to the ALS subproblems

```
Input:
    Problem description $\{\mathbf{U}_p\}, \{\mathbf{V}_p\}, \{\mathbf{B}_j\}, \{\mathbf{C}_j\}, \{\mathbf{Y}_i^{(l)}\}$
1: Determine each $\hat{\mathbf{T}}_t^{(l)}$ for $t = 1, \ldots, N$ according to its definition in Equation (2-37).
2: Calculate each $\mathbf{H}_{t,i}^{(l)}$ for $t = 1, \ldots, N$ and $i = 1, \ldots, N$ according to Equation (2-38).
3: Assemble $\hat{\mathbf{T}}^{(l)}$ and $\mathbf{H}^{(l)}$ block matrices from these terms, as shown in (2-41).
4: Solve $\hat{\mathbf{T}}^{(l)} = \mathbf{H}^{(l)}\hat{\mathbf{X}}$ and recover the solutions $\{\mathbf{X}_i^{(l+1)}\}$.
```

2-4-2 Complexity

In this section we will derive the computational complexity of the previously outlined gradient-based algorithm.

Theorem 2.4. *Assuming $M, N \ll n$, the computational complexity of the gradient-based approach Algorithm 2.3 is of the order $\mathcal{O}(f_{grad}(M, N) \cdot n^3)$, where $f_{grad}(M, N) = 2MN + 4N^3/3 + 3N^2$. The dependency on the number of P terms is negligible after precomputations, assuming $P \ll n$.*

Proof. We give the (approximate) computational cost of each of the steps given in the outlined algorithm; for certain matrix operations, the cost is taken from the book [28] with the corresponding page number given in parenthesis. For clarity, the steps are given in a more detailed manner than before.

- (1) Precomputing all possible $\mathbf{B}_j^{\mathrm{T}}\mathbf{U}_p$, $\mathbf{C}_j^{\mathrm{T}}\mathbf{V}_p$, and $\mathbf{B}_{j_1}^{\mathrm{T}}\mathbf{B}_{j_2}$ matrices for $j, j_1, j_2 = 1, \ldots, M$ and $p = 1, \ldots, P$ takes MP, MP, and $M(M+1)/2$ matrix-matrix multiplications respectively for a total of $2(2MP + M(M+1)/2)n^3$ flops. These appear in the equations for each $\hat{\mathbf{T}}^{(l)}$ and $\mathbf{H}_{t,i}^{(l)}$ and this step only needs to be evaluated once for the entire ALS algorithm.
- (2) Evaluating all $\hat{\mathbf{T}}^{(l)}$ matrices using (2-37) for $t = 1, \ldots, N$ requires $4MNPn^2$ operations. This stems from computing all possible inner products $\left\langle \mathbf{C}_j^{\mathrm{T}}\mathbf{V}_p,\ \mathbf{Y}_t^{(l)} \right\rangle$, multiplying them by the respective $\mathbf{B}_j^{\mathrm{T}}\mathbf{U}_p$ terms, and summing up the results, each of which are $\mathcal{O}(n^2)$ operations using the precomputed matrices.
- (3) The matrix products $\mathbf{C}_{j_2}\mathbf{Y}_i^{(l)}$ and $\mathbf{C}_{j_1}\mathbf{Y}_t^{(l)}$ in the definition (2-38) of each $\mathbf{H}_{t,i}^{(l)}$ do not have to be evaluated separately. It is enough to compute all possible combination $\mathbf{C}_j\mathbf{Y}_i$ for $j = 1, \ldots, M$ and $i = 1, \ldots, N$ due to the repetitions. This involves $2MNn^3$ flops.
- (4) Calculating each $\mathbf{H}_{t,i}^{(l)}$ for $t = 1, \ldots, N$ and $i = 1, \ldots, N$ then requires a total of about $2M^2N(N+1)n^2$ operations. Exploiting symmetry, it is enough to evaluate the M^2 inner products, matrix-scalar multiplications, and matrix-matrix additions for the $N(N+1)/2$ number of distinct (t, i) unordered pairs.
- (5) The QR decomposition of $\mathbf{H}^{(l)} \in \mathbb{R}^{(Nn)\times(Nn)}$ can be found in $2(Nn)^2\,(Nn - Nn/3) = 4N^3n^3/3$ flops (pg. 225).
- (6) Calculating ${\mathbf{Q}_H^{(l)}}^{\mathrm{T}}\hat{\mathbf{T}}^{(l)}$ by applying the orthogonal factor ${\mathbf{Q}_H^{(l)}}^{\mathrm{T}}$ to $\hat{\mathbf{T}}^{(l)} \in \mathbb{R}^{(Nn)\times n}$ requires $2Nn^2\,(2Nn - Nn) = 2N^2n^3$ operations (pg. 213).
- (7) Finally, computing the solutions $\{\mathbf{X}_i\}$ by performing backsolves with the factor $\mathbf{R}_H^{(l)} \in \mathbb{R}^{(Nn)\times(Nn)}$ to the n columns of ${\mathbf{Q}_H^{(l)}}^{\mathrm{T}}\hat{\mathbf{T}}^{(l)}$ can be done at a cost of N^2n^3 operations (pg. 240).

As we can see, the most computationally heavy steps are (3) and (5)-(7). Summing the cost of these steps reveals that the entire algorithm consists of $\mathcal{O}((2MN + 4N^3/3 + 3N^2)n^3)$ operations, as desired. □

This algorithm has lower computational complexity per iteration compared to the distributed least squares approach, whose complexity for the case when $\mathbf{T}$ admits a low Kronecker rank form was given in Corollary 2.2.1. This is due to the fact that we are solving the normal equations instead of taller least squares problems for the unknown matrices. In return, the precomputation step is more expensive in this case, as the dependency of $\mathcal{O}(n^3)$ calculations is quadratic in M instead of linear. For large M and cases when the ALS algorithm converges in a few iterations, the least-squares approach can be expected to perform faster. It is also numerically more robust due to the fact that we are not forming the normal equations, where the condition number becomes squared.

2-4-3 Imposing symmetry

We follow the methods presented in Giraldi et al. [21] for enforcing symmetry on the solution matrices $\mathbf{X}_i$ and $\mathbf{Y}_i$. Let us denote the set of symmetric matrices by $\mathcal{S}$ and denote the projection onto this set by:

$$\Pi_S(\mathbf{X}) := \frac{\mathbf{X} + \mathbf{X}^{\mathrm{T}}}{2}. \tag{2-43}$$

If the solutions are sought within the subspace of symmetric matrices, then it is enough for the stationarity condition to also only hold for step directions within that same subspace. The simplified form (2-39) thus becomes:

$$\left\langle \hat{\mathbf{T}}_t^{(l)} - \sum_{i=1}^{N} \mathbf{H}_{t,i}^{(l)} \mathbf{X}_i,\ \Pi_S(\delta \mathbf{X}_t) \right\rangle = 0, \tag{2-44}$$

which can also be written as:

$$\left\langle \Pi_S \left(\hat{\mathbf{T}}_t^{(l)} - \sum_{i=1}^{N} \mathbf{H}_{t,i}^{(l)} \mathbf{X}_i \right),\ \delta \mathbf{X}_t \right\rangle = 0 \tag{2-45}$$

since the adjoint of the operator $\Pi_S(\cdot)$ is the same as itself. This equation must hold for any $\delta\mathbf{X}_t$, which implies the condition

$$\sum_{i=1}^{N} \mathbf{H}_{t,i}^{(l)} \mathbf{X}_i + \sum_{i=1}^{N} \mathbf{X}_i^{\mathrm{T}}\, \mathbf{H}_{t,i}^{(l)\,\mathrm{T}} = \hat{\mathbf{T}}_t^{(l)} + \hat{\mathbf{T}}_t^{(l)\,\mathrm{T}} := \hat{\mathbf{S}}_t^{(l)} \tag{2-46}$$

for all $t = 1, \ldots, N$. If there is only one unknown matrix, i.e. $N = 1$ as discussed in [21], this becomes the so-called Lyapunov equation:

$$\mathbf{H}_{1,1}^{(l)} \mathbf{X}_1 + \mathbf{X}_1\, \mathbf{H}_{1,1}^{(l)\,\mathrm{T}} = \hat{\mathbf{S}}_1^{(l)}. \tag{2-47}$$

An efficient solution to this is given by Bartels and Stewart [31] with a complexity of roughly $\mathcal{O}((4\sigma + 11/2)n^3)$ operations. The solution uses the so-called QR algorithm for calculating eigenvalues [32]; σ is the average number of iterations required for this to terminate.

For the more general case, (2-46) constitutes the system of equations:

$$\begin{array}{ccccccc}
\left(\mathbf{H}_{1,1}^{(l)}\mathbf{X}_1 + \mathbf{X}_1^{\mathrm{T}}\,\mathbf{H}_{1,1}^{(l)\,\mathrm{T}}\right) & + & \left(\mathbf{H}_{1,2}^{(l)}\mathbf{X}_2 + \mathbf{X}_2^{\mathrm{T}}\,\mathbf{H}_{1,2}^{(l)\,\mathrm{T}}\right) & + \ldots + & \left(\mathbf{H}_{1,N}^{(l)}\mathbf{X}_N + \mathbf{X}_N^{\mathrm{T}}\,\mathbf{H}_{1,N}^{(l)\,\mathrm{T}}\right) & = & \hat{\mathbf{S}}_1^{(l)} \\
\left(\mathbf{H}_{2,1}^{(l)}\mathbf{X}_1 + \mathbf{X}_1^{\mathrm{T}}\,\mathbf{H}_{2,1}^{(l)\,\mathrm{T}}\right) & + & \left(\mathbf{H}_{2,2}^{(l)}\mathbf{X}_2 + \mathbf{X}_2^{\mathrm{T}}\,\mathbf{H}_{2,2}^{(l)\,\mathrm{T}}\right) & + \ldots + & \left(\mathbf{H}_{2,N}^{(l)}\mathbf{X}_N + \mathbf{X}_N^{\mathrm{T}}\,\mathbf{H}_{2,N}^{(l)\,\mathrm{T}}\right) & = & \hat{\mathbf{S}}_2^{(l)} \\
\vdots & & \vdots & & \vdots & & \vdots \\
\left(\mathbf{H}_{N,1}^{(l)}\mathbf{X}_1 + \mathbf{X}_1^{\mathrm{T}}\,\mathbf{H}_{N,1}^{(l)\,\mathrm{T}}\right) & + & \left(\mathbf{H}_{N,2}^{(l)}\mathbf{X}_2 + \mathbf{X}_2^{\mathrm{T}}\,\mathbf{H}_{N,2}^{(l)\,\mathrm{T}}\right) & + \cdots + & \left(\mathbf{H}_{N,N}^{(l)}\mathbf{X}_N + \mathbf{X}_N^{\mathrm{T}}\,\mathbf{H}_{N,N}^{(l)\,\mathrm{T}}\right) & = & \hat{\mathbf{S}}_N^{(l)}
\end{array}$$

whose solutions are sought in a symmetric form. We leave the study of this system for future work, such as evaluating whether the iterative gradient-based scheme handling a single equation of this form proposed by Xie et al. [33] could be extended and how it performs.

Remark. *The ADMM scheme presented for the distributed least squares approach could also be applied to impose symmetry, as regularization could also be accommodated with the gradient-based approach. In the context of this thesis, it was the preferred choice due to its ease of implementation.*

2-4-4 Imposing sparsity

We again employ the same method for enforcing a given sparsity solution (under the same assumption as in Section 2-3-4) as for enforcing symmetry, i.e. following the ideas of Giraldi et al. [21] and the projection operator corresponding to this case.

By the assumption, the sparsity patterns are identical for each of the unknown $\mathbf{X}_i$ matrices, so it is enough to work with the single projection operator defined according to its indicator set $\mathcal{I}_P$, which contains the pairs of (i,j) indices corresponding to the non-zero elements in the enforced pattern $\mathcal{P}$. With this definition, the projection operator for this case becomes:

$$[\Pi_P(\mathbf{X})]_{ij} := \begin{cases} [X]_{ij}, & \text{if } (i,j) \in \mathcal{I}_P, \\ 0, & \text{if } (i,j) \notin \mathcal{I}_P. \end{cases} \tag{2-48}$$

It is easy to verify that the adjoint of this operator is equal to itself, thus a similar derivation to the one presented in the previous section reveals the following condition for stationarity:

$$\left\langle \Pi_P\left(\hat{\mathbf{T}}_t^{(l)} - \sum_{i=1}^{N} \mathbf{H}_{t,i}^{(l)}\mathbf{X}_i\right),\ \delta\mathbf{X}_t \right\rangle = 0, \tag{2-49}$$

which must hold for any $\delta\mathbf{X}_t$, $t = 1,\dots,N$; i.e. the projected quantity should always be zero.

To proceed, let is introduce the following notation for the kth column $\hat{\mathbf{T}}_t^{(l)}$ and each $\mathbf{X}_i$, both vectors of length n:

$$\hat{\boldsymbol{t}}_{t,k}^{(l)} := \hat{\mathbf{T}}_t^{(l)}(\texttt{:,k}), \qquad \boldsymbol{x}_{i,k} := \mathbf{X}_i(\texttt{:,k}). \tag{2-50}$$

When applying the projection operator, only p entries of $\hat{\boldsymbol{t}}_{t,k}^{(l)}$ will remain non-zero, furthermore, only these same index entries of $\boldsymbol{x}_{i,k}$ are allowed to be non-zero; let us select and denote these with $\hat{\boldsymbol{t}}_{t,k}^{s(l)} \in \mathbb{R}^p$ and $\boldsymbol{x}_{i,k}^s \in \mathbb{R}^p$, respectively. Finally, let the corresponding rows and columns selected from each $\mathbf{H}_{t,i}^{(l)}$ be assembled into the separate matrix $\mathbf{H}_{t,i,k}^{s(l)} \in \mathbb{R}^{p\times p}$. For the stationarity condition (2-49), we must thus have:

$$\hat{\boldsymbol{t}}_{t,k}^{s(l)} - \sum_{i=1}^{N} \mathbf{H}_{t,i,k}^{s(l)}\boldsymbol{x}_{i,k}^s = \mathbf{0}, \qquad \text{for } \forall k = 1,\dots,n. \tag{2-51}$$

which represent N equations for each the kth columns of the unknown matrices obtained from $t = 1,\dots,N$. Assembled together, this yields n separate subproblems for each of the k columns of the unknowns:

$$\begin{bmatrix} \hat{\boldsymbol{t}}_{1,k}^{s(l)} \\ \vdots \\ \hat{\boldsymbol{t}}_{N,k}^{s(l)} \end{bmatrix} - \begin{bmatrix} \mathbf{H}_{1,1,k}^{s(l)} & \cdots & \mathbf{H}_{1,N,k}^{s(l)} \\ \vdots & \ddots & \vdots \\ \mathbf{H}_{N,1,k}^{s(l)} & \cdots & \mathbf{H}_{N,N,k}^{s(l)} \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_{1,k}^s \\ \vdots \\ \boldsymbol{x}_{N,k}^s \end{bmatrix} = \mathbf{0}. \tag{2-52}$$

The coefficient matrix is of size $\mathbb{R}^{(Np)\times(Np)}$, so solving all n columns entails $\mathcal{O}(N^3np^3)$ calculations. Similarly to the dense case, the least squares solution is more expensive due to its additional dependency on M at this step. For imposing symmetry on the solution as well, we have to apply the projection of the last section to these new equations.

2-5 Accelerated ALS

Nesterov's acceleration scheme was first proposed in order to improve upon the convergence of first order methods for minimizing convex functions [34]. It is based on the intuitive idea that solution iterates often 'zig-zag' when descending a valley, so taking larger steps than computed will often provide a better prediction for the true optimum. This is known as over-relaxation, and the correction $\hat{\boldsymbol{x}}^{(l)}$ of some unknown $\boldsymbol{x}^{(l)}$ at iteration l is determined according to:

$$\hat{\boldsymbol{x}}^{(l)} = \boldsymbol{x}^{(l)} + \gamma^{(l)} \left(\boldsymbol{x}^{(l)} - \boldsymbol{x}^{(l-1)} \right), \tag{2-53}$$

where $\gamma^{(l)} \geq 0$. This parameter represents the momentum carried by how the solution has changed previously, driving it towards the same direction, and is usually increased each iteration from an initial value of zero.

The acceleration scheme has been successfully applied to other optimization methods, such as the alternating direction methods ADMM and AMA [30]. Therein, the authors also recommend applying a restart rule to avoid too much momentum from gathering and delaying convergence. At the end of each iteration we check whether the objective decreased by a given factor, e.g. $\nu = 0.999$, compared to its previous value. If not, restart is accomplished by ignoring the results of the current iteration and resetting γ to zero so that no over-relaxation is employed in the next iteration. In the context of ALS, following the ideas for its use in ADMM and AMA, Nesterov's correction can applied at the end of each iteration and used during the first update of the next. This accelerated scheme is given as Algorithm 2.4 on the next page.

It is interesting to note that authors using ALS do not seem to employ or mention such an acceleration as a possibility, even though it is simple, cheap to implement, and can be quite effective. A counterexample can be found in a very recent work by Liavas et al. [35], where it is used during alternating optimization for finding nonnegative tensor factorizations, though with a different heuristic for determining γ than what Nesterov proposed. Unlike in the acceleration of ALS therein, we argue that any normalization of the unknown variables should be done *after* applying corrections to the iterates. This is because over-relaxation should be computed in terms of the difference between the iterates $\mathbf{Y}_i^{(l+1)}$ and $\mathbf{Y}_i^{(l)}$ directly before and after an update step in order to capture the direction of change. In our case, if done otherwise, the accelerated scheme often does not converge. This is not the case if a restart rule prevents the corrections from actually being applied, which is the reason we suspect this issue was not noticed in [35].

The effectiveness of the accelerated ALS scheme is demonstrated in Figure 2-2, where the convergence of the Kron-ALS, the accelerated Kron-ALS(F), and the accelerated with restart Kron-ALS(FR) schemes are compared for a sample inverse approximation problem solved from two different initial conditions. The figure shows that the fast variants not only improve the rate of convergence, but decrease its dependency on the initial estimates of the unknowns as well. The restart rule seems especially effective in helping the accelerated version converge even faster near the found optimum.

Algorithm 2.4 (Kron-ALS(F)) Accelerated ALS scheme for Problem 2.1

```
    Input:
        Problem description $\mathbf{T}$, $\{\mathbf{B}_j\}$, $\{\mathbf{C}_j\}$
        Initial values $\{\mathbf{X}_i^{(0)}\}$, $\{\mathbf{Y}_i^{(0)}\}$
    Initialize:
        $\{\hat{\mathbf{Y}}_i^{(0)}\} \leftarrow \{\mathbf{Y}_i^{(0)}\}$
        $\alpha_0 \leftarrow 1$
 1: for $l = 0, 1, 2, \dots$ do
 2:     Set
            $\{\mathbf{X}_i^{(l+1)}\} = \arg\min_{\{\mathbf{X}_i\}} \left\| \mathbf{T} - \left(\sum_{j=1}^{M} \mathbf{B}_j \otimes \mathbf{C}_j\right)\left(\sum_{i=1}^{N} \mathbf{X}_i \otimes \hat{\mathbf{Y}}_i^{(l)}\right) \right\|_F$
 3:     Set
            $\{\mathbf{Y}_i^{(l+1)}\} = \arg\min_{\{\mathbf{Y}_i\}} \left\| \mathbf{T} - \left(\sum_{j=1}^{M} \mathbf{B}_j \otimes \mathbf{C}_j\right)\left(\sum_{i=1}^{N} \mathbf{X}_i^{(l+1)} \otimes \mathbf{Y}_i\right) \right\|_F$
 4:     $\alpha_{l+1} \leftarrow \frac{1+\sqrt{1+4\alpha_l^2}}{2}$
 5:     for $i = 1, \dots, N$ do
 6:         $\hat{\mathbf{Y}}_i^{(l+1)} \leftarrow \mathbf{Y}_i^{(l+1)} + \frac{\alpha_l - 1}{\alpha_{l+1}}\left(\mathbf{Y}_i^{(l+1)} - \mathbf{Y}_i^{(l)}\right)$
 7:     end for
 8:     Normalize each $\mathbf{X}_i^{(l+1)}, \mathbf{Y}_i^{(l+1)}$ pair such that $\left\|\mathbf{X}_i^{(l+1)}\right\|_F = \left\|\mathbf{Y}_i^{(l+1)}\right\|_F$ for $i = 1, \dots, N$
 9:     Apply the same scaling to $\hat{\mathbf{Y}}_i^{(l+1)}$ as was needed for $\mathbf{Y}_i^{(l+1)}$
10:     Check termination criteria
11: end for
```

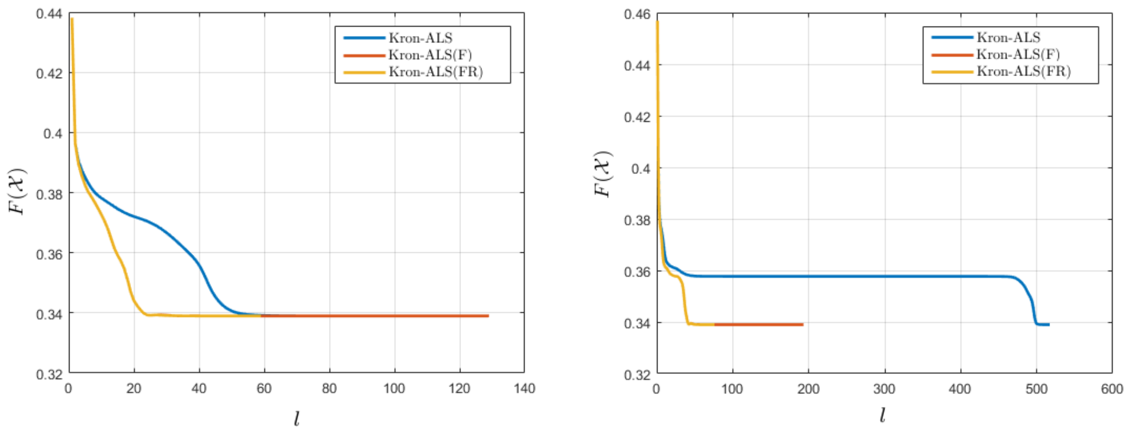


Figure 2-2: Comparison of the traditional and accelerated variants of ALS in the context of the low Kronecker-rank inverse approximation problem. The left and right images show the achieved objective as a function of the iteration count starting from two different initial conditions.

2-6 Conclusions

In the course of this chapter we derived two possible solution approaches for solving the ALS subproblems encountered during the low Kronecker-rank inverse approximation; one based on decomposing the problem into smaller least-squares ones, and another based on formulating the normal equations through the stationarity condition. A detailed complexity analysis reveals the former is slightly less efficient than the latter; in exchange, it is more robust, as will be seen in the coming chapter. These observations also hold when attempting to enforce additional structures on the unknowns, such as the explored symmetry and sparsity. We concluded our mathematical study of the inverse approximation problem by highlighting the importance of utilizing acceleration schemes in the general higher level ALS framework through a demonstrative example.

In the coming chapter, we will explore the potential of the derived algorithms in numerical applications, namely structure preserving algorithms and preconditioning linear systems.

Chapter 3

Numerical applications

In this chapter, we turn our attention to the more practical aspects of utilizing low Kronecker-rank inverse approximations. In particular, we focus on numerical applications related to preserving the low Kronecker-rank representation such as in the context of the matrix-sign iterations.

We also review two iterative solvers for linear equations and least squares problems, namely SQMR and MLSQR. Our previous findings are evaluated in the context of using low Kronecker-rank inverse approximations as preconditioners for these algorithms. The specific choice of SQMR and MLSQR is motivated by their potential for efficiently solving the type of problems we will encounter when tackling the adaptive optics wavefront control problem, which is the subject of the coming chapter.

3-1 Inverse structure preservation

We begin by conducting a detailed numerical study in order to motivate our interest in low Kronecker-rank inverses and uncover the limits of their use in practice. The main questions arising through the course of our analysis are as follows:

(i) whether low Kronecker-rank matrices admit sufficient structure preservation through the inverse operator,

(ii) in what situations are approximations of the inverse the most accurate and show potential usefulness in practice, and

(iii) how is the performance of the full Kron-ALS algorithm compare to that of the Progressive-ALS scheme (especially in these cases), as this was not discussed in the work of Giraldi et al. [21].

We note that our findings related to these question do not rely on rigorous mathematical proofs and only serve as an instructive preliminary study of exploring the potential of low Kronecker-rank inverse approximations for the considered numerical applications. A sample application of structure preservation is the presented in the context of the matrix sign algorithm in light of our findings.

3-1-1 Motivation

The goal of this section is to uncover theoretical and practical relations between the separation ranks M and N of a matrix $\mathbf{A}$ and its inverse. In particular, we are interested in how this property is translated:

$$\mathbf{A} = \sum_{j=1}^{M} \mathbf{B}_j \otimes \mathbf{C}_j \qquad \overset{N=?}{\Longrightarrow} \qquad \mathbf{A}^{-1} = \sum_{i=1}^{N} \mathbf{X}_i \otimes \mathbf{Y}_i, \tag{3-1}$$

where $\mathbf{A} \in \mathbb{R}^{n^2 \times n^2}$ and $\mathbf{B}_j$, $\mathbf{C}_j$, $\mathbf{X}_i$, $\mathbf{Y}_i \in \mathbb{R}^{n \times n}$. We begin by recollecting some results indicating the possibilities in terms of the Kronecker rank M of the original matrix.

(i) $M = 1$. If our matrix is composed of a single Kronecker product, the property (A-2b) shows that this will hold for its inverse as well; we have $\mathbf{A}^{-1} = \mathbf{B}_1^{-1} \otimes \mathbf{C}_1^{-1}$ and $N = 1$.

(ii) $M = 2$. This represents a special intermediate case where provable results exist on the upper bounds of the inverse separation rank. Indeed, in Tyrtyshnikov [36] it was shown that in this situation the inverse will always admit a sum-of-Kronecker representation with at most $N = n$ Kronecker pairs. Note, however, that this does not constitute as a low Kronecker-rank representation and the practical significance of this interesting theoretical result seems marginal. For example, just in terms of evaluating matrix-vector multiplications as summarized in Table A-1, using the unstructured $\mathbf{A}$ takes $2n^4$ operations, while with the n Kronecker terms we have $n \cdot 4n^3 = 4n^4$ flops.

(iii) $M > 2$. Simple examples reveal that with over two Kronecker terms in the original matrix, the inverse in general will only be bounded by the maximal $N = n^2$; any matrix of dimension n^2 by n^2 admits such a representation.

The three cases suggest that the inverse operation on low Kronecker-rank matrices does not have favorable structure preserving properties. For any small separation rank $M > 1$, we will have $N \gg M$ in general. Our motivation is thus mainly in studying the situations where the inverse may still admit an acceptable low-Kronecker rank approximation as opposed to a pure representation of such form.

Decaying Kronecker singular values Intuitively, we can expect that for matrices approaching a single Kronecker product form, the inverse will also tend towards such structure. This leads to the idea that low Kronecker-rank approximations of inverse may quite useful for matrices which exhibit a decay in their so-called *Kronecker singular values*, denoted by the set $\{\sigma_i^{\text{kron}}\}$ in the range of the possible $i = 1, \ldots, n^2$ assembled in decreasing order.

Kronecker singular values are defined in terms of the rearrangement (A-10), restated here for convenience:

$$\mathcal{R}(\mathbf{A}) = \sum_{j=1}^{M} \text{vec}(\mathbf{B}_j) \cdot \text{vec}(\mathbf{C}_j)^{\mathrm{T}}. \tag{3-2}$$

This operator establishes a connection between the separation rank of a matrix $\mathbf{A}$ and the traditional rank of its rearrangement. A single Kronecker product will have a rank-one rearrangement with one non-zero singular value. The singular values of $\mathcal{R}(\mathbf{A})$ can thus serve as a good quantifying measure of how close a matrix is to having a Kronecker-rank of one and will be referred to as the Kronecker singular values of the matrix.

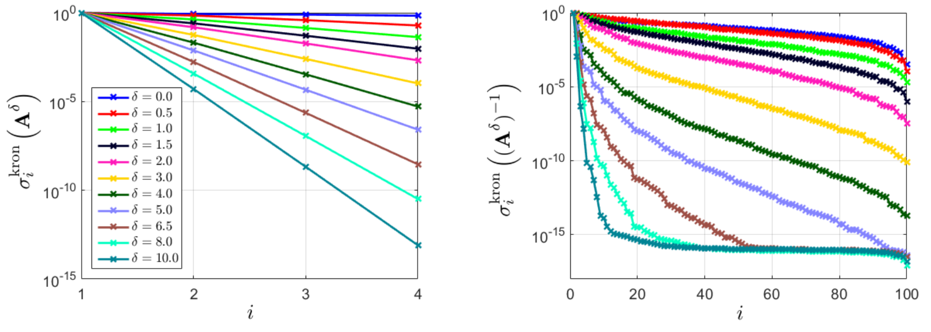
Matrices whose inverses exhibit a large decay in their Kronecker singular values can be expected to have the most potential to admit accurate low Kronecker-rank approximations. In order to examine the relation between the expected decay rates of a matrix inverse as a function of the decay in the matrix itself, we conducted a numerical experiment whose results are depicted in Figure 3-1.

In the experiment, we generated a set of $M = 4$ matrices, denoted by $\{\mathbf{B}_j^0\}$ and $\{\mathbf{C}_j^0\}$, using the `rand` MATLAB command; the matrix elements thus have a zero-mean Gaussian distribution with unit variance. These were then used as a basis for generating low Kronecker-rank matrices with exponentially decaying structure using the expression:

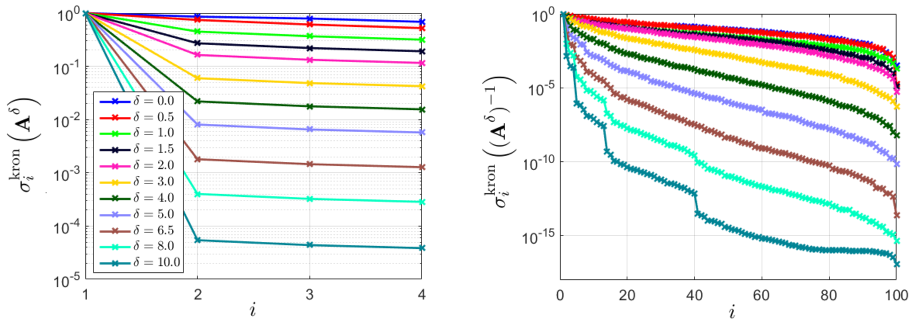
$$\mathbf{A}^\delta := \sum_{j=1}^{M} e^{-(j-1)\cdot\delta} \left(\mathbf{B}_j^0 \otimes \mathbf{C}_j^0\right). \tag{3-3}$$

Here the parameter δ is a good characterization of the decay in the Kronecker singular values of $\mathbf{A}^\delta \in \mathbb{R}^{n^2 \times n^2}$; we used a dimension of $n = 10$ for this study. The corresponding matrix inverses were then calculated, along with their Kronecker singular values. The obtained relation between the two for different values of δ can be seen on the figure on the top. For the step decay case one the bottom, we used the same construction (3-3) but with the weighing exponent kept constant after $j = 2$ to achieve a single drop in the Kronecker singular values.

The results verify our expectations: the increasing decay rates in the Kronecker singular values of a matrix translate to their inverses, allowing more accurate low Kronecker-rank representations of the latter.



(a) exponential decay



(b) step decay

Figure 3-1: Relation between the Kronecker singular values $\sigma_i^{\text{kron}}(\cdot)$ of a matrix and its inverse for different δ decay rates and types. They have been normalized to better visualize and ease the comparison of the decaying structures.

Quantifying the decay We aim to introduce a mathematical measure, which, as we will see, shows potential to capture the relation between the decay in the Kronecker singular values of a matrix and its inverse. The idea is motivated by the important role of the single Kronecker rank case, as well as the observation that the translated decay depends heavily on all the Kronecker values of the original, e.g. not just a drop compared to the first one. This behavior is demonstrated in Figure 3-1 by the difference between the (a) and (b) cases.

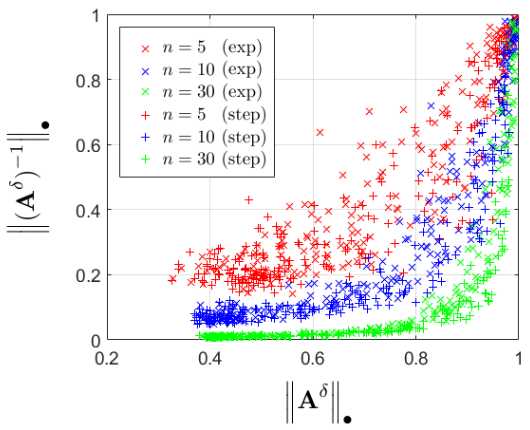
In an attempt to encompass the dependency of the decay on all the Kronecker singular values, let us to define the so-called *relative Kronecker nuclear norm* as:

$$\|\mathbf{A}\|_{\bullet} := \frac{\|\mathcal{R}(\mathbf{A})\|_{\infty}}{\|\mathcal{R}(\mathbf{A})\|_{*}} = \frac{\sigma_{\max}^{\text{kron}}(\mathbf{A})}{\sigma_1^{\text{kron}}(\mathbf{A}) + \sigma_2^{\text{kron}}(\mathbf{A}) + \cdots + \sigma_{n^2}^{\text{kron}}(\mathbf{A})}, \tag{3-4}$$

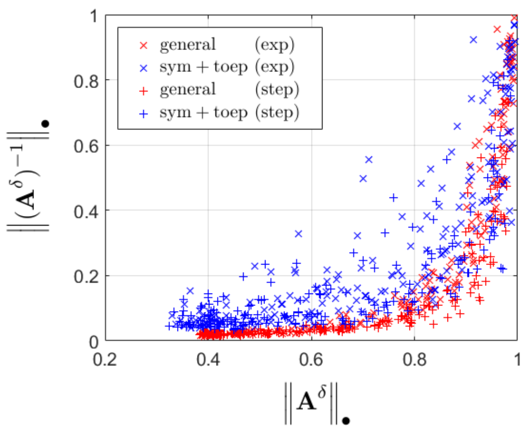
where the prominence of the first Kronecker singular values is measured against all of them together. It is our hope that this will serve as a good quantifying measure between the possible decay rates translated from a matrix to its inverse:

$$\left\|\mathbf{A}^{\delta}\right\|_{\bullet} \quad \overset{?}{\Longrightarrow} \quad \left\|(\mathbf{A}^{\delta})^{-1}\right\|_{\bullet}. \tag{3-5}$$

To study this relation, we again conducted a numerical analysis built upon the previous methodology of generating random (structured) matrices with different decay rates. Figures 3-2a and 3-2b demonstrate that the introduced measure can capture the effect of these decays more or less independently from its exact behavior: cases corresponding the exponential and step decays are plotted using the same color to readily show that they become bundled together, at least compared to other possible factors at play. We can see that smaller and more structured matrices have higher tendency for the prominence of the single terms – we can expect better low Kronecker-rank inverse approximations for such cases[1].



(a) effect of matrix size



(b) effect of matrix structure

Figure 3-2: Examining the relation between the relative Kronecker singular value of a matrix and its inverse. The dependency on the decay type – exponential or step decay, corresponding to the two cases in Figure 3-1 – seems negligible compared to other parameters, such as (a) matrix size or (b) structure. For the latter, we look at a demonstrative comparison between matrices without structure and symmetric Toeplitz ones.

[1]The matrices in this study were generated using the `rand` command and only capture general tendencies.

3-1-2 Low Kronecker-rank inverse approximation in practice

In this section we examine the practical use of the Kron-ALS algorithm discussed in the previous chapter for calculating low Kronecker-rank inverse approximations. In the previous section we found that these latter will only be most useful for matrices with large Kronecker singular values decays – here we aim to uncover the situations in which using the full optimization procedure to recover all terms of an inverse approximation is superior to the greedy progressive scheme by Giraldi et al. [21] as reviewed at the end of Section 2-2. A comparison there was not conducted in the context of his study.

We start by noting that the inverse approximation is the special case of the generalized least squares Problem 2.1, i.e. when $\mathbf{T} = \mathbf{I}$ in the form:

$$\underset{\{\mathbf{X}_i\},\{\mathbf{Y}_i\}}{\text{minimize}}\ F(\mathcal{X}) = \left\| \mathbf{I} - \left(\sum_{j=1}^{M} \mathbf{B}_j \otimes \mathbf{C}_j \right) \cdot \left(\sum_{i=1}^{N} \mathbf{X}_i \otimes \mathbf{Y}_i \right) \right\|_F^2 . \tag{3-6}$$

We ran the respective inverse approximation algorithms using this scheme for three demonstrative cases for randomly generated decaying Kronecker matrices of dimension $n = 10$. These correspond to the different regions in Figure 3-1, namely when taking a small, a large, and a value of the relative Kronecker nuclear norm $\left\|\mathbf{A}^{\delta}\right\|_{\bullet}$ in the middle region. These choices are motivated by the following observations:

(i) For a small rate of decay, Figure 3-1 shows a tendency for the Kronecker singular values of the inverse to flatten out as well. We expect that the difference between the full and progressive optimizations will be minimal, as the different Kronecker terms have similar impact and the attainable inverse approximation is of poor quality to begin with. Based on the point clouds in Figure 3-2, such a case seems to be described by roughly $\left\|\mathbf{A}^{\delta}\right\|_{\bullet} \leq 0.8$.

(ii) For a very large rate of decay, the inverse will approximate the limiting case of having one Kronecker rank. Thus the rank one approximation – which is the same for both the full and progressive alternating least squares (ALS) schemes – is already expected to be an accurate inverse approximation, and the difference between the two results (at least in terms of absolute value) should be marginal. The point cloud shows that such a case is mostly characterized by relative Kronecker nuclear norm values very near unity, e.g. $\left\|\mathbf{A}^{\delta}\right\|_{\bullet} \geq 0.99$.

(iii) For decay rates in the middle region, e.g. $0.8 < \left\|\mathbf{A}^{\delta}\right\|_{\bullet} < 0.99$, the behavior of the two algorithms is questionable and remains to be inspected.

Figure 3-3 shows a comparison of the performance of the different ALS algorithms for Kronecker nuclear norms representative of these three cases. Our observation for the limiting cases are verified by the graphs. Optimizing for the inverse approximation with all the terms at once seems most crucial for low Kronecker-rank matrices of moderate decay rates. The study also reveals, unfortunately, that warm-starting this algorithm from the result obtained from the (faster) greedy scheme does not aid convergence in a significant manner. Nevertheless, our observations are meaningful in practice, because noisy or imperfect models of Kronecker product structured systems can be expected to fall into just this region of moderate decays.

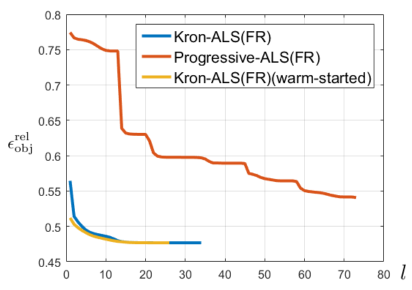
We also note the tendency for the full Kron-ALS algorithm to converge in a very limited number of iterations for cases where the Kronecker singular values exhibit a very large decay. Indeed, this can be expected by a continuity argument, as demonstrated for the limiting case by the following result.

Theorem 3.1. *Given* $\mathbf{B} \in \mathbb{R}^{n\times n}$ *and* $\mathbf{C} \in \mathbb{R}^{n\times n}$ *nonsingular matrices, and an estimate of the unknown term* $\mathbf{Y} \in \mathbb{R}^{n\times n}$ *such that* $\mathrm{tr}(\mathbf{CY}) \neq 0$*, the Kron-ALS algorithm 2.1 converges in a single iteration for the inverse approximation problem (3-6) .*

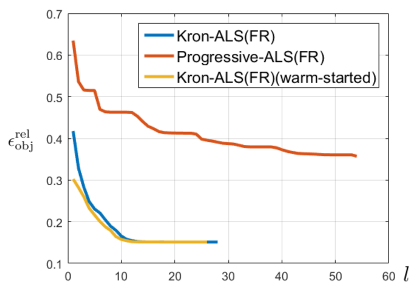
Proof. With the matrices defined by Equation (2-37) and (2-38) from the derivations of the gradient-based algorithm we have in this case ($N = 1$) the single terms $\mathbf{H}_{1,1} = \mathbf{B}^{\mathrm{T}}\mathbf{B}\,\langle\mathbf{CY},\ \mathbf{CY}\rangle$ and $\mathbf{T}_1 = \mathbf{B}^{\mathrm{T}}\,\langle\mathbf{I}_n,\ \mathbf{CY}\rangle$. Furthermore, let the scalar $\gamma = \langle\mathbf{I}_n,\ \mathbf{CY}\rangle \,/\, \langle\mathbf{CY},\ \mathbf{CY}\rangle$. The stationarity condition for optimality from the final result (2-41) then becomes:

$$\mathbf{B}^{\mathrm{T}}\mathbf{B}\mathbf{X} = \gamma\mathbf{B}^{\mathrm{T}} \qquad \Longrightarrow \qquad \mathbf{B}^{\mathrm{T}}\left(\mathbf{B}\mathbf{X} - \gamma\mathbf{I}_n\right) = 0. \tag{3-7}$$

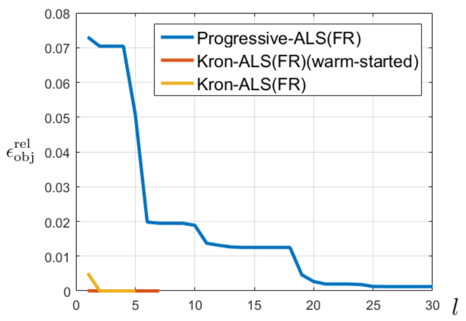
Since $\mathbf{B}$ is nonsingular, this is satisfied if and only if the term within the parenthesis is zero, yielding the result $\mathbf{X} = \gamma\mathbf{B}^{-1}$ such that $\mathbf{X}$ is a scaled version of $\mathbf{B}^{-1}$. A similar formula derived with this initial value of $\mathbf{X}$ for the second step of the ALS iterations shows that optimizing $\mathbf{Y}$ leads to the result $\mathbf{Y} = \gamma^{-1}\mathbf{C}^{-1}$. Their Kronecker product is thus indeed the inverse $(\mathbf{B} \otimes \mathbf{C})^{-1} = \mathbf{B}^{-1} \otimes \mathbf{C}^{-1}$, as desired. $\square$



(a) $\left\|\mathbf{A}^{\delta}\right\|_{\bullet} = 0.60;\quad \left\|(\mathbf{A}^{\delta})^{-1} = 0.18\right\|_{\bullet}$



(b) $\left\|\mathbf{A}^{\delta}\right\|_{\bullet} = 0.89;\quad \left\|(\mathbf{A}^{\delta})^{-1} = 0.67\right\|_{\bullet}$



(c) $\left\|\mathbf{A}^{\delta}\right\|_{\bullet} = 0.998;\quad \left\|(\mathbf{A}^{\delta})^{-1} = 0.962\right\|_{\bullet}$

Figure 3-3: Comparison of inverse approximation algorithms for various Kronecker singular value decay rates in a sample case of dimension $n = 10$ and problem sizes $M = 2$ and $N = 5$. In limiting cases of very large decays, the ALS subproblems can become very ill-conditioned and the gradient-based solution becomes unstable as opposed to the more robust least-squares approach.

3-1-3 Matrix sign iterations

We conclude our analysis of inverse approximations by briefly examining a structure preserving algorithm called the *matrix sign iterations* [23]. These are used to calculate the so-called *matrix sign function*, whose use in the context of structure preservation for efficient calculation related to many applications in controller synthesis and analysis has been examined in the context of sequentially semi-separable matrices by Rice [24]. The matrix sign function is a generalization of the scalar sign function to matrices; essentially, the returned matrix will have eigenvalues of ± 1 (or 0) depending on the sign of the eigenvalues in the original matrix.

For calculating the sign($\mathbf{A}$), an iterative procedure can be employed starting with $\mathbf{Z}_{(0)} := \mathbf{A}$ and repeating the following steps until convergence, i.e. $\text{sign}(\mathbf{A}) = \lim_{l\to\infty}(\mathbf{Z}_{(l)})$:

$$\mathbf{Z}_{(l+1)} = \frac{1}{2}\left(\mathbf{Z}_{(l)} + \mathbf{Z}_{(l)}^{-1}\right). \tag{3-8}$$

We can see that only the matrix and its inverse are required for the update, suggesting that a low Kronecker-rank approximation of the latter might be useful to maintain this structure throughout for an efficient implementation. Following the thoughts laid out in this direction by Sinquin [37], in our context we use the following structure-preservation procedure:

(1) Calculate an inverse approximation of $\mathbf{Z}_{(l)}$ using N_{inv} Kronecker pairs.

(2) Truncate[2] the Kronecker sum $\frac{1}{2}\left(\mathbf{Z}_{(l)} + \mathbf{Z}_{(l)}^{-1}\right)$ so $\mathbf{Z}_{(l+1)}$ retains a separation rank M_{mat}.

The efficiency of using low Kronecker-rank structures depends on the parameters N_{inv} and M_{mat} needed for an accurate estimation. Example iterations for randomly generated exponential decaying Kronceker singular valued matrices are shown on Figure 3-4. We can see that matrix structure will play a crucial role in the achievable performance.

This preliminary study leaves many open questions, such as the choice of the separation rank parameters or which structures admit accurate iteration results. Nevertheless, the findings suggest limited practical use of low Kronecker-rank structure preserving iterations; the inverses of this structure are mainly only good for approximations, and the accuracy deteriorates quickly during the course of a series of consecutive steps.

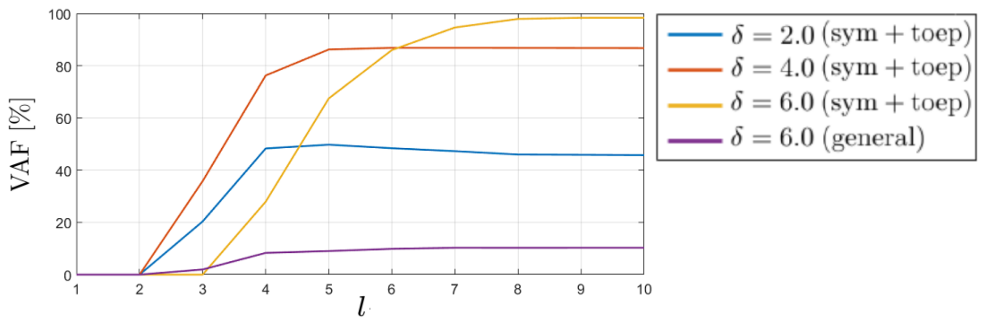


Figure 3-4: Variance-accounted-for values obtained for the matrix sign iterations for different decay rates and structured matrices. The VAF between the iterates $\mathbf{Z}_{(l)}$ and the true value $\mathbf{Z}_{(\infty)}$ is computed as $\text{VAF} = \max\left(0, 1 - \left\|\mathbf{Z}_{(l)} - \mathbf{Z}_{(\infty)}/\right\|_F \left\|\mathbf{Z}_{(\infty)}\right\|_F\right) \cdot 100\,\%$. For this example, the problem dimensions and separation ranks were $n = 20$, $N_{\text{inv}} = 5$, and $M_{\text{mat}} = 8$.

[2]This step can also be done using the Kron-ALS in the context of Problem 2.1 by setting $M = 1$, $N = M_{\text{mat}}$, and $\mathbf{T}$ as the sum of Kronecker pairs we wish to truncate therein.

3-2 Preconditioning linear equations

In our second application example, we look at the effectiveness of low Kronecker-rank inverse approximations as preconditioners for solving linear equations. This section also serves as a background review of the two solvers SQMR and MLSQR we will employ in the context of the adaptive optics wavefront control problem, to be the subject of the next chapter. Therefore, our study and review is limited to the scope of these two iterative solvers in the context of indefinite symmetric equations and least squares problem. For a more comprehensive overview of available solver, along with guideline for their selection for a given application, we refer to the templates given by Barrett et al. [38].

The efficiency of iterative solvers is mainly dependent on the computational costs for calculating matrix-vector multiplications with the system coefficient matrix, its transpose, or a preconditioner which serves to improve the conditioning of the equations. For a review of the considerations regarding their evaluations, we refer to Appendix A-2.

3-2-1 SQMR

We first take a look at preconditioning linear equations of the form:

$$\mathbf{A}\boldsymbol{x} = \boldsymbol{b}, \tag{3-9}$$

where $\mathbf{A}$ is a symmetric indefinite matrix. A sample solver for tackling these types of problems is the so-called symmetric quasi-minimum residual (SQMR) algorithm by Freund and Nachtigal [39]. Among other areas, it has been advocated for in the context of saddle point problems [40] as will be the case in our future study of the. Here we assume so-called *right preconditioning*, i.e. in the form of a variable transform:

$$(\mathbf{AP})(\mathbf{P}^{-1}\boldsymbol{x}) = \boldsymbol{b} \qquad \Longrightarrow \qquad \tilde{\mathbf{A}}\tilde{\boldsymbol{x}} = \boldsymbol{b}. \tag{3-10}$$

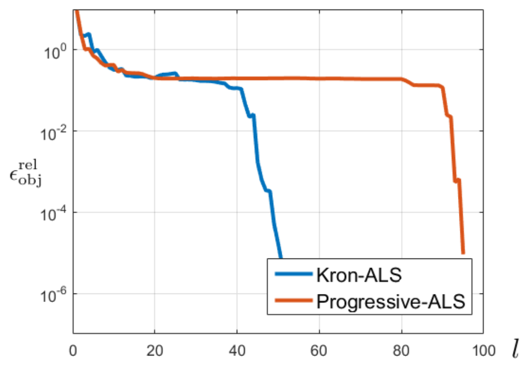
For the purpose of this review, we assume $\mathbf{A}$ admits the low-Kronecker rank representation for which the developed Kron-ALS algorithm can be used to find $\mathbf{P}$ in the same form[3]:

$$\mathbf{A} = \sum_{j=1}^{M} \mathbf{B}_j \otimes \mathbf{C}_j, \qquad \mathbf{P} = \sum_{i=1}^{N} \mathbf{P}_{L,j} \otimes \mathbf{P}_{R,j} \approx \mathbf{A}^{-1}. \tag{3-11}$$

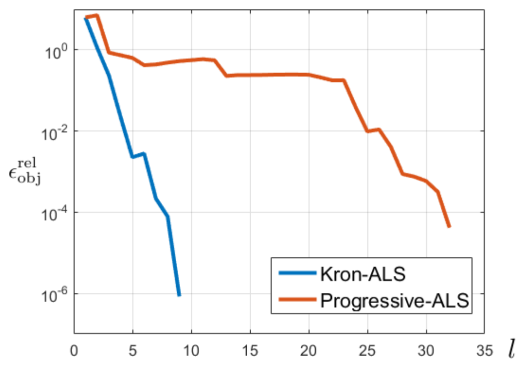
For its use in the SQMR algorithm, $\mathbf{P}$ must retain the symmetric property of $\mathbf{A}$; this can be enforced using our methods. The preconditioned system (3-10) is then solved for the transformed variable $\tilde{\boldsymbol{x}}$, from which the solution can be recovered as $\boldsymbol{x} = \mathbf{P}\tilde{\boldsymbol{x}}$. In this expression, as well as in the transformed coefficient matrix, taking the inverse of $\mathbf{P}$ is avoided, which allows its efficient representation in low Kronecker-rank form as it does not need to be determined.

Based on our results in Section 3-1-2, we expect favorable results and the most benefit of using the full Kron-ALS solution as opposed to the progressive scheme for relatively large decay rates. Figure 3-5 shows the performance of the inverse approximations calculated in the previous section (see Figure 3-3) in the context of solving a sample linear system using SQMR. It is clear that using the full optimization can be very advantageous for improving the performance of iterative solvers; we will also use it for the numerical study conducted in the context of the adaptive optics (AO) control problem.

[3] In the context of our wavefront control problem, only a certain block of the system matrix will have such structure



(a) $\left\|\mathbf{A}^{\delta}\right\|_{\bullet} = 0.60;\quad \left\|(\mathbf{A}^{\delta})^{-1} = 0.18\right\|_{\bullet}$



(b) $\left\|\mathbf{A}^{\delta}\right\|_{\bullet} = 0.89;\quad \left\|(\mathbf{A}^{\delta})^{-1} = 0.67\right\|_{\bullet}$

Figure 3-5: Effectiveness of low-Kronecker rank inverse approximations for preconditioning linear systems with different Kronecker singular value decays using the SQMR algorithm. The relative objective error $\epsilon^{\mathrm{rel}}_{\mathrm{obj}} = \|\mathbf{A}\boldsymbol{x} - \boldsymbol{b}\|_2 \, \|\boldsymbol{b}\|_2$ is plotted on a logarithmic scale as a function of the solver iterations. The advantage of using the full Kron-ALS solver is clear.

3-2-2 MLSQR

The Kron-ALS algorithm cannot be used to compute preconditioners for least squares problems with tall, non-square coefficient matrices. In the literature, an often employed technique for handling such equations is to instead approximate the inverse of normal system using a symmetric positive definite matrix, i.e working with $\|\mathbf{A}\boldsymbol{x} - \boldsymbol{b}\|_2$:

$$\mathbf{M} := \mathbf{P}^{\mathrm{T}}\mathbf{P} \approx (\mathbf{A}^{\mathrm{T}}\mathbf{A})^{-1}. \tag{3-12}$$

The preconditioner $\mathbf{P}$ to be used in e.g. the LSQR algorithm [41] is then recovered from the Cholesky decomposition of $\mathbf{M}$. An example for a simple two block-row Kronecker-product problem leading to a GSVD-based preconditioner can be found in the literature [6].

In the context of our problems with coefficient matrices composed of low Kronecker-rank matrices, or block rows of such forms, the Kron-ALS algorithm can be used to recover the symmetric approximation $\mathbf{M}$. However, the issue remains that its decomposition in a similarly computationally efficient manner only exists for separation ranks of one, such as:

$$\mathbf{M} = \mathbf{X} \otimes \mathbf{Y} = (\mathbf{L}_X\mathbf{L}_X^{\mathrm{T}}) \otimes (\mathbf{L}_Y\mathbf{L}_Y^{\mathrm{T}}) = (\mathbf{L}_X \otimes \mathbf{L}_Y) \cdot (\mathbf{L}_X^{\mathrm{T}} \otimes \mathbf{L}_Y^{\mathrm{T}}) := \mathbf{P}^{\mathrm{T}}\mathbf{P}, \tag{3-13}$$

where the Cholesky factorizations of the Kronecker terms exists by enforcing their symmetry[4]. For sums of $\mathbf{X}_i$ and $\mathbf{Y}_i$ terms, the preconditioner $\mathbf{P}$ will not admit an efficient low Kronecker-rank representation such as $\mathbf{M}$. A similar issue is encountered when working with sparse matrices, where low fill-in of the matrix $\mathbf{M}$ does not imply a sparse structure on the terms $\mathbf{P}$. We employ the same solution which has been developed to overcome this issue in the context of the LSQR solver by using the so-called matrix-free LSQR (MLSQR) [42]. It relies on reformulating the traditional LSQR iterations such that factorizing $\mathbf{M}$ can be avoided, allowing efficient matrix-vector multiplications with the structure possessed by this term itself. Note that when approximating $\mathbf{M}$, forming $\mathbf{A}^{\mathrm{T}}\mathbf{A}$ amplifies Kronecker decay rates.

[4]Actually, for this we need the positive definiteness of $\mathbf{X}$ and $\mathbf{Y}$, a property not explicitly imposed on the solutions but which nevertheless seems to be exhibited in practice as we are approximating such matrices.

3-3 Conclusions

The analysis in this chapter revealed that low Kronecker-rank matrix inversion does not exhibit favorable structure preserving properties. For applications where the thus unavoidable approximation errors are accumulated, such as through the matrix sign iterations, a preliminary study suggests that the possible practical usage of truncated separation rank inverse estimates in these situations will be limited to very special cases.

On the other hand, for applications such as preconditioning where an inverse approximation can be satisfactory, low Kronecker-rank matrices with a medium to high range of decay in their Kronecker singular values show great potential to admit such an inverse representation. In these situations, using the full Kron-ALS algorithm as opposed to the greedy Progressive-ALS scheme for inverse approximation is especially important as it produces superior results. A quantifying measure termed the relative Kronecker nuclear norm has been introduced as an attempt to relate the decay rates observed between the Kronecker singular values of matrices and those of their inverses.

The findings of this chapter were based on numerical experimentations and are not supported by rigorous mathematical analysis or proofs. Nevertheless, our observations highlight many directions for future research, such as exploring which matrix structures are most prone to low Kronecker-rank structure preservation.

Chapter 4

Wavefront control for adaptive optics

In this chapter, we examine a Kronecker structured, bound-constrained least squares optimization problem arising in the context of wavefront control for large-scale adaptive optics (AO) systems. We present a brief review of the field, formulate the wavefront control problem, and examine possible computationally efficient solutions from the mathematical perspective of exploiting the Kronecker structures appearing in the problem description. Note that this work only represents a preliminary study; the numerical examples used to evaluate the performance of the proposed approaches are only partially realistic, thus their feasibility and the results should be interpreted with caution.

4-1 Background

This presentation on the background of adaptive optics systems and the corresponding wavefront control problem is a condensed version of our discussion in the Literature Survey [43]; for a more comprehensive overview, we also refer to the very informative lecture notes on the subject by Verhaegen, Vdovin, and Soloviev [1].

Adaptive optics makes it possible for ground-based telescopes to achieve diffraction limited resolutions comparable to those of space telescopes. This is accomplished by compensating for imaging errors caused by turbulence within the atmosphere. Building and maintaining complex optical instruments on our planet is much more cost-effective than doing so in space, thus advancing the AO technology to allow wavefront corrections for the increasingly large-scale telescopes necessary to achieve high spatial resolution images of scientific value is crucial in order to maintain their competitiveness and significance [44]. This gives incentive to study and improve wavefront control algorithms, aiming to reduce their computational complexity and thus allow their implementation in real-time. In this section, we explain the basic principles of AO and review a state-of-the-art Kronecker structured formulation for wavefront correction using minimum variance control (MVC). The notation used throughout is specific to this background presentation and will differ from the subsequent sections, in which potential solution approaches are proposed and analyzed from the more mathematical perspective of efficiently exploiting the structure of the control problem.

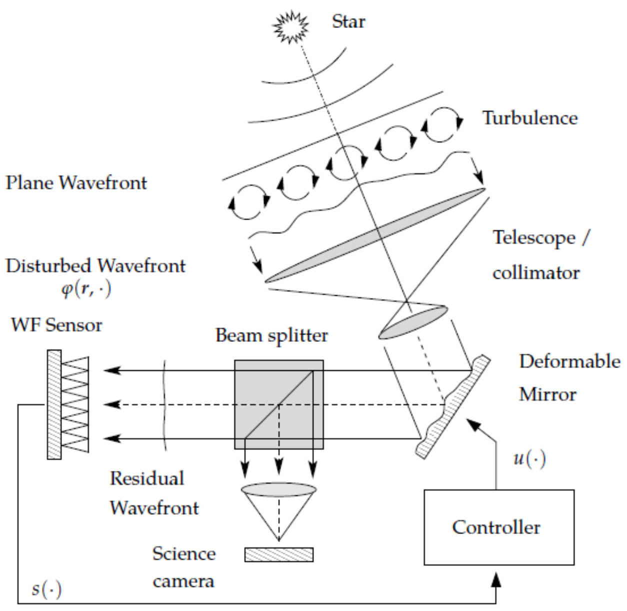


Figure 4-1: Simplified schematic of the imaging process of an AO system aided telescope [1].

4-1-1 Wavefront control

The schematic representation of an adaptive optics system for ground-based telescopes is depicted in Figure 4-1 above. The imaging process can be briefly explained as follows. Atmospheric turbulence introduces a phase distortion in the wavefronts of light coming from distant astronomical objects. This is an issue because the optical system and scientific camera are designed to produce sharp, high resolution images for flat wavefronts, not the disturbed and time-varying $\boldsymbol{\phi}_{tur}(\boldsymbol{r}, t)$ entering the telescope. The goal of an AO system is to eliminate imaging errors stemming from the atmospheric aberrations using *wavefront control.* To this end, a correction $\boldsymbol{\phi}_{DM}(\boldsymbol{r}, t)$ is applied using a deformable mirror (DM), while the gradient residual wavefront $\boldsymbol{\phi}_{res} = \boldsymbol{\phi}_{tur} - \boldsymbol{\phi}_{DM}$ is measured using a wavefront sensor (WFS). For example, with a Shack-Hartmann (SH) sensor, the $s(t)$ slopes corresponding to the x and y directional gradients of the wavefront plane are measured on a discretized lenslet grid. The DM can be a continuous sheet whose shape is altered by poking it with a grid of actuators.

One of the most promising methods for wavefront control is minimum variance control in the discrete-time framework, taking the finite control loop frequency into account [45]. Traditionally, it is accomplished through the course of two separate steps. First, the residual phase $\boldsymbol{\phi}_{res}(k)$ at time step k is reconstructed from the $\boldsymbol{s}(k)$ slope measurements. Filtering techniques are employed in order to take the dynamics and stochastic properties of atmospheric turbulence into account and provide a minimum variance prediction $\hat{\boldsymbol{\phi}}_{tur}(k+1|k)$ of the turbulent wavefront $\boldsymbol{\phi}_{tur}(k+1)$ at the next time step. Second, the $u(k)$ actuator commands are chosen such that the expected residual $\hat{\boldsymbol{\phi}}_{res}(k+1|k) = \hat{\boldsymbol{\phi}}_{tur}(k+1|k) - \boldsymbol{\phi}_{DM}(k+1)$ is minimized. Here it is assumed that the phase correction depends linearly on $u(k)$, i.e. the dynamics of the DM are neglected and $\boldsymbol{\phi}_{DM}(k+1) = \mathbf{H}\boldsymbol{u}(k)$, which allows its determination through a linear least squares problem. Efficient solutions exploiting the sparsity of the so-called influence matrix $\mathbf{H}$ have been proposed and analyzed in detail in the work of Konnik [46].

A novel approach based on the work of Sinquin and Verhaegen [10] aims to formulate the control law by minimizing the measured slopes, as these would also be zero if the residual wavefront were flat. The turbulence dynamics and the relation between the actuator inputs and the slopes on a rectangular grid are described by a so-called Kronecker VARX model, which takes the following form:

$$\boldsymbol{s}(k+1) = \sum_{d=1}^{N_F} \mathbf{F}_d \boldsymbol{s}(k-d+1) - \sum_{d=1}^{N_G} \mathbf{G}_d \boldsymbol{u}(k-d+1) + \boldsymbol{w}(k). \tag{4-1}$$

Here the coefficient matrices $\mathbf{F}_d, \mathbf{G}_d$ have block-wise low Kronecker rank representations:

$$\mathbf{F}_d = \begin{bmatrix} \sum_{j=1}^{F_d^x} \mathbf{F}_{d,j,L}^x \otimes \mathbf{F}_{d,j,R}^x & \mathbf{0} \\ \mathbf{0} & \sum_{j=1}^{F_d^y} \mathbf{F}_{d,j,L}^y \otimes \mathbf{F}_{d,j,R}^y \end{bmatrix} \quad \text{and} \quad \mathbf{G}_d = \begin{bmatrix} \sum_{j=1}^{G_d^x} \mathbf{G}_{d,j,L}^x \otimes \mathbf{G}_{d,j,R}^x \\ \sum_{j=1}^{G_d^y} \mathbf{G}_{d,j,L}^y \otimes \mathbf{G}_{d,j,R}^y \end{bmatrix} \tag{4-2}$$

in a partitioning according to the x and y slope measurements. The error $\boldsymbol{w}(k)$ is a zero-mean Gaussian white noise with positive definite covariance matrix[1] $\mathbf{C}_w$; it can alternatively be represented as $\mathbf{C}_w^{1/2}\boldsymbol{w}_e(k)$ where $\boldsymbol{w}_e(k)$ has identity covariance and $\mathbf{C}_w^{1/2}$ is the unique positive definite square root of $\mathbf{C}_w$. A framework for identifying models with such structure using input-output data has been discussed in the previous work [27].

Let us denote the unbiased estimate of $\boldsymbol{s}(k+1)$ corresponding to the case of applying no actuation in the kth time step by $\hat{\boldsymbol{s}}_0(k+1)$. Using the Kronecker VARX model (4-1), it can be expressed as:

$$\hat{\boldsymbol{s}}_0(k+1) = \sum_{d=1}^{n_F} \mathbf{F}_d \boldsymbol{s}(k-d+1) - \sum_{d=2}^{N_G} \mathbf{G}_d \boldsymbol{u}(k-d+1). \tag{4-3}$$

In order to zero out the slope measurements $\boldsymbol{s}(k+1)$, from (4-1) we have the condition

$$\hat{\boldsymbol{s}}_0(k+1) - \mathbf{G}_1\boldsymbol{u}(k) + \mathbf{C}_w^{1/2}\boldsymbol{w}_e(k) = \mathbf{0}, \tag{4-4}$$

which we aim to satisfy with error $\boldsymbol{w}_e(k)$ of the smallest possible norm. This gives rise to the following weighted[2] linear least squares problem for the unknown actuator commands [47]:

$$\min_{\boldsymbol{u}(k)} \left\| \hat{\boldsymbol{s}}_0(k+1) - \mathbf{G}_1\boldsymbol{u}(k) \right\|_{\mathbf{C}_w^{-1}}. \tag{4-5}$$

We augment this optimization problem by also adding regularization and including bound constraints on $\boldsymbol{u}(k)$. The former allows us to express a trade-off between accuracy and actuation power consumption. The latter has been shown to be important to take into consideration due to actuator saturation, especially for strong turbulences [48]; it also allows us to cope with dead actuators and non-rectangular actuator/sensor grids, as will be discussed later. The final form of the control law is thus given as the following quadratic optimization problem:

$$\begin{aligned} \min_{\boldsymbol{u}(k)} \quad & \left\| \mathbf{C}_w^{-1/2}\hat{\boldsymbol{s}}_0(k+1) - \mathbf{C}_w^{-1/2}\mathbf{G}_1\boldsymbol{u}(k) \right\|^2 + \lambda \left\| \boldsymbol{u}(k) \right\|^2 \\ \text{s.t.} \quad & \boldsymbol{b}_l \leq \boldsymbol{u}(k) \leq \boldsymbol{b}_u. \end{aligned} \tag{4-6}$$

[1]The covariance of a stochastic variable $\boldsymbol{w}(k)$ is defined as the expectation $E[\boldsymbol{w}(k)\boldsymbol{w}(k)^{\mathrm{T}}]$.

[2]The weighted norm induced by a symmetric positive definite matrix $\mathbf{A}$ is defined as $\|\boldsymbol{x}\|_{\mathbf{A}} = \|\mathbf{A}^{1/2}\boldsymbol{x}\|_2$.

The advantage of this formulation of the wavefront control law is twofold in terms of possible computational efficiency. First, by directly modeling the dynamics of the slope measurements, wavefront reconstruction and filtering can be avoided, though the sub-optimality of this scheme compared to the traditional two step approach is yet to be evaluated. Second, the Kronecker structured model allows fast evaluation of the slope prediction $\hat{\boldsymbol{s}}_0(k+1)$, and, as is the topic of this chapter, may also accelerate the solution of (4-6) for the optimal control inputs. On the other hand, the disadvantage in this case is that the Kronecker representation, by nature, requires circular DM and WFS arrays to be embedded in rectangular grids. This introduces extra 'virtual' slope measurements and actuators that have to be handled with care and also increases the problem size compared to e.g. a sparse representation.

To fully maintain the efficiency of the calculations, however, it is important for all the matrices in the control law to admit a Kronecker representation. Let us partition $\mathbf{C}_w^{-1/2}$ in a block form according to the x and y slope measurements:

$$\mathbf{C}_w^{-1/2} = \begin{bmatrix} \mathbf{C}^x & \mathbf{C}^{xy} \\ \mathbf{C}^{yx} & \mathbf{C}^y \end{bmatrix} \tag{4-7}$$

with

$$\mathbf{C}^x = \sum_{j=1}^{M_w^x} \mathbf{C}_{j,L}^x \otimes \mathbf{C}_{j,R}^x \qquad \text{and} \qquad \mathbf{C}^y = \sum_{j=1}^{M_w^y} \mathbf{C}_{j,L}^y \otimes \mathbf{C}_{j,R}^y; \tag{4-8}$$

the cross-covariances $\mathbf{C}^{xy} = (\mathbf{C}^{yx})^{\mathrm{T}}$ are assumed to be negligible. With this structure, the coefficient matrix $\mathbf{C}_w^{-1/2}\mathbf{G}_1$ in problem (4-6) can be written as[3]:

$$\mathbf{C}_w^{-1/2}\mathbf{G}_1 = \begin{bmatrix} \sum_{j_1=1}^{M_w^x} \sum_{j_2=1}^{M_1^x} \left(\mathbf{C}_{j_1,L}^x \mathbf{G}_{1,j_2,L}^x\right) \otimes \left(\mathbf{C}_{j_1,R}^x \mathbf{G}_{1,j_2,R}^x\right) \\ \sum_{j_1=1}^{M_w^y} \sum_{j_2=1}^{M_1^y} \left(\mathbf{C}_{j_1,L}^y \mathbf{G}_{1,j_2,L}^y\right) \otimes \left(\mathbf{C}_{j_1,R}^y \mathbf{G}_{1,j_2,R}^y\right) \end{bmatrix} := \begin{bmatrix} \sum_{j=1}^{M^x} \mathbf{B}_j^x \otimes \mathbf{C}_j^y \\ \sum_{j=1}^{M^y} \mathbf{B}_j^x \otimes \mathbf{C}_j^y \end{bmatrix} \tag{4-9}$$

with the Kronecker ranks $M^x = M_w^x \cdot M_1^x$ and $M^y = M_w^y \cdot M_1^y$. This form will maintain computational efficiency in case these ranks are small. In theory, it has been shown that $M_1^x = 1$ and $M_1^y = 1$ for actuators and sensors arranged in a rectangular grid [10]. This is connected with how the effect of poking the actuators of a DM can be described by a separable (Gaussian) function in the x and y directions. Even though there will be small magnitude correction terms due to imperfections in practice, we can assume the low Kronecker ranks will be retained. A recent study has also revealed that the covariance matrices corresponding to atmospheric turbulence models also admit low Kronecker rank representations[9]; furthermore, their Kronecker rearrangements are shown to exhibit a decaying spectrum. It is our hope that $\mathbf{C}^x$ and $\mathbf{C}^y$, and thus the final Kronecker sums will retain this favorable property (see the discussion of Section 3-1).

Remark. *Additional constraints on the actuator commands, such as preventing large spatial or temporal variations for safe operation, are also important to consider when formulating the control law [46]. However, their inclusion is outside the scope of this thesis work and is left for future exploration.*

[3]If the cross-covariances are not neglected, we would have the same form but with higher Kronecker ranks.

4-2 Problem formulation

We will now give the mathematical formulation of the AO control law (4-6) that will be studied throughout the rest of this chapter. The notation is slightly changed to lighten the equations and conform with the previous chapters. To this end, we represent the unknown actuators commands at a given time step k by $\boldsymbol{x} := \boldsymbol{u}(k)$, while the weighted left hand side becomes $\boldsymbol{t} := \mathbf{C}_w^{-1/2}\hat{\boldsymbol{s}}_0(k+1)$. We also assume the two blocks of the coefficient matrix (4-9) to have the same Kronecker rank M. The AO-control problem is then formulated as follows.

Problem 4.1 (AO-control). Given the slope predictions $\boldsymbol{t}^x, \boldsymbol{t}^y \in \mathbb{R}^{m^2}$, the upper and lower actuator input bounds $\boldsymbol{b}^u, \boldsymbol{b}^l \in \mathbb{R}^{n^2}$, and the system description $\mathbf{B}_j^x, \mathbf{C}_j^x, \mathbf{B}_j^y, \mathbf{C}_j^y \in \mathbb{R}^{m\times n}$ for $j = 1, \ldots, M$ in low Kronecker rank representation, solve

$$\begin{aligned}\min_{\boldsymbol{x}} F(\boldsymbol{x}) &= \left\| \begin{bmatrix} \boldsymbol{t}^x \\ \\ \boldsymbol{t}^y \end{bmatrix} - \begin{bmatrix} \sum_{j=1}^{M} \mathbf{B}_j^x \otimes \mathbf{C}_j^x \\ \sum_{j=1}^{M} \mathbf{B}_j^y \otimes \mathbf{C}_j^y \end{bmatrix} \cdot \boldsymbol{x} \right\|^2 + \lambda \left\|\boldsymbol{x}\right\|^2 \\ \text{s.t. } & \boldsymbol{b}^l \le \boldsymbol{x} \le \boldsymbol{b}^u \end{aligned} \tag{4-10}$$

for the unknown $\boldsymbol{x} \in \mathbb{R}^{n^2}$ actuator inputs, where $\lambda > 0$ is a regularization parameter.

Assumption. The sensor array is denser than the actuator array, i.e. $m > n$ such that each $\mathbf{B}_j^x, \mathbf{C}_j^x, \mathbf{B}_j^y, \mathbf{C}_j^y$ are full column rank. Furthermore, the Kronecker ranks are low compared to the problem size, i.e. $M \ll n$.

The assumption guarantees that the solution to the AO-control problem is unique even without regularization, and will be satisfied for the test laboratory data used to verify the proposed solutions. Typical values for the n^2 number of unknown actuator commands can be up to the order of a thousands or tens of thousands for next generation telescopes such as the European Extremely Large Telescope (E-ELT) [44]. The main challenge of this work is to provide an efficient solution to the Kronecker structured AO-control problem, working towards the direction of allowing the target 1 kHz control loop frequencies [49] to be implemented in real-time for such large-scale systems.

In our discussion, we will mostly use the form (4-10) when analyzing the performance of possible solutions in detail. For higher-level descriptions, it will be more convenient to use the following shorthand notation:

$$\boldsymbol{t} = \begin{bmatrix} \boldsymbol{t}^x \\ \boldsymbol{t}^y \end{bmatrix}, \quad \mathbf{A} = \begin{bmatrix} \mathbf{A}^x \\ \\ \mathbf{A}^y \end{bmatrix} = \begin{bmatrix} \sum_{j=1}^{M} \mathbf{B}_j^x \otimes \mathbf{C}_j^x \\ \sum_{j=1}^{M} \mathbf{B}_j^y \otimes \mathbf{C}_j^y \end{bmatrix}, \text{ and } \quad \mathcal{B} = \left\{ \boldsymbol{x} \in \mathbb{R}^{n^2} \;\middle|\; \boldsymbol{b}^l \le \boldsymbol{x} \le \boldsymbol{b}^u \right\}, \tag{4-11}$$

with $\boldsymbol{t} \in \mathbb{R}^{2m^2}$, $\mathbf{A}^x, \mathbf{A}^y \in \mathbb{R}^{m^2 \times n^2}$, and $\mathbf{A} \in \mathbb{R}^{(2m^2)\times n^2}$. These notations allow Problem 4.1 to be rewritten compactly as:

$$\begin{aligned} \min_{\boldsymbol{x}} \; & F(\boldsymbol{x}) = \left\|\boldsymbol{t} - \mathbf{A}\boldsymbol{x}\right\|^2 + \lambda \left\|\boldsymbol{x}\right\|^2 \\ \text{s.t. } & \boldsymbol{x} \in \mathcal{B}. \end{aligned} \tag{4-12}$$

4-2-1 Solution approaches

The most important points to keep in mind when solving Problem 4.1 is that any proposed efficient solution algorithm should possess the following key features:

- It must be effective for solving a series of consecutive problems with small deviations (known as *warm-start efficiency*), since the slope measurements are expected to change gradually in each time step of the control loop.
- It must exploit both the Kronecker and possibly sparse (banded) structure of the system matrices.

Based on the discussion presented in the Literature Survey [43] and the analysis of warm-start efficiencies of quadratic programming algorithms by Konnik and De Doná [50], in this thesis we have chosen to examine the following three solution approaches: alternating direction method of multipliers (ADMM), projected alternating Barzilai-Borwein (PABB), and active sets (AS) methods. For each of these approaches, we will comment on how the performance can be expected to compare with a corresponding purely sparse solution, such as the ones examined in the dissertation of Konnik [46].

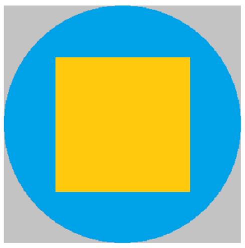
Remark. *Interior point (IP) methods were also considered, but it proved difficult to exploit the Kronecker structures in its framework, as will be noted in our discussion of the AS approach. Furthermore, as they are also known to have difficulties in utilizing warm-starts, we will not analyze interior point (IP) methods in detail.*

4-2-2 Data for numerical study and performance evaluation

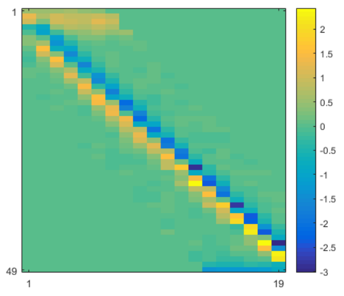
In the course of this thesis, we planned on testing the performance of proposed solutions to Problem 4.1 based on realistic AO system and slope measurement data from a laboratory setup. Unfortunately, in the end this was not possible, as only the static influence matrix $\mathbf{G}_1$ from the Kronecker VARX model (4-1) was identified in a reliable manner. In order to avoid issues from virtual actuators and sensors, the identification was performed for a rectangular embedding within the circular devices, as shown on Figure 4-2. The grid consists of 19×19 actuators and 49×49 sensor lenslets, i.e. we have the dimensions $n = 19$ and $m = 49$. The Kronecker rank of the identified matrices is $M = 3$. The numerical study thus only uses the realistic $\mathbf{G}_1$ influence matrix as the (4-9) coefficient matrix of the AO-control problem; realistic covariance information is not considered. We also did not add regularization on top of the bound constraints, though the solutions show how this could be handled.

For the weighted $\boldsymbol{t}$ slope predictions, we simply use slope measurements from an in-house turbulence generation toolbox, simulated for 100 consecutive time steps. The slopes were then scaled such that actuator input bounds of ± 1 units would lead to saturation rates commonly seen for moderate turbulences. For an AO system, the number of saturated actuators for such a case is roughly 5-10 % of the total [46]; with our data, the average and standard deviation for the 100 time steps is 6.5 ± 2.4 %[4]. We also note that the simulated data leads to an average number of 15.4 ± 4.3 actuators becoming (un)saturated from one time step to the next, which is also not realistic for high control frequencies and should be much less. Our discussion thus mainly serves to examine potential solutions, *not* to determine their feasibility in practice.

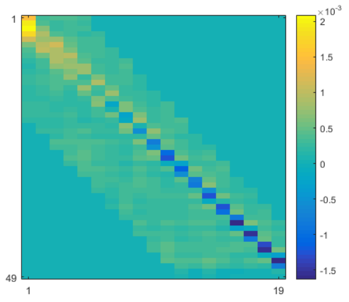
[4]The performance of algorithms can depend heavily on the saturation rate, which we thus aimed to keep consistent for the purpose of this thesis.



(a)



(b) $\mathbf{B}_1^x$



(c) $\mathbf{C}_1^x$

Figure 4-2: (a) Identified region of the laboratory DM (yellow). The blue region corresponds to available actuators, while the grey represents virtual actuators which would be needed for a full Kronecker structured embedding. (b),(c) Sample identified system matrices.

4-3 ADMM solution

The first solution approach to Problem 4.1 is based on an algorithm called the alternating direction method of multipliers. ADMM is useful for minimizing functions with separable additive convex terms, a framework which also allows handling constraints such as variable bounds. The optimization problem is thus decomposed into simpler subproblems which can be solved in a more efficient manner. The potential of ADMM for real-time application in AO has been verified by Konnick and De Doná [50], where speed-up factors of around 1.4-1.7 were achieved using warm-start. The method is known to have a fast initial convergence, which is often enough to yield an acceptable approximate solution within the limited time available for control calculations; slower convergence near the optimum is thus not necessarily an issue. For a brief review of ADMM, and its symmetric variant used in this section, we refer to Appendix B. More detailed discussions, including theoretical results on convergence rates and possible accelerations, are available in the literature [29], [30].

In this section, we present two possible ADMM formulations to tackle the AO-control problem. The first relies on the traditional 2-split form, where the objective and constraints (including regularization) are separated, leading to an iterative solution with two subproblems. In the second scheme, we further divide the objective along the x and y slope measurement in hopes of possibly improving performance, which results in a 3-split form. For both cases, we will present the respective ADMM iterations, discuss the solution to each of the subproblems, and evaluate the achieved performance using the realistic model of the laboratory DM and our simulated slope measurement data.

For each of the formulations, we will encounter two types of subproblems: least squares problems involving rows of sums of Kronecker products, and projections onto the feasible bound set $\mathcal{B}$. For the former, we refer to the discussion about the MLSQR algorithm in Section 3-2-2. For the latter, we note that the projection of a vector $\boldsymbol{v}$ onto box inequality sets is a very cheap element-wise operation and can be evaluated simply as:

$$[\Pi_{\mathcal{B}}(\boldsymbol{v})]_i = \begin{cases} b_i^l, & v_i \leq b_i^l \\ v_i, & b_i^l \leq v_i \leq b_i^u \\ b_i^u, & v_i \geq b_i^u. \end{cases} \tag{4-13}$$

4-3-1 2-split approach

Let us reformulate Problem 4.1 to conform with the traditional 2-split ADMM scheme. To simplify the expressions, we will work with the shorthand notation (4-12) and denote the indicator function[5] of the box constraints by $\mathcal{I}_{\mathcal{B}}$. Written as a global variable consensus with regularization problem, the ADMM formulation (B-2) for the AO-control problem takes the form:

$$\begin{aligned} \min_{\boldsymbol{x}_1,\boldsymbol{z}} \quad & \overbrace{\left(\|\boldsymbol{t}-\mathbf{A}\boldsymbol{x}_1\|^2\right)}^{f_1(\boldsymbol{x}_1)} + \overbrace{\left(\lambda\|\boldsymbol{z}\|^2 + \mathcal{I}_{\mathcal{B}}(\boldsymbol{z})\right)}^{g(\boldsymbol{z})} \\ \text{s.t.} \quad & \boldsymbol{x}_1 = \boldsymbol{z}. \end{aligned} \tag{4-14}$$

The corresponding ADMM iterations in the scaled form (B-5) thus become:

$$\boldsymbol{x}_1^{(l+1)} := \arg\min_{\boldsymbol{x}_1} \left(\|\boldsymbol{t}-\mathbf{A}\boldsymbol{x}_1\|^2 + \frac{\rho}{2}\left\|\boldsymbol{x}_1 - \boldsymbol{z}^{(l)} + \boldsymbol{u}_1^{(l)}\right\|^2\right) \tag{4-15a}$$

$$\boldsymbol{z}^{(l+1)} := \arg\min_{\boldsymbol{z}\in\mathcal{B}} \left(\lambda\|\boldsymbol{z}\|^2 + \frac{\rho}{2}\left\|\boldsymbol{z} - \boldsymbol{x}_1^{(l+1)} - \boldsymbol{u}_1^{(l)}\right\|^2\right) \tag{4-15b}$$

$$\boldsymbol{u}_1^{(l+1)} := \boldsymbol{u}_1^{(l)} + \boldsymbol{x}_1^{(l+1)} - \boldsymbol{z}^{(l+1)}. \tag{4-15c}$$

Note that we have a single local variable $\boldsymbol{x}_1 \in \mathbb{R}^{n^2}$, it's corresponding dual variable $\boldsymbol{u}_1 \in \mathbb{R}^{n^2}$, and the shared global variable $\boldsymbol{z} \in \mathbb{R}^{n^2}$ which will serve as the consensus solution. We will now examine the computational aspects of each of the variable update steps in detail.

Iteration updates

- **x-updates:** The two squared norms in the objective function (4-15a) can be combined under a single one. Substituting in the expanded forms (4-11) of $\boldsymbol{t}$ and $\mathbf{A}$, this yields the following optimization problem:

$$\min_{\boldsymbol{x}_1} \left\| \begin{bmatrix} \boldsymbol{t}^x \\ \boldsymbol{t}^y \\ \nu\left(\boldsymbol{z}^{(l)} - \boldsymbol{u}_1^{(l)}\right) \end{bmatrix} - \begin{bmatrix} \sum_{j=1}^M \mathbf{B}_j^x \otimes \mathbf{C}_j^x \\ \sum_{j=1}^M \mathbf{B}_j^y \otimes \mathbf{C}_j^y \\ \nu\mathbf{I}_{n^2} \end{bmatrix} \boldsymbol{x}_1 \right\|^2 := \left\|\boldsymbol{t}_{\text{aug}} - \mathbf{A}_{\text{aug}}\boldsymbol{x}_1\right\|^2, \tag{4-16}$$

  where the parameter $\nu = \sqrt{\rho/2}$. This least squares problem has the same form as the one examined in Section 3-2-2, as the coefficient matrix is composed of rows of sums of Kronecker products. In order to make use this structure in a computationally efficient manner, we will employ the MLSQR algorithm discussed therein. This iterative solver relies on matrix-vector products, thus any sparsity of the Kronecker terms is naturally exploited. A preconditioner for the least squares problem can be obtained by finding an approximation $\mathbf{M} \approx (\mathbf{A}_{\text{aug}}^{\mathrm{T}}\mathbf{A}_{\text{aug}})^{-1}$ in low rank Kronecker form using Algorithm 2.1 (Kronecker-LS), which is applicable as $\mathbf{A}_{\text{aug}}^{\mathrm{T}}\mathbf{A}_{\text{aug}}$ can be written as the Kronecker sum:

$$\sum_{j_1=1}^{M}\sum_{j_2=1}^{M}\left[\left(\left(\mathbf{B}_{j_1}^x\right)^{\mathrm{T}}\mathbf{B}_{j_2}^x\right)\otimes\left(\mathbf{C}_{j_1}^{x\ \mathrm{T}}\mathbf{C}_{j_2}^x\right) + \left(\mathbf{B}_{j_1}^{y\ \mathrm{T}}\mathbf{B}_{j_2}^y\right)\otimes\left(\left(\mathbf{C}_{j_1}^y\right)^{\mathrm{T}}\mathbf{C}_{j_2}^y\right)\right] + (\nu\mathbf{I}_n)\otimes(\nu\mathbf{I}_n). \tag{4-17}$$

[5]The indicator function of a constraint returns 0 in case it is satisfied and $+\infty$ otherwise.

The preconditioner is thus obtained in the form:

$$\mathbf{M} = \sum_{i=1}^{N} \mathbf{M}_{L,i} \otimes \mathbf{M}_{R,i}, \tag{4-18}$$

where we also impose symmetry and a sparse pattern on the terms $\mathbf{M}_{L,i}, \mathbf{M}_{R,i} \in \mathbb{R}^{n\times n}$. The main computational cost of the MLSQR algorithm is two consecutive matrix-vector multiplications by $\mathbf{A}_{\text{aug}}^{\mathrm{T}}$ and $\mathbf{M}$ during initialization, followed by a series of multiplications by $\mathbf{A}_{\text{aug}}$, $\mathbf{A}_{\text{aug}}^{\mathrm{T}}$, and $\mathbf{M}$ in each iteration.

- **z-updates:** The update (4-15a) for the shared variable can be reformulated as the following proximal minimization problem [43]:

$$\boldsymbol{z}^{(l+1)} = \arg\min \left(\mathcal{I}_{\mathcal{B}}(\boldsymbol{z}^{(l+1)}) + \frac{\tilde{\rho}}{2} \left\| \boldsymbol{z}^{(l+1)} - \tilde{\boldsymbol{z}}^{(l+1)} \right\|^2 \right), \tag{4-19}$$

where the introduced $\tilde{\rho}/2 = \lambda + \rho/2$ and $\tilde{\boldsymbol{z}}^{(l+1)}$ is given by the weighted sum:

$$\tilde{\boldsymbol{z}}^{(l+1)} := \frac{\rho/2}{\lambda + \rho/2} (\boldsymbol{x}_1^{(l+1)} + \boldsymbol{u}_1^{(l)}). \tag{4-20}$$

The solution is then found by a simple projection onto the box inequality constraints:

$$\boldsymbol{z}^{(l+1)} = \Pi_{\mathcal{B}} \left(\tilde{\boldsymbol{z}}^{(l+1)} \right). \tag{4-21}$$

The steps constitute a negligible $\mathcal{O}(n^2)$ flops.

- **scaled dual updates:** The vector additions and subtraction required for the scaled dual update require only $\mathcal{O}(n^2)$ operations.

The symmetric version of these iteration updates are summarized as the 2-split S-ADMM algorithm below.

Algorithm 4.1 (2-split S-ADMM) Symmetric 2-split ADMM solution to Problem 4.1

```
   Input:
      Problem description $\mathbf{A}$, $\boldsymbol{t}$, $\lambda$, $\mathcal{B}$; parameter $\rho$; stopping criteria $\epsilon^{\text{rel}}$, $\epsilon^{\text{abs}}$
      Initial estimates[6] $\boldsymbol{u}_1^{(0)}$ and $\boldsymbol{x}_1^{(0)} = \boldsymbol{z}^{(0)}$
1: for $l = 0, 1, 2, \dots$ do
2:    Solve $\boldsymbol{x}_1^{(l+1)} := \arg\min_{\boldsymbol{x}_1} \left( \|\boldsymbol{t} - \mathbf{A}\boldsymbol{x}_1\|^2 + \frac{\rho}{2} \left\| \boldsymbol{x}_1 - \boldsymbol{z}^{(l)} + \boldsymbol{u}_1^{(l)} \right\|^2 \right)$ using MLSQR
3:    $\boldsymbol{u}_1^{(l+1/2)} := \boldsymbol{u}_1^{(l)} + \boldsymbol{x}_1^{(l+1)} - \boldsymbol{z}^{(l)}$
4:    $\boldsymbol{z}^{(l+1)} := \Pi_{\mathcal{B}} \left( \frac{\rho/2}{\lambda + \rho/2} (\boldsymbol{x}_1^{(l+1)} + \boldsymbol{u}_1^{(l+1/2)}) \right)$
5:    $\boldsymbol{u}_1^{(l+1)} := \boldsymbol{u}_1^{(l+1/2)} + \boldsymbol{x}_1^{(l+1)} - \boldsymbol{z}^{(l+1)}$
6:    Check stopping criteria.
7: end for
```

[6] In case of solving consecutive problems, these are set from the previous solution to achieve warm-start.

Fixing the penalty parameter The preconditioner $\mathbf{M}$ for the MLSQR algorithm in the x-updates should be pre-calculated in order to retain the efficiency of the solution approach. This requires the penalty parameter ρ to be kept constant throughout the ADMM iterations[7]. We thus need to choose its value in a way that leads to the minimal number of iterations needed for convergence. To this end, we solved the 100 sample problems of the AO numerical dataset using the 2-split S-ADMM algorithm with various ρ parameters. In each case, we recorded the average and standard deviation of the $L_{\text{2-ADMM}}$ number of iterations needed for convergence[8], along with the achieved relative objective and solution errors defined as:

$$\epsilon_{\text{obj}}^{\text{rel}} := \frac{F(\boldsymbol{z}^L) - F(\boldsymbol{z}^*)}{F(\boldsymbol{z}^*)} \qquad \text{and} \qquad \epsilon_{\text{sol}}^{\text{rel}} := \frac{\left\| \boldsymbol{z}^L - \boldsymbol{z}^* \right\|}{\left\| \boldsymbol{z}^* \right\|}, \tag{4-22}$$

where $\boldsymbol{z}^{L_{\text{2-ADMM}}}$ is the solution returned after the last iteration and $\boldsymbol{z}^*$ is the true optimum computed using CVX [51].

The results of this study are summarized in Table 4-1. For stopping criteria, we used the common rule of requiring the norm of the primal and scaled dual residuals (B-7) to drop below given threshold values defined by the relations in (B-8). We found the absolute and relative parameters $\epsilon^{\text{abs}} = \epsilon^{\text{rel}} = 10^{-2}$ produce results with acceptable accuracy; the optimal value of the objective is always achieved with negligible relative error. The average number of ADMM iterations are minimized at approximately $\rho = 0.02$. The evolution of the objective function in the optimization problems for the first two time steps of the AO numerical data with this choice are shown in a latter comparison on Figure 4-6.

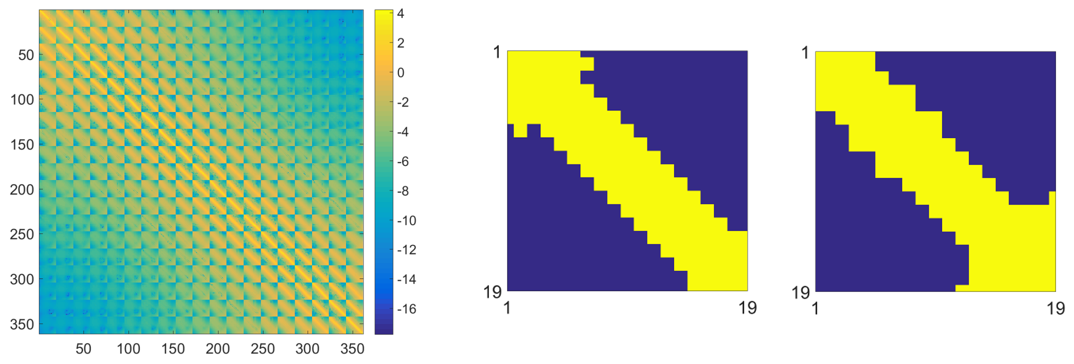
Remark. *It is interesting to note that the standard deviation of $L_{\text{2-ADMM}}$ becomes essentially zero for $\rho = 0.02$, which implies that the ADMM algorithm converges in 4 iterations for all consecutive problems. Such consistency is a very favorable property for real-time execution.*

Table 4-1: Effect of the penalty parameter ρ in the 2-split S-ADMM scheme on the number of iterations needed to solve the 100 consecutive AO control problems of the sample numerical dataset. The achieved relative errors of the objective and solution are also displayed. For each case, the averages are given along with the corresponding standard deviations. The optimal choice of the penalty parameter is highlighted in bold.

ρ	$L_{\text{2-ADMM}}$	$\epsilon_{\text{obj}}^{\text{rel}}$ $[\cdot 10^3\ \%]$	$\epsilon_{\text{sol}}^{\text{rel}}$ $[\%]$
0.008	7.2 ± 0.4	6.4 ± 1.6	0.83 ± 0.22
0.012	5.1 ± 0.3	4.6 ± 1.5	1.19 ± 1.01
0.020	$\mathbf{4.0 \pm 0.0}$	$\mathbf{7.2 \pm 8.6}$	$\mathbf{2.90 \pm 2.36}$
0.030	4.1 ± 0.3	22.1 ± 13.5	4.81 ± 2.72
0.050	5.1 ± 0.3	60.5 ± 24.4	7.20 ± 3.32

[7]Alternatively, a set of preconditioners could be pre-calculated for a series of penalty parameters, but exploring this possibility is outside the scope of this work.

[8]The iterations for solving the first time step do not benefit from warm-start and were excluded from the statistics.



(a) **(b)** **(c)**

Figure 4-3: (a) Inverse of the matrix $\mathbf{A}_{\text{aug}}^{\mathrm{T}}\mathbf{A}_{\text{aug}}$ on a logarithmic scale; (b),(c) the corresponding sparsity patterns chosen for the preconditioner Kronecker terms $\mathbf{M}_{L,i}$ and $\mathbf{M}_{R,i}$ in (4-18). They have 131 and 141 non-zero elements (in yellow), respectively – fill-ins of $36.3\,\%$ and $39.1\,\%$.

Choosing a preconditioner Having found the optimal penalty parameter, the coefficient matrix $\mathbf{A}_{\text{aug}}$ of the x-updates (4-15a) becomes fixed. The next step towards an efficient ADMM solution is choosing an effective sparsity pattern on the preconditioner (4-18). This will further accelerate the matrix-vector multiplications required by the MLSQR algorithm on top of exploiting the low Kronecker rank structure itself.

The inverse of $\mathbf{A}_{\text{aug}}^{\mathrm{T}}\mathbf{A}_{\text{aug}}$ and the chosen sparsity patterns for the Kronecker terms of the preconditioner are depicted in Figure 4-3. The latter were obtained for each element of the matrices individually by first averaging the absolute values of the elements of $(\mathbf{A}_{\text{aug}}^{\mathrm{T}}\mathbf{A}_{\text{aug}})^{-1}$ that they influence. The results were then thresholded to allow non-zeros in the sparsity patterns of the Kronecker matrices only for matrix elements where this average reaches at least $4\,\%$ of the maximum among them all. For larger matrices, in case it is not feasible to explicitly calculate the inverse, an alternative method would be to determine a dense Kronecker preconditioner using full matrices and threshold its values instead. However, for an efficient calculation it would be best if the sparsity patterns were selected adaptively during the preconditioner approximation.

The relation between the preconditioner Kronecker rank N and the average number of MLSQR iterations needed to solve the x-updates for each ADMM iteration for the AO dataset is shown in Table 4-2[9]. The solution accuracy was controlled by setting the stopping criteria parameter $\epsilon_{\text{MLSQR}}^{\text{abs}} = 10^{-3}$. For ranks $N > 3$, the achievable performance becomes limited by the imposed sparsity pattern of the preconditioner, thus a reasonable choice is $N = 3$.

Table 4-2: Average number L_{MLSQR} of MLSQR iterations for solving the x-updates during each 2-split S-ADMM iteration as a function of the preconditioner's Kronecker rank N.

N	1	2	**3**	4	5
L_{MLSQR}	6.45 ± 0.48	3.96 ± 0.16	$\mathbf{3.04 \pm 0.19}$	3.00 ± 0.00	3.00 ± 0.00

[9]The effectiveness of the preconditioner may also depend on the penalty parameter, thus it might be worthwhile to optimize their selection together in order to minimize the total number of MLSQR iterations.

Performance The performance of the 2-split S-ADMM mainly depends on the number of *consecutive* matrix-vector multiplications with the coefficient matrix $\mathbf{A}_{\text{aug}}$ (or $\mathbf{A}_{\text{aug}}^{\mathrm{T}}$) and the preconditioner $\mathbf{M}$ during the MLSQR iterations for solving the x-updates. With $L_{\text{2-ADMM}}$ and L_{MLSQR} iterations of these respective algorithms, these numbers are $L_{\text{2-ADMM}}(1+2L_{\text{MLSQR}})$ and $L_{\text{2-ADMM}}(1+L_{\text{MLSQR}})$, respectively. In the context of the numerical study conducted with the AO dataset, this amounts to an average of around 28 and 16 multiplications for each time step, which have to be evaluated in series.

4-3-2 3-split approach

We will now examine an alternative ADMM formulation, a 3-split scheme for solving the AO control Problem 4.1. In comparison to the 2-split scheme, here we further divide the objective function along the slope residual norms corresponding to the x and y directions. Our hope is that this will lead to even simpler local variable updates with individually tailored preconditioners. Written as a global variable consensus with regularization problem, the ADMM formulation (B-2) for this scheme takes the form:

$$\begin{aligned}\min_{\boldsymbol{x}_1,\boldsymbol{x}_2,\boldsymbol{z}} \quad & \overbrace{\left(\|\boldsymbol{t}^x - \mathbf{A}^x\boldsymbol{x}_1\|^2\right)}^{f_1(\boldsymbol{x}_1)} + \overbrace{\left(\|\boldsymbol{t}^y - \mathbf{A}^y\boldsymbol{x}_2\|^2\right)}^{f_2(\boldsymbol{x}_2)} + \overbrace{\left(\lambda\|\boldsymbol{z}\|^2 + \mathcal{I}_{\mathcal{B}}(\boldsymbol{z})\right)}^{g(\boldsymbol{z})} \\ \text{s.t.} \quad & \boldsymbol{x}_i = \boldsymbol{z}, \qquad i = 1,2,\end{aligned} \tag{4-23}$$

and the resulting ADMM iterations in the scaled form (B-5) become[10]:

$$\boldsymbol{x}_i^{(l+1)} := \arg\min_{\boldsymbol{x}_i} \left(\left\|\boldsymbol{t}^{x/y} - \mathbf{A}^{x/y}\boldsymbol{x}_i\right\|^2 + \frac{\rho}{2}\left\|\boldsymbol{x}_i - \boldsymbol{z}^{(l)} + \boldsymbol{u}_i^{(l)}\right\|^2 \right) \tag{4-24a}$$

$$\boldsymbol{z}^{(l+1)} := \arg\min_{\boldsymbol{z}\in\mathcal{B}} \left(\lambda\|\boldsymbol{z}\|^2 + \sum_{i=1}^{2}\frac{\rho}{2}\left\|\boldsymbol{z} - \boldsymbol{x}_i^{(l+1)} - \boldsymbol{u}_i^{(l)}\right\|^2 \right) \tag{4-24b}$$

$$\boldsymbol{u}_i^{(l+1)} := \boldsymbol{u}_i^{(l)} + \boldsymbol{x}_i^{(l+1)} - \boldsymbol{z}^{(l+1)}. \tag{4-24c}$$

As opposed to the 2-split approach, we now have a two local variables $\boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathbb{R}^{n^2}$, their respective dual variables $\boldsymbol{u}_1, \boldsymbol{u}_2 \in \mathbb{R}^{n^2}$, and the shared global variable $\boldsymbol{z} \in \mathbb{R}^{n^2}$ which serves as the consensus solution. We will now go through the same procedure of examining the computational aspects of the variable updates and tuning the resulting algorithm for the sample numerical data of AO control problems in a similar manner as before.

Iteration updates

- **x-updates:** The two squared norms in the objective function (4-24a) can again be combined under a single one. Let us consider the $\boldsymbol{x}_1$ updates corresponding to the x slopes for simplicity; the case for $\boldsymbol{x}_2$ would follow the same derivation. Substituting in the expanded forms (4-11) of $\boldsymbol{t}^x$ and $\mathbf{A}^x$, we have the following optimization problem:

$$\min_{\boldsymbol{x}_1} \left\| \begin{bmatrix} \boldsymbol{t}^x \\ \nu\left(\boldsymbol{z}^{(l)} - \boldsymbol{u}_1^{(l)}\right) \end{bmatrix} - \begin{bmatrix} \sum_{j=1}^{M} \mathbf{B}_j^x \otimes \mathbf{C}_j^x \\ \nu\mathbf{I}_{n^2} \end{bmatrix} \boldsymbol{x}_1 \right\|^2 := \left\|\boldsymbol{t}_{\text{aug}}^x - \mathbf{A}_{\text{aug}}^x\boldsymbol{x}_1\right\|^2, \tag{4-25}$$

[10]The notation $\boldsymbol{t}^{x/y}$ and $\mathbf{A}^{x/y}$ refers to $\boldsymbol{t}^x$, $\mathbf{A}^x$ for $i=1$ and $\boldsymbol{t}^y$, $\mathbf{A}^y$ for $i=2$ throughout this section.

where the parameter $\nu = \sqrt{\rho/2}$. This least squares problem again has the same form as the one examined in Section 3-2-2, as the coefficient matrix is composed of rows of sums of Kronecker products; it can be solved using the MLSQR algorithm. A notable difference compared to the 2-split case is that the preconditioner obtained by finding an approximation $\mathbf{M} \approx \left((\mathbf{A}^x_{\text{aug}})^{\mathrm{T}} \mathbf{A}^x_{\text{aug}} \right)^{-1}$ can be found more efficiently since $\mathbf{A}^{\mathrm{T}}_{\text{aug}}\mathbf{A}_{\text{aug}}$, expressed as:

$$\sum_{j=1}^{M} \left(\left(\mathbf{B}^x_{j_1}\right)^{\mathrm{T}} \mathbf{B}^x_{j_2} \right) \otimes \left(\mathbf{C}^{x\ \mathrm{T}}_{j_1} \mathbf{C}^x_{j_2} \right) + (\nu \mathbf{I}_n) \otimes (\nu \mathbf{I}_n)$$

only has a Kronecker rank of $(M+1)$ as opposed to (M^2+1). There might also be a larger decay in the Kronecker singular values, allowing a good preconditioner

$$\mathbf{M}^x = \sum_{i=1}^{N^x} \mathbf{M}^x_{L,i} \otimes \mathbf{M}^x_{R,i} \tag{4-26}$$

with possibly lower rank than before. Again, we also impose symmetry and a sparse pattern on the terms $\mathbf{M}^x_{L,i}, \mathbf{M}^x_{R,i} \in \mathbb{R}^{n \times n}$ when solving for the inverse approximation using the Kron-ALS algorithm.

The main computational cost of the MLSQR algorithm for the $\boldsymbol{x}_1$ updates are two consecutive matrix-vector multiplications by $(\mathbf{A}^x_{\text{aug}})^{\mathrm{T}}$ and $\mathbf{M}^x$ during initialization, followed by a series of multiplications by $\mathbf{A}^x_{\text{aug}}$, $(\mathbf{A}^x_{\text{aug}})^{\mathrm{T}}$, and $\mathbf{M}^x$ in each iteration. The updates corresponding to $\boldsymbol{x}_2$ can be computed in parallel in a similar manner; the performance of the entire algorithm will be limited by the the slower of the two.

- **z-updates:** The update (4-24a) for the shared variable can again be reformulated as the following proximal minimization problem [43]:

$$\boldsymbol{z}^{(l+1)} = \arg\min \left(\mathcal{I}_{\mathcal{B}}(\boldsymbol{z}^{(l+1)}) + \frac{\tilde{\rho}}{2} \left\| \boldsymbol{z}^{(l+1)} - \tilde{\boldsymbol{z}}^{(l+1)} \right\|^2 \right), \tag{4-27}$$

where, for this 3-split case, the introduced $\tilde{\rho}/2 = \lambda + \rho$ and $\tilde{\boldsymbol{z}}^{(l+1)}$ is given by the weighted sum:

$$\tilde{\boldsymbol{z}}^{(l+1)} := \frac{1}{2} \frac{\rho}{\lambda + \rho} \sum_{i=1}^{2} (\boldsymbol{x}_i^{(l+1)} + \boldsymbol{u}_i^{(l)}). \tag{4-28}$$

The solution is then found by a simple projection onto the box inequality constraints:

$$\boldsymbol{z}^{(l+1)} = \Pi_{\mathcal{B}} \left(\tilde{\boldsymbol{z}}^{(l+1)} \right), \tag{4-29}$$

which takes a negligible amount of $\mathcal{O}(n^2)$ flops.

- **scaled dual updates:** The vector additions and subtraction required for the scaled dual update require only $\mathcal{O}(n^2)$ operations each and can be computed in parallel.

The symmetric version of these iteration updates are summarized as the 3-split S-ADMM algorithm below.

Algorithm 4.2 (3-split S-ADMM) Symmetric 3-split ADMM solution to Problem 4.1

```
    Input:
        Problem description $\boldsymbol{t}^x$, $\boldsymbol{t}^y$, $\mathbf{A}^x$, $\mathbf{A}^y$, $\lambda$, $\mathcal{B}$; parameter $\rho$
        Initial estimates $\boldsymbol{u}_i^{(0)}$ and $\boldsymbol{x}_i^{(0)} = \boldsymbol{z}^{(0)}$
1: for $l = 0, 1, 2, \ldots$ do
2:     Solve each $\boldsymbol{x}_i^{(l+1)} := \arg\min_{\boldsymbol{x}_i} \left( \left\| \boldsymbol{t}^{x/y} - \mathbf{A}^{x/y}\boldsymbol{x}_i \right\|^2 + \frac{\rho}{2} \left\| \boldsymbol{x}_i - \boldsymbol{z}^{(l)} + \boldsymbol{u}_i^{(l)} \right\|^2 \right)$ using MLSQR
3:     $\boldsymbol{u}_i^{(l+1/2)} := \boldsymbol{u}_i^{(l)} + \boldsymbol{x}_i^{(l+1)} - \boldsymbol{z}^{(l)}$
4:     $\boldsymbol{z}^{(l+1)} := \Pi_{\mathcal{B}} \left( \frac{1}{2} \frac{\rho}{\lambda + \rho} \sum_{i=1}^{2} (\boldsymbol{x}_i^{(l+1)} + \boldsymbol{u}_i^{(l+1/2)}) \right)$
5:     $\boldsymbol{u}_i^{(l+1)} := \boldsymbol{u}_i^{(l+1/2)} + \boldsymbol{x}_i^{(l+1)} - \boldsymbol{z}^{(l+1)}$
6:     Check stopping criteria.
7: end for
```

Fixing the penalty parameter As in the 2-split scheme, the penalty parameter ρ should be kept constant throughout the ADMM iterations so that we can pre-compute effective preconditioners for the local variable updates. We thus need to choose its value in a way that leads to the fastest rate of convergence. To this end, we solved the 100 sample problems of the AO numerical dataset using the 3-split S-ADMM algorithm with various values for ρ. In each case, the average and standard deviation of the $L_{\text{2-ADMM}}$ number of iterations needed for convergence were recorded[11] along with the achieved relative objective and solution errors defined as (4-22) before.

The results of this study are summarized in Table 4-3. For the stopping criteria, we used the same values of the absolute and relative parameters $\epsilon^{\text{abs}} = \epsilon^{\text{rel}} = 10^{-2}$ as in the 2-split scheme, which again lead to solutions of acceptable accuracy. The average number of ADMM iterations are minimized with minimal variation at approximately $\rho = 0.02$. The evolution of the objective function in the optimization problems for the first two time steps of the AO numerical data with this choice are shown in a latter comparison on Figure 4-6.

Table 4-3: Effect of the penalty parameter ρ in the 3-split S-ADMM scheme on the average number of iterations needed to solve the 100 consecutive AO control problems of the sample numerical dataset. The achieved relative errors of the objective and solution are also shown with their standard deviations. The optimal choice of the penalty parameter is highlighted in bold.

ρ	$L_{\text{3-ADMM}}$	$\epsilon^{\text{rel}}_{\text{obj}}$ [$\cdot 10^3$ %]	$\epsilon^{\text{rel}}_{\text{sol}}$ [%]
0.008	5.5 ± 0.5	12.1 ± 7.8	2.61 ± 1.89
0.012	6.6 ± 0.5	12.2 ± 11.6	1.49 ± 1.07
0.020	$\mathbf{5.0 \pm 0.0}$	$\mathbf{32.8 \pm 21.2}$	$\mathbf{5.59 \pm 3.18}$
0.030	5.4 ± 0.5	90.4 ± 31.8	8.40 ± 3.53
0.050	6.9 ± 0.3	211.5 ± 64.1	11.81 ± 4.29

[11] The iterations for solving the first time step were again excluded from the statistics.

Choosing the preconditioners With the choice $\rho = 0.02$, the coefficient matrices $\mathbf{A}^x_{\text{aug}}$ and $\mathbf{A}^y_{\text{aug}}$ of the local variable updates (4-24a) become fixed. We can now examine their structure in order to choose an effective sparsity pattern for each of the MLSQR preconditioners $\mathbf{M}^x$ and $\mathbf{M}^y$ in the low rank Kronecker form (4-26). The inverse of the corresponding Hessians $\left(\mathbf{A}^x_{\text{aug}}\right)^{\mathrm{T}} \mathbf{A}^x_{\text{aug}}$ and $\left(\mathbf{A}^y_{\text{aug}}\right)^{\mathrm{T}} \mathbf{A}^y_{\text{aug}}$ and the chosen sparsity patterns for their preconditioner Kronecker terms are depicted in Figures 4-4 and 4-5, respectively. The patterns were obtained using the same averaging method described for the 2-split scheme; our comments about determining them using an *a priori* dense inverse approximation (or a possibly adaptive pattern selection algorithm) remain relevant in this case as well. Notice that for each Kronecker pair, either the left or right term is about half as sparse as its 2-split counterpart in Figure (4-3)[12].

The relation between the preconditioner Kronecker rank N and the average number of MLSQR iterations needed to solve the local updates for both the x and y slope residuals during each ADMM iteration of the entire AO dataset is shown in Table 4-4[13]. The solution accuracy was controlled by setting the stopping criteria parameter $\epsilon^{\text{abs}}_{\text{MLSQR}} = 10^{-3}$. The achievable performance becomes limited by the sparsity patterns imposed on the preconditioners; reasonable choices are $N^x = 2$ and $N^y = 3$ for the two cases.

Performance The performance of the 3-split S-ADMM mainly depends on the number of *consecutive* matrix-vector multiplications with the coefficient matrices $\mathbf{A}^{x/y}_{\text{aug}}$ (or its transpose) and the preconditioners $\mathbf{M}^x$ or $\mathbf{M}^y$ during the MLSQR iterations required to solve the local variable updates. With $L_{\text{3-ADMM}}$ and $L^{x/y}_{\text{MLSQR}}$ iterations of these respective algorithms, these numbers are $L_{\text{3-ADMM}}(1 + 2L^{x/y}_{\text{MLSQR}})$ and $L_{\text{3-ADMM}}(1 + L^{x/y}_{\text{MLSQR}})$, respectively. The overall performance will be limited by the update which takes longer to evaluate. In our case though, the preconditioners $\mathbf{M}^x$ and $\mathbf{M}^y$ are similarly sparse, while Table 4-4 shows we can expect $L^x_{\text{MLSQR}} \approx L^y_{\text{MLSQR}}$, so the two computational loads are well balanced. In the context of the numerical study conducted with the AO dataset, we have an average of around 35 multiplications with $\mathbf{A}^{x/y}_{\text{aug}}$ (or its transpose) and 20 with $\mathbf{M}^x$ or $\mathbf{M}^y$ during each time step.

Compared to the 2-split scheme, it is clear that further decomposing the objective function allows for faster solution of the local subproblems. The preconditioners $\mathbf{M}^x$ and $\mathbf{M}^y$ are sparser than $\mathbf{M}$ while they are just as good preconditioners and require an average of around 3 MLSQR iterations until convergence. Since in each Kronecker pair of $\mathbf{M}^x$ and $\mathbf{M}^y$, one of them contains half the elements as the corresponding term in $\mathbf{M}$, multiplications with them are expected to be evaluated 25 % faster. On the other hand, the 3-split S-ADMM iterations themselves converge at a slower rate, which increases the total computational burden. This trade-off will determine which scheme is better suited for real-time implementation.

Table 4-4: Average number L_{MLSQR} of MLSQR iterations for solving the local variable updates during each 3-split S-ADMM iteration as a function of the preconditioner's Kronecker rank N.

N	1	**2**	**3**	4	5
L^x_{MLSQR}	4.32 ± 0.45	$\mathbf{3.00 \pm 0.06}$	3.00 ± 0.00	3.00 ± 0.00	3.00 ± 0.00
L^y_{MLSQR}	5.00 ± 0.00	3.89 ± 0.27	$\mathbf{3.00 \pm 0.00}$	3.00 ± 0.00	3.00 ± 0.00

[12]It will be interesting to see whether this difference increases for larger-scale problems.

[13]As in the 2-split scheme, optimizing the selection of the preconditioners and ρ together could be beneficial.

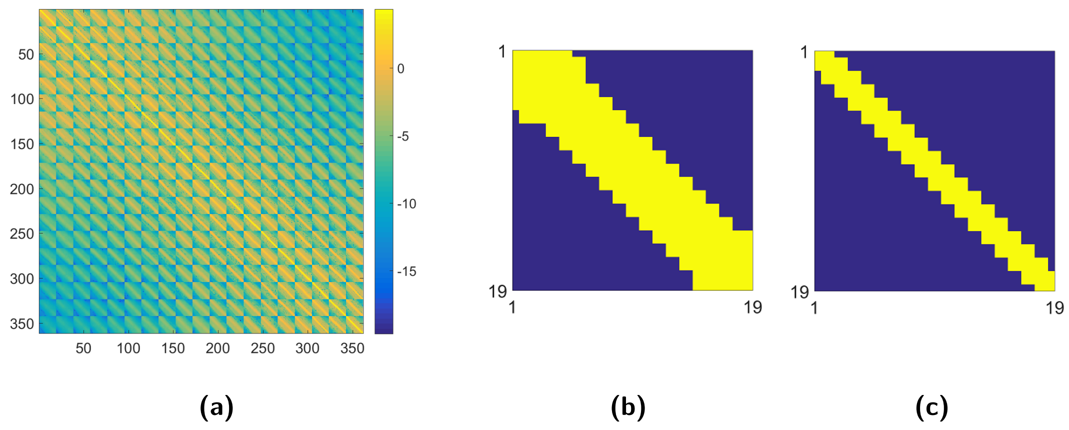


Figure 4-4: (a) Inverse of the matrix $\left(\mathbf{A}^x_{\text{aug}}\right)^{\mathrm{T}}\mathbf{A}^x_{\text{aug}}$ on a logarithmic scale; (b),(c) the corresponding sparsity patterns chosen for the preconditioner Kronecker terms $\mathbf{M}^x_{L,i}$ and $\mathbf{M}^x_{R,i}$. They have 127 and 55 non-zero elements (in yellow), respectively – fill-ins of $35.2\,\%$ and $15.2\,\%$.

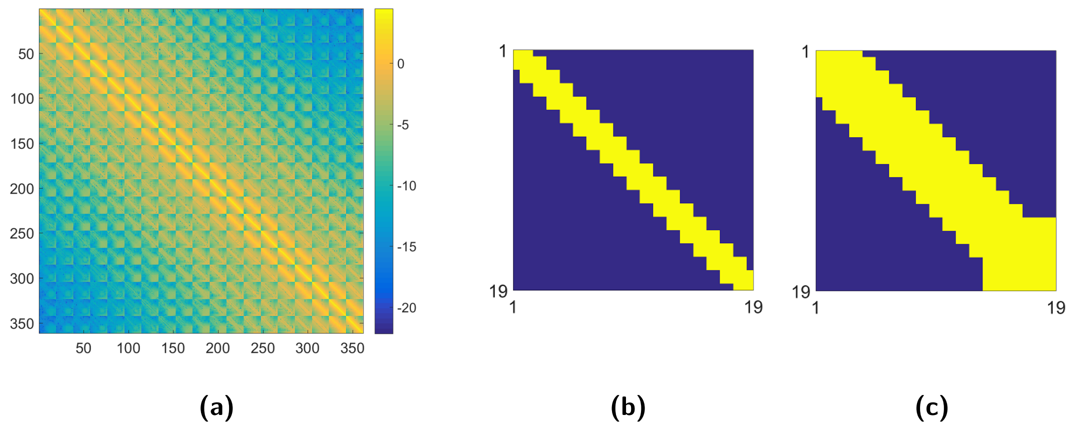


Figure 4-5: (a) Inverse of the matrix $\left(\mathbf{A}^y_{\text{aug}}\right)^{\mathrm{T}}\mathbf{A}^y_{\text{aug}}$ on a logarithmic scale; (b),(c) the corresponding sparsity patterns chosen for the preconditioner Kronecker terms $\mathbf{M}^y_{L,i}$ and $\mathbf{M}^y_{R,i}$. They have 55 and 127 non-zero elements (in yellow), respectively – fill-ins of $15.2\,\%$ and $35.2\,\%$.

4-4 PABB solution

Projected gradient methods are one of the most straightforward iterative methods used for solving constrained optimization problems. In a detailed study by Konnik [46], their applicability and efficiency has been demonstrated for wavefront control in adaptive optics systems. The simplicity of projected gradient (PG) methods stems from the fact that at any step l they rely on the gradient $\boldsymbol{g}^{(l)}$ of the objective function and a projection onto the feasible set to compute subsequent solution approximations. In the context of Problem 4.1, which can be alternatively written as:

$$\min_{\boldsymbol{x}\in\mathcal{B}} \; F(\boldsymbol{x}) = \left\| \begin{bmatrix} \boldsymbol{t}^x \\ \boldsymbol{t}^y \\ \mathbf{0} \end{bmatrix} - \begin{bmatrix} \sum_{j=1}^{M} \mathbf{B}_j^x \otimes \mathbf{C}_j^x \\ \sum_{j=1}^{M} \mathbf{B}_j^y \otimes \mathbf{C}_j^y \\ \sqrt{\lambda}\mathbf{I}_{n^2} \end{bmatrix} \boldsymbol{x} \right\|^2 := \left\| \boldsymbol{t}_{\text{aug}} - \mathbf{A}_{\text{aug}}\boldsymbol{x} \right\|^2 , \tag{4-30}$$

the gradient of the objective is

$$\boldsymbol{g}^{(l)} = \mathbf{A}_{\text{aug}}^{\mathrm{T}} \left(\mathbf{A}_{\text{aug}}\boldsymbol{x}^{(l)} - \boldsymbol{t}_{\text{aug}} \right) . \tag{4-31}$$

The next iterate is then determined by taking a given α_l step in this direction and projecting the result onto the set of constraints:

$$\boldsymbol{x}^{(l+1)} = \Pi_{\mathcal{B}} \left(\boldsymbol{x}^{(l)} - \alpha_l \boldsymbol{g}^{(l)} \right) . \tag{4-32}$$

These updates are most useful when the projection operator $\Pi_{\mathcal{B}}(\cdot)$ is cheap to evaluate, as is the case for box inequality constraints. Most of the computational expense then falls onto calculating the gradient, for which the sparse and Kronecker structure of $\mathbf{A}_{\text{aug}}$ can be exploited. Projected Gradient methods generally differ in the way the step size α_l is chosen. In our second approach to solving Problem 4.1, here we examine one if its variants called the projected alternating Barzilai-Borwein method for box-constrained quadratic programming [52], which has also been advocated for in the PhD dissertation of Konnik.

4-4-1 Algorithm

The selection of step lengths for the PABB method stem from a study conducted for improving the unconstrained gradient method by Barzilei and Borwein [53]. Therein, the authors propose a two-point scheme based on the differences

$$\Delta\boldsymbol{x}^{(l-1)} := \boldsymbol{x}^{(l)} - \boldsymbol{x}^{(l-1)} \qquad \text{and} \qquad \Delta\boldsymbol{g}^{(l-1)} = \boldsymbol{g}^{(l)} - \boldsymbol{g}^{(l-1)} \tag{4-33}$$

at each iteration l. Two choices for determining the step length are then proposed as:

$$\alpha_l^{\text{BB1}} = \frac{\left\langle \Delta\boldsymbol{x}^{(l-1)},\, \Delta\boldsymbol{x}^{(l-1)} \right\rangle}{\left\langle \Delta\boldsymbol{x}^{(l-1)},\, \Delta\boldsymbol{g}^{(l-1)} \right\rangle} \qquad \text{and} \qquad \alpha_l^{\text{BB2}} = \frac{\left\langle \Delta\boldsymbol{x}^{(l-1)},\, \Delta\boldsymbol{g}^{(l-1)} \right\rangle}{\left\langle \Delta\boldsymbol{g}^{(l-1)},\, \Delta\boldsymbol{g}^{(l-1)} \right\rangle}.$$

The (non-monotonic) convergence of the method for strictly convex quadratic problems using either of these options has been proven by Raydan [54].

The first proposal α_l^{BB1} is generally believed to perform better than the second one. However, in the projected gradient framework, this does not necessarily seem to be the case. In a study by Dai and Fletcher [52] for large-scale box-constrained quadratic problems, it was recommended to use both possibilities as the step length in an alternating fashion, leading to the following PABB scheme:

$$\alpha_l^{\mathrm{PABB}} := \begin{cases} \alpha_l^{\mathrm{BB1}} & \text{for odd } l \\ \alpha_l^{\mathrm{BB2}} & \text{for even } l. \end{cases} \tag{4-34}$$

As opposed to the unconstrained case, convergence is not guaranteed as the iterates may cycle between several points indefinitely. However, as we will also see for the AO problem, PABB often works well in practice. In the study by Dai and Fletcher, it has been shown to be superior to the traditional steepest descent step length selection as well as to achieve performance comparable to the combination of gradient projection methods with conjugate gradient methods [55].

The complete algorithm is summarized as Algorithm 4.3 below. Warm-start is achieved by initializing the iterations with the solution and gradient obtained as the result for the previous time step. Stopping criteria can be based on an absolute and relative tolerance on the change in the unknown, i.e.:

$$\left\|\boldsymbol{x}^{(l+1)} - \boldsymbol{x}^{(l)}\right\| \leq n\epsilon^{\mathrm{abs}} + \epsilon^{\mathrm{rel}} \left\|\boldsymbol{x}^{(l+1)}\right\| \tag{4-35}$$

similarly to the criteria (B-8) used in the ADMM approach.

Algorithm 4.3 (PABB) projected alternating Barzilai-Borwein solution to Problem 4.1

```
   Input:
      Problem description $\mathbf{A}_{\mathrm{aug}}$, $\boldsymbol{t}_{\mathrm{aug}}$, $\mathcal{B}$
      Initial values $\boldsymbol{x}^{(0)}$, $\boldsymbol{g}^{(0)}$, $\alpha_0$
   Initialize:
      $\boldsymbol{x}^{(1)} := \Pi_{\mathcal{B}}\left(\boldsymbol{x}^{(0)} - \alpha_0 \boldsymbol{g}^{(0)}\right)$
1: for $l = 1, 2, 3, \ldots$ do
2:    $\boldsymbol{g}^{(l)} := \mathbf{A}_{\mathrm{aug}}^{\mathrm{T}}\left(\mathbf{A}_{\mathrm{aug}}\boldsymbol{x}^{(l)} - \boldsymbol{t}_{\mathrm{aug}}\right)$
3:    $\Delta\boldsymbol{x}^{(l-1)} := \boldsymbol{x}^{(l)} - \boldsymbol{x}^{(l-1)}$
4:    $\Delta\boldsymbol{g}^{(l-1)} := \boldsymbol{g}^{(l)} - \boldsymbol{g}^{(l-1)}$
5:    $\alpha_l^{\mathrm{PABB}} := \begin{cases} \dfrac{\left\langle \Delta\boldsymbol{x}^{(l-1)},\ \Delta\boldsymbol{x}^{(l-1)} \right\rangle}{\left\langle \Delta\boldsymbol{x}^{(l-1)},\ \Delta\boldsymbol{g}^{(l-1)} \right\rangle} & \text{for odd } l \\ \dfrac{\left\langle \Delta\boldsymbol{x}^{(l-1)},\ \Delta\boldsymbol{g}^{(l-1)} \right\rangle}{\left\langle \Delta\boldsymbol{g}^{(l-1)},\ \Delta\boldsymbol{g}^{(l-1)} \right\rangle} & \text{for even } l. \end{cases}$
6:    $\boldsymbol{x}^{(l+1)} := \Pi_{\mathcal{B}}\left(\boldsymbol{x}^{(l)} - \alpha_l^{\mathrm{PABB}} \boldsymbol{g}^{(l)}\right)$
7:    Check stopping criteria
8: end for
```

4-4-2 Performance

To evaluate the performance of the PABB solution approach, we solved the 100 consecutive AO control problems which form the basis of our numerical study. The average number of L_{PABB} iterations, along with the achieved relative objective and solution errors as defined by (4-22), were found to be:

$$L_{\text{PABB}} = 5.28 \pm 0.88, \qquad \epsilon^{\text{rel}}_{\text{obj}} = 0.040 \pm 0.025\,\%, \qquad \text{and} \qquad \epsilon^{\text{rel}}_{\text{sol}} = 6.00 \pm 3.11\,\%.$$

Stopping criteria for the algorithm were controlled by the parameters $\epsilon^{\text{abs}} = \epsilon^{\text{rel}} = 10^{-2}$, which achieved errors similar to the 2- and 3-split S-ADMM algorithms. Figure 4-6 shows the convergence of the three solution schemes for the first two time steps of the AO numerical data. Note that warm-start has a more substantial effect on the ADMM based methods, where the number of iterations are essentially halved for the second time step.

One of the main advantages of PABB is that it is very inexpensive. Each iteration of Algorithm 4.3 requires two consecutive matrix-vector multiplications by $\mathbf{A}_{\text{aug}}$ and its transpose, respectively, where the Kronecker and sparse structures can be readily exploiting. The additional $\mathcal{O}(n^2)$ operations needed to compute differences and norms of vectors and the projection (4-13) onto the bound constraints are negligible in comparison. For solving a single time step, a good approximation of the computational effort is thus $2L_{\text{PABB}}$ multiplications by $\mathbf{A}_{\text{aug}}$ or $\mathbf{A}^{\text{T}}_{\text{aug}}$, which amounts to a total number of around 10.6 for our case. A further advantage of PABB is that it requires no pre-calculations nor effective preconditioning, although its performance is known to degrade with decreasing number of saturated actuators and increasing ill-conditioning of the coefficient matrix $\mathbf{A}_{\text{aug}}$ [46].

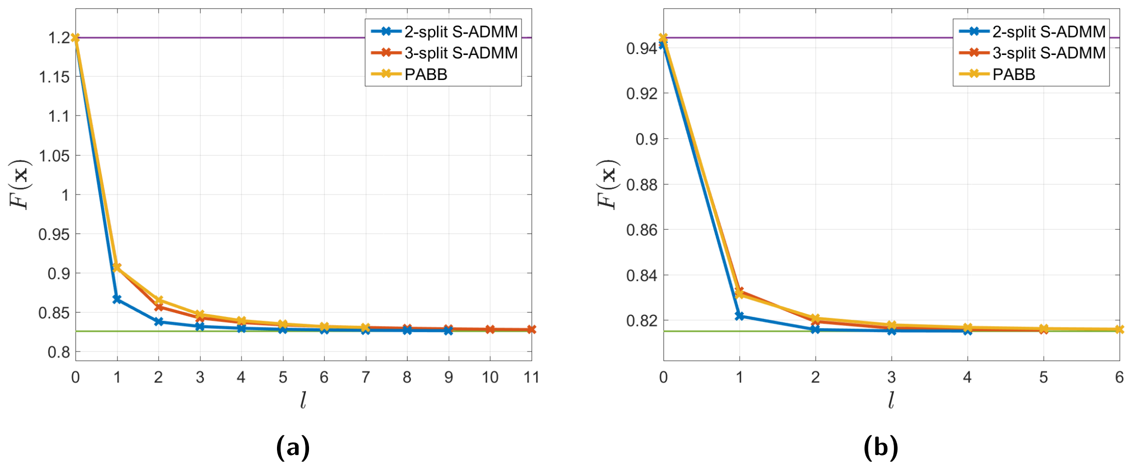


Figure 4-6: Convergence of the 2-split S-ADMM, 3-split S-ADMM, and PABB solution algorithms when solving the AO control problem of the (a) first and (b) second time steps of our numerical dataset. The horizontal lines show the initial and optimal objective values. With all three methods, the latter is reached well within an acceptable tolerance for real-time applications. The effectiveness of warm-start is evident from the second time step, especially for the ADMM schemes.

4-5 Active sets solution

Active sets methods have been shown to be extremely effective for solving a series of similar quadratic optimization problems due to their efficient warm-start capabilities. In fact, the thesis of Konnik [46] argues that one of its simplest variants is the best option for the real-time solution of the bound-constrained wavefront control problem formulated therein. Thus this method will serve as our third approach for solving Problem 4.1 and in the following we examine whether it can be adapted to exploit the structures within our Kronecker formulation. For a detailed overview of the AS method used throughout this section, we refer to the books by Björck [56] or by Nocedal and Wright [57] as an extension to the following brief summary of the main concepts outlined in the Literature Survey [43].

Within the nomenclature of AS methods, the *active set* denotes all the inequality (and equality) constraints satisfied by a current approximation $\boldsymbol{x}^{(l)}$ of the unknown solution. The goal of the algorithm is to find the optimal active set, i.e. the one corresponding to the true solution $\boldsymbol{x}^*$. If this were known in advance, the inactive inequalities could be ignored and $\boldsymbol{x}^*$ could be found by solving the original problem constrained only by the equalities within the optimal active set. However, as this set is unknown, we instead search for a candidate solution within its estimate, the *working set*, at each iteration of the AS algorithm. The working set is then updated based on the results in the following (simplified) manner. If moving towards the new candidate involves crossing an inequality bound, we include it in the working set. Otherwise, the Karush-Kuhn-Tucker (KKT) conditions for optimality are evaluated for the the new candidate solution; if these are not satisfied, then an appropriate constraint is dropped from the working set based on the values of the dual variables corresponding to the active constraints. The iterations continue with the updated working set and terminate when the KKT conditions reveal that the true solution has been found.

Solving the series of equality constrained optimization problems defined by the iteratively updated working sets comprises the main computational expense of the AS algorithm. Consequently, the overall performance depends on the number of required updates, which can drop drastically as the control loop frequency increases in real-time applications.

In the coming sections, we focus on solving these types of equality constrained problems while exploiting the Kronecker structures of our matrices. For our main Problem 4.1, the lth working set $\mathcal{W}^{(l)} \subseteq \mathcal{B}$ can be expressed as the linear constraint:

$$\mathcal{W}^{(l)} = \{\boldsymbol{x} \mid \mathbf{E}^{(l)}\boldsymbol{x} = \boldsymbol{b}^{(l)}\}, \tag{4-36}$$

where $\mathbf{E}^{(l)} \in \mathbb{R}^{p\times n}$ selects the components of $\boldsymbol{x}$ which must actively satisfy a p number of upper or lower bound constraints assembled in $\boldsymbol{b} \in \mathbb{R}^p$. To simplify our discussion, we will use the augmented form (4-30) introduced in the previous section to denote the objective:

$$F(\boldsymbol{x}) = \left\| \begin{bmatrix} \boldsymbol{t}^x \\ \boldsymbol{t}^y \\ \mathbf{0} \end{bmatrix} - \begin{bmatrix} \sum_{j=1}^{M} \mathbf{B}_j^x \otimes \mathbf{C}_j^x \\ \sum_{j=1}^{M} \mathbf{B}_j^y \otimes \mathbf{C}_j^y \\ \sqrt{\lambda}\mathbf{I}_{n^2} \end{bmatrix} \boldsymbol{x} \right\|^2 := \left\|\boldsymbol{t}_{\text{aug}} - \mathbf{A}_{\text{aug}}\boldsymbol{x}\right\|^2. \tag{4-37}$$

4-5-1 Optimizing within the working set

Using the notation introduced in the previous section, the at iteration l we will search for a new candidate solution in the direction $\Delta\boldsymbol{x}^{(l+1)} \in \mathbb{R}^{n^2}$ which solves:

$$\begin{aligned} &\min_{\Delta\boldsymbol{x}} \left\| \boldsymbol{t}_{\text{aug}} - \mathbf{A}_{\text{aug}}(\boldsymbol{x}^{(l)} + \Delta\boldsymbol{x}) \right\|^2 \\ &\text{s.t. } \mathbf{E}^{(l)}(\boldsymbol{x}^{(l)} + \Delta\boldsymbol{x}) = \boldsymbol{b}. \end{aligned} \tag{4-38}$$

The corresponding KKT conditions can be written in the form:

$$\begin{bmatrix} \mathbf{A}_{\text{aug}}^{\text{T}}\mathbf{A}_{\text{aug}} & \left(\mathbf{E}^{(l)}\right)^{\text{T}} \\ \mathbf{E}^{(l)} & \mathbf{0} \end{bmatrix} \cdot \begin{bmatrix} \Delta\boldsymbol{x} \\ \boldsymbol{\nu} \end{bmatrix} = \begin{bmatrix} \mathbf{A}_{\text{aug}}^{\text{T}}\boldsymbol{r}^{(l)} \\ \mathbf{0} \end{bmatrix}, \tag{4-39}$$

where the residual $\boldsymbol{r}^{(l)} = \boldsymbol{t}_{\text{aug}} - \mathbf{A}_{\text{aug}}\boldsymbol{x}^{(l)} \in \mathbb{R}^{n^2}$ and the dual variable $\boldsymbol{\nu} \in \mathbb{R}^p$. Solutions to symmetric indefinite linear equations of this form, so-called *saddle point problems*, have been studied extensively; a comprehensive overview is given by Benzi et al. [58].

One popular approach relies on the LDL^{T} decomposition of the coefficient matrix. The key observation here is that the $(1,1)$ block $\mathbf{A}_{\text{aug}}^{\text{T}}\mathbf{A}_{\text{aug}}$ is independent of the working set, which can be exploited to efficiently update the lower triangular $\mathbf{L}$ and diagonal $\mathbf{D}$ components instead of having to recompute them every iteration, as demonstrated e.g. in Wong [59] or Maes [60]. The gain is substantial, since the factorization allows solving (4-39) using backsolves with the triangular matrices, an operation essentially as inexpensive as matrix-vector multiplications themselves. The updates in case of including new constraints for a simple Kronecker structured problem is demonstrated by the following example.

Example. ***(Updating the* LDL^{T} *factorization of saddle point matrices)***
In this example, we assume $\mathbf{A}_{\text{aug}}^{\text{T}}\mathbf{A}_{\text{aug}}$ *is composed of the single Kronecker product*

$$\mathbf{A}_{\text{aug}}^{\text{T}}\mathbf{A}_{\text{aug}} = (\mathbf{B}^{\text{T}}\mathbf{B}) \otimes (\mathbf{C}^{\text{T}}\mathbf{C}) \tag{4-40}$$

with $\mathbf{B}, \mathbf{C} \in \mathbb{R}^{m\times n}$ *and where the two Kronecker terms admit the decompositions*

$$\mathbf{B}^{\text{T}}\mathbf{B} = \mathbf{L}_B\mathbf{D}_B\mathbf{L}_B^{\text{T}} \qquad \textit{and} \qquad \mathbf{C}^{\text{T}}\mathbf{C} = \mathbf{L}_C\mathbf{D}_C\mathbf{L}_C^{\text{T}}. \tag{4-41}$$

The LDL^{T} *decompositions here are such that the* $\mathbf{L}$ *terms are unit lower triangular*[14] *and the* $\mathbf{D}$ *terms are diagonal matrices of size* n *by* n*; in this manner, the decomposition becomes unique for symmetric, full rank matrices [28]. Dropping the index* (l) *from* $\mathbf{E}^{(l)} := \mathbf{E} \in \mathbb{R}^{p\times n^2}$ *for simplicity, the corresponding decomposition of the coefficient matrix in Equation (4-39) can be written as:*

$$\begin{bmatrix} \mathbf{A}_{\text{aug}}^{\text{T}}\mathbf{A}_{\text{aug}} & \mathbf{E}^{\text{T}} \\ \mathbf{E} & \mathbf{0} \end{bmatrix} = \begin{bmatrix} \mathbf{L}_B \otimes \mathbf{L}_C & \mathbf{0} \\ \mathbf{F}_E & \mathbf{L}_E \end{bmatrix} \cdot \begin{bmatrix} \mathbf{D}_B \otimes \mathbf{D}_C & \mathbf{0} \\ \mathbf{0} & \mathbf{D}_E \end{bmatrix} \cdot \begin{bmatrix} \mathbf{L}_B^{\text{T}} \otimes \mathbf{L}_C^{\text{T}} & \mathbf{F}_E^{\text{T}} \\ \mathbf{0} & \mathbf{L}_E^{\text{T}} \end{bmatrix},$$

where $\mathbf{L}_E, \mathbf{D}_E \in \mathbb{R}^{p\times p}$ *have unit lower triangular and diagonal structure, while* $\mathbf{F}_E \in \mathbb{R}^{p\times n^2}$ *is in general a full matrix.*

[14]The term *unit triangular* refers to triangular matrices with ones on the main diagonal.

We now consider the case when the $(p+1)$th constraint is included in the working set, expanding the matrix $\mathbf{E}$ by a new row $\boldsymbol{z}^{\mathrm{T}} \in \mathbb{R}^{1\times n^2}$. The $\mathrm{LDL}^{\mathrm{T}}$ *decomposition of the new coefficient matrix will then admit the following representation:*

$$\begin{bmatrix} \mathbf{A}_{\mathrm{aug}}^{\mathrm{T}}\mathbf{A}_{\mathrm{aug}} & \mathbf{E}^{\mathrm{T}} & \boldsymbol{z} \\ \mathbf{E} & \mathbf{0} & \mathbf{0} \\ \boldsymbol{z}^{\mathrm{T}} & \mathbf{0} & 0 \end{bmatrix} = \begin{bmatrix} \mathbf{L}_B \otimes \mathbf{L}_C & \mathbf{0} & \mathbf{0} \\ \mathbf{F}_E & \mathbf{L}_E & \mathbf{0} \\ \boldsymbol{f}_z^{\mathrm{T}} & \boldsymbol{l}_z^{\mathrm{T}} & 1 \end{bmatrix} \cdot \begin{bmatrix} \mathbf{D}_B \otimes \mathbf{D}_C & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{D}_E & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & d_z \end{bmatrix} \cdot \begin{bmatrix} \mathbf{L}_B^{\mathrm{T}} \otimes \mathbf{L}_C^{\mathrm{T}} & \mathbf{F}_E^{\mathrm{T}} & \boldsymbol{f}_z \\ \mathbf{0} & \mathbf{E}_L & \boldsymbol{l}_z \\ \mathbf{0} & \mathbf{0} & 1 \end{bmatrix},$$

where the $\mathbf{L}$ *and* $\mathbf{D}$ *factors become augmented by the unknown row vectors* $\boldsymbol{f}_z^{\mathrm{T}} \in \mathbb{R}^{1\times n^2}$, $\boldsymbol{l}_z^{\mathrm{T}} \in \mathbb{R}^{1\times p}$, *and the* d_z *is scalar. Note that the Kronecker structure is preserved in the upper left blocks of these terms. For the updated decomposition to hold, the first entry of the last column (or row) of this matrix equation reveals we must have the relation:*

$$\boldsymbol{z} = \left(\mathbf{L}_B^{\mathrm{T}} \otimes \mathbf{L}_C^{\mathrm{T}}\right)(\mathbf{D}_B \otimes \mathbf{D}_C)\,\boldsymbol{f}_z.$$

This can be reformulated using the property (A-12) of Kronecker products and the notation $\mathrm{vec}(\mathbf{Z}) = \boldsymbol{z}$ *and* $\mathrm{vec}(\mathbf{F}_z) = \boldsymbol{f}_z$ *in the following manner:*

$$\mathbf{Z} = \mathbf{L}_C^{\mathrm{T}}\left(\mathbf{D}_C \mathbf{F}_z \mathbf{D}_B\right)\mathbf{L}_B,$$

which can be solved in $\mathcal{O}(n^3)$ *operations using* n *backsolves with each triangular matrices (and dividing by the diagonal entries) for the unknown* $\mathbf{F}_z$. *The rest of the steps do not benefit anymore from the Kronecker structure and the gain of its previous exploitation remain substantial only if the number of constraints* $p \ll n$. *The equation corresponding to the second row of the last column reads: We must then have:*

$$\mathbf{0} = \mathbf{F}_E\,(\mathbf{D}_B \otimes \mathbf{D}_C)\,\boldsymbol{f}_z + \mathbf{L}_E \mathbf{D}_E \boldsymbol{l}_z,$$

where only $\boldsymbol{l}_z$ *is unknown and the backsolves needed to compute it have to be applied using the full lower matrix* $\mathbf{L}_E$. *The final unknown* d_z *can now be determined by the scalar equation corresponding to the lower right block:*

$$d_z = -(\boldsymbol{f}_z^{\mathrm{T}}\boldsymbol{f}_z + \boldsymbol{l}_z^{\mathrm{T}}\boldsymbol{l}_z).$$

As we can see, the decomposition remains unique and holds for any added $\boldsymbol{z}^{\mathrm{T}}$ *constraint.*

Unfortunately, such an efficient update scheme does not seem possible if the term $\mathbf{A}_{\mathrm{aug}}^{\mathrm{T}}\mathbf{A}_{\mathrm{aug}}$ is not composed of a single Kronecker product. It can be easily verified that for such a case the $\mathrm{LDL}^{\mathrm{T}}$ decomposition does not retain a favorable structure, even for low Kronecker ranks. Instead, we resort to using the iterative method SQMR to solve Equation (4-39) following the recommendations . SQMR has been discussed for its use in solving symmetric indefinite linear systems in Section 3-2-1. Based on the arguments presented in Lukšan and Vlček [61], we will use a block preconditioner $\mathbf{P}^{(l)}$ to approximate the inverse of the coefficient matrix in each iteration l of the AS method. With the Schur-complement of the upper left block, this can be expressed as:

$$\mathbf{P}^{(l)} := \begin{bmatrix} \mathbf{M}^{-1} & (\mathbf{E}^{(l)})^{\mathrm{T}} \\ \mathbf{E}^{(l)} & \mathbf{0} \end{bmatrix}^{-1} = \begin{bmatrix} \mathbf{I} & -\mathbf{M}(\mathbf{E}^{(l)})^{\mathrm{T}} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \cdot \begin{bmatrix} \mathbf{M} & \mathbf{0} \\ \mathbf{0} & \left(\mathbf{E}^{(l)}\mathbf{M}(\mathbf{E}^{(l)})^{\mathrm{T}}\right)^{-1} \end{bmatrix} \cdot \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ -\mathbf{E}^{(l)}\mathbf{M} & \mathbf{I} \end{bmatrix}, \quad (4\text{-}42)$$

where $\mathbf{M}$ is an approximation of the term $(\mathbf{A}_{\mathrm{aug}}^{\mathrm{T}}\mathbf{A}_{\mathrm{aug}})^{-1}$ that should be inexpensive to apply in matrix-vector multiplications.

4-5-2 Preconditioner selection

Examining the form (4-42) of the proposed preconditioner, it is clear that the inverse approximation $\mathbf{M}$ remains constant throughout the working set updates. Hence, it can be pre-calculated in a low Kronecker rank form in order to preserve the efficiency of the Kronecker structure using the Kron-ALS algorithm with symmetry enforced for its use in SQMR. To this end, we follow the same procedure to determine its sparsity pattern and optimal Kronecker rank as before in the context of solving the ADMM update equations.

For finding our preconditioner using Kron-ALS, we can express $\left(\mathbf{A}_{\text{aug}}^{\mathrm{T}}\mathbf{A}_{\text{aug}}\right)^{-1}$ as[15]:

$$\sum_{j_1=1}^{M}\sum_{j_2=1}^{M}\left[\left(\left(\mathbf{B}_{j_1}^{x}\right)^{\mathrm{T}}\mathbf{B}_{j_2}^{x}\right)\otimes\left(\mathbf{C}_{j_1}^{x\ \mathrm{T}}\mathbf{C}_{j_2}^{x}\right)+\left(\mathbf{B}_{j_1}^{y\ \mathrm{T}}\mathbf{B}_{j_2}^{y}\right)\otimes\left(\left(\mathbf{C}_{j_1}^{y}\right)^{\mathrm{T}}\mathbf{C}_{j_2}^{y}\right)\right]+(\sqrt{\lambda}\mathbf{I}_n)\otimes(\sqrt{\lambda}\mathbf{I}_n).$$

The inverse of this matrix for our case without regularization ($\lambda = 0$) is depicted below in Figure 4-7. Unfortunately, applying the same thresholding scheme as for the ADMM cases results in sparsity patterns for the preconditioner Kronecker terms which are fully dense, even with 10 % of the maximum values kept. Compared to the approximation of (4-17) for the 2-split S-ADMM scheme[16], it is thus clear that added regularization plays an important role for the sparse structure to appear. Here, as we are forced to approximate a dense preconditioner, the efficiency of working with sparse matrices throughout the solution is lost.

To determine an optimal choice of the preconditioner Kronecker rank, we solved the 100 AO control problems using SQMR with the working sets assumed to be known for simplicity. The average number of required iterations L_{SQMR} for convergence and achieved relative solution errors are summarized in Table 4-5. Since the accuracy would be important for updating the working set, we used a higher relative tolerance of $\epsilon^{\text{rel}} = 10^{-4}$ in SQMR. The Kronecker rank 5 seems to be a good choice as the decrease in L_{SQMR} becomes marginal afterwards.

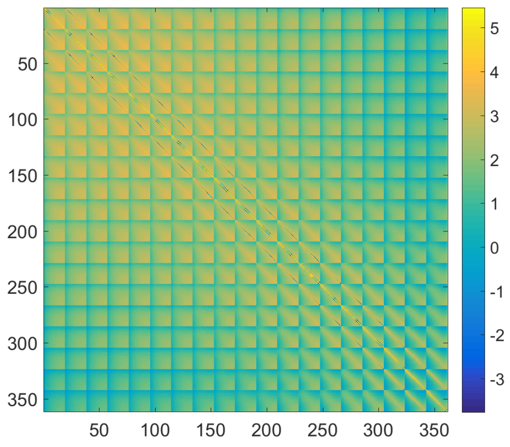


Figure 4-7: Inverse of the matrix $\mathbf{A}_{\text{aug}}^{\mathrm{T}}\mathbf{A}_{\text{aug}}$ on a logarithmic scale. It does not have a sparse structure and hence cannot be well approximated with any sparse pattern.

[15]For an interior point solution, this expression becomes augmented with a non-Kronecker diagonal term in the coefficient matrix of (4-39). It changes every iteration, which makes preconditioning difficult.

[16]To achieve similar performance as in this case, we could set $\lambda = 0.01$. This results in an around 10% increase in the achieved slope residual norms, which might be unacceptably large.

Table 4-5: Average number L_{SQMR} of SQMR iterations required for solving the KKT conditions in the AS approach to an accuracy of $\epsilon^{\text{rel}} = 10^{-4}$ for the problems within the AO numerical dataset as a function of the preconditioner Kronecker rank N.

N	1	3	4	**5**	7
L_{SQMR}	24.02 ± 1.76	9.76 ± 0.67	5.68 ± 0.49	$\mathbf{5.03 \pm 0.17}$	4.98 ± 0.14
$\epsilon_{\text{sol}}^{\text{rel}}$ $[\cdot 10^3\ \%]$	9.32 ± 8.31	1.60 ± 2.60	1.20 ± 2.24	$\mathbf{1.23 \pm 2.23}$	0.96 ± 2.31

4-5-3 Performance

For this performance analysis, we assume the number of active constraints are negligible compared to the problem size, i.e. $p \ll n$. Thus multiplications by $\mathbf{E}^{(l)}$ or the by the inverse of the Schur complement $\mathbf{E}^{(l)}\mathbf{M}\left(\mathbf{E}^{(l)}\right)^{\text{T}}$ will not be considered.

Remark. *The selection matrix $\mathbf{E}^{(l)}$ is an extremely sparse matrix allowing fast matrix-vector multiplications. As for the Schur-complement, we can perform its inverse operation using its* LDL^{T} *decomposition, which can be updated efficiently (in $\mathcal{O}(p^2)$) if p is not extremely large.*

Each iteration and the initialization of the SQMR algorithm requires one multiplication by the KKT coefficient matrix and the preconditioner $\mathbf{P}^{(l)}$. From equations (4-39) and (4-42), we can see that this constitutes a total of $2L_{\text{SQMR}} + 1$ and $3L_{\text{SQMR}}$ multiplications by $\mathbf{A}_{\text{aug}}$ (or its transpose) and the approximation $\mathbf{M}$, respectively. Based on the analysis presented by Konnik and De Doná [50], we can expect that the KKT equations will have to be solved only about once or twice each time step due to the low number of saturation changes during real-time control. Assuming this will tend towards 2 for larger actuator grids, these numbers of multiplications would be 22 and 32 for our numerical dataset. However, the Kronecker terms which compose $\mathbf{M}$ are fully dense, which leads to severe degradation in performance.

4-6 Results

In the course of this chapter, we examined three possible solutions to the AO wavefront control problem for ground-based telescope in a minimum variance control formulation. We now summarize the strengths and weaknesses of each approach in the context of exploiting the Kronecker structures of this problem, along with take-aways of their potential for real-time implementation. We also remark on differences expected compared to using a sparse solver. Note that due to the unrealistic nature of our generated wavefront control problems, the observations here should not be interpreted as a decisive feasibility study, though they do give some insight into the potential of each solution.

For evaluating the performance of the proposed solutions in the context of our numerical study, we assume a fully parallel implementation such that the cost of matrix-vector multiplications by any $\mathbf{A}_{\text{aug}}$ (or $\mathbf{A}_{\text{aug}}^{x/y}$) is essentially the same as with a single pair $\mathbf{B}_j \otimes \mathbf{C}_j$ of the summands in (4-12), as their effect can be calculated separably. Merging the obtained results, and other $\mathcal{O}(n^2)$ computations, will not be taken into account. A similar argument can be made for the cost of applying preconditioners; we will ignore their Kronecker ranks. Thus only the number of consecutive multiplications with these matrices (and their sparsity) are taken into account, as this is a good indication of the expected computational efforts.

Based on the performance evaluations for each individual approach, it is clear that PABB is the most promising for real-time implementation. The simplicity and effectiveness of the algorithm allows the control problems to be solved with the minimal number of multiplications involving the augmented coefficient matrix $\mathbf{A}_{\text{aug}}$. The ADMM methods are expected to perform about 4 times slower, while the AS approach is even worse especially due to the dense structure of its SQMR preconditioner. An additional advantage of PABB is that it does not require any pre-calculations. Furthermore, as a first order method, its effectiveness only depends on the conditioning of the augmented coefficient matrix, not on the decays in the Kronecker singular values of the Hessian. These latter play a vital role in the quality of attainable low separation rank preconditioners for the other two cases and may severely impact their usability. The main drawback of PABB compared to ADMM and AS is that it is less suited to accommodate more complex constraints, since its efficiency relies on a single inexpensive projection operation onto the feasible set.

Compared to a fully sparse solution, the performance of our Kronecker structured PABB will depend on the Kronecker ranks within the system matrices as well as their sparsity. Based on the study of the computational aspects of multiplications with structured Kronecker matrices in Appendix A-2, Kronecker products only provide a marginal gain if they are also sparse. With increasing separation ranks, the gap narrows, while with increasing nonzero entries, the gap widens. It remains to be seen whether the balance tips in favor of a sparse or a Kronecker structured formulation for a realistic AO system.

It is interesting to see that the other solutions perform much worse than PABB, while the differences in the sparse solutions examined by Konnik and De Doná [50] are not nearly as substantial. In fact, in their work, AS is concluded to be the most efficient option by a slight advantage over PABB. This highlights the main disadvantage of Kronecker structures: the iterative solutions which can possibly exploit the low Kronecker-rank structure are simply not as efficient as a factorization-based solution, even with good quality preconditioners. Pre-calculated factorizations preserve the banded structures of the problem, and allow backsolves to be performed with a complexity comparable to simple matrix-vector multiplications[17]. Furthermore, this operation needs to be evaluated once to obtain a solution, as opposed to the case of iterative solvers. Indeed, if the MLSQR and SQMR iterations had the ability to terminate in a single iteration, the ADMM and AS approaches would be much more competitive with PABB, similarly to the case of the sparse solvers.

We note that in preparation for laboratory experiments, the most promising PABB method has been implemented for a graphics processing unit (GPU) device using CUDA [62] in context of this thesis work. Although the algorithm is still in its infancy and does not yet exploit the bandedness of the Kronecker system matrices nor has it been fully optimized for the underlying hardward architecture, it provides a good foundation for evaluating the performance of the discussed minimum variance wavefront control in a closed-loop laboratory setting in the future. The details of this work, however, are outside the scope of this thesis report.

[17]Note, however, that the $\mathcal{O}(n^2)$ forward or backsolves with an n by n triangular matrix require consecutive operations, whereas matrix-vector products are much more parallelizable.

4-7 Future work

Our preliminary study of the computational aspects of possible Kronecker structure exploiting solutions to the AO wavefront control problem leaves many practical aspects open for further investigation.

One of the most critical features missing in our discussion is how the virtual actuators and lenslets can be handled. These appear for circular deformable mirrors and wavefront sensors which must be embedded into larger rectangles in order to admit a Kronecker structured system description. For the actuators, we assume this will not be a real issue; similarly to handling dead actuators, the corresponding upper and lower bounds can be set to 0 during optimization. The virtual lenslets, however, will appear in the slope measurement vector. Their assigned values will most likely influence the calculated actuator commands, although it is our hope that this effect will only degrade performance near the edge of the scientific image.

The scalability of the methods for even larger-scale systems also remains to be analyzed. The 19 by 19 actuator grid of our numerical study does not represent the dimensions appearing in next-generation telescopes, which might be an order of magnitude higher. It is thus important to consider how the performances of the proposed algorithms change with the problem size. The procedures for tuning the solutions and evaluating their performance in this thesis work should provide a usable reference for such a study, which should also be carried out on more realistic AO data.

For further improvements of the wavefront control scheme itself, one might also want to consider the possibility of additional constraint, e.g. to limit the possible difference between the neighboring calculated actuator commands into account for safe operation. The effect of adding regularization should also be analyzed, especially as it may impact the achievable performances for the ADMM and AS methods by the sparsity pattern of the preconditioners.

Finally, for a practical real-time implementation, the proposed methods should be evaluated in the context of their suitability for a parallel GPU architecture; the first steps in this direction have already been made through our implementation of PABB.

Chapter 5

Design of sparse aperture mask

In this chapter, we shift our attention to another important part of adaptive optics, namely wavefront sensing. In particular, we will examine an interesting optimization problem arising in the design of a so-called *sparse aperture mask (SAM)*. The SAM is a novel low-order wavefront sensor currently under development for its potential application in the direct imaging of exoplanets using space-based telescopes. It has been advocated for previously by Subedi et al. [63], where the proof-of-concept and underlying principles were laid out. The work therein also serves a feasibility study of the methodology in terms of potential use in future space missions. Here, we focus on the more mathematical aspect of optimizing the shape of the SAM, which is crucial for allowing it to reach its maximum potential and competitiveness with other sensors. We will also discuss how the Kronecker structures appearing in the problem can be exploited during the proposed solution process.

The optimization presented here was part of a collaboration with and serves as a contribution to the work of Hari Subedi[1]*, and relies on the simulations and* MATLAB *code he developed during the proof-of-concept phase of the sensor design to generate the models and evaluate the performance of SAMs.*

5-1 Background

The following brief background review of exoplanet imaging using coronagraphs and the principles of the SAM wavefront sensor are based on the paper of Subedi et al. [63]. An extensive discussion on the motivation behind and the principles of direct-imaging techniques are well outside the scope of this work, and we focus on the SAM shape design problem at hand.

The main difficulty in the direct imaging of exoplanets stems from the fact that they are overshadowed by the brightness of their host star. A coronagraphic optical system can be employed to address this issue by blocking out most of the starlight within the vicinity of

[1]Graduate student under the supervision of Prof. Jeremy Kasdin at Princeton University, Department of Mechanical and Aerospace Engineering.

the host, allowing the exoplanets to be observed. An example schematic for such a system is depicted in Figure 5-1. Here, a binary mask called the shaped pupil (SP) serves as the coronagraph, whose shape is designed in a way such that high-contrast regions are created by diffraction at the focal plane. A focal plane mask (FPM) then prevents most of the starlight from continuing on towards the science path, allowing nearby objects to become visible.

Current developments target contrast rates of around 10^{-9} between the brightness of the star and its exoplanets. Such orders of magnitude allow direct observations of (larger) Earth-like exoplanets, which can carry information about the composition of their atmosphere and other properties of scientific value. In order to achieve such high contrasts, it is essential that the incoming wavefronts be as flat as possible when they reach the shaped pupil, as its shape is designed under that assumption. Similarly to handling atmospheric disturbances for ground-based telescopes, an adaptive optics (AO) system can be employed to correct for the high-order wavefront aberrations caused by the imperfections of the optical system as well as the low-order ones due to resonant vibrations of the telescope. In this study, we focus on measuring disturbances of this latter type using the novel SAM wavefront sensor.

5-1-1 Wavefront sensing using a SAM

Similarly to achieving high contrast with a shaped pupil coronagraph, the methodology of wavefront sensing using a sparse aperture mask is based upon the principle of diffraction and can be explained following the schematic of Figure 5-1. As with other sensors, only the starlight reflected off the FPM is used for our measurements; this is of vital importance because we do not want to take away from the minimal amount of light coming from exoplanets towards the science path. The redirected starlight is then passed through the mask before being refocused and imaged by a detector camera.

Low-order wavefronts are generally described by so-called *Zernike modes*. Zernike modes form a basis on a circular region, which allows surfaces, such as wavefronts, to be expressed as their linear combination. Examples of these include common aberrations such as tip, tilt, and defocus, as seen on Figure 5-2. For more information on how this basis is constructed, we refer to the lecture notes on high resolution imaging by Verhaegen et al. [1].

The main idea behind the SAM sensor is that different Zernike modes will produce different diffraction patterns after passing through the mask. Thus, small low-order aberrations around a nominal wavefront can be detected by measuring the change in the image captured by the sensing detector. The disturbing wavefront is then reconstructed as a linear combination of a given set of Zernike modes; we estimate the corresponding coefficients of these such that the difference between the measured and expected detector image is minimized.

The proof-of-concept of the SAM sensor has been demonstrated in the work of Subedi et al. [63]. Compared to other sensors, such as the Zernike phase-contrast WFS [64] where the performance of a Shack-Hartmann lenslet array is enhanced by introducing phase shifts, the SAM is at a disadvantage because it blocks a significant portion of the starlight available for measurements. This is because masks which can properly distinguish different Zernike modes through diffraction tend to have sparse structures; there is a trade-off between these two aspects. For increasingly sparse masks, the detected images become relatively noisier, which hampers the accuracy of wavefront reconstruction and thus the achievable performance during closed-loop control.

This behavior was observed for the sample SAM seen on Figure 5-1 below, which was used to conduct the preliminary study of the sensor's performance through realistic simulations. However, the shape of this SAM was determined from a set of intuitive structures using Monte Carlo experiments. The authors expressed hope that with a proper optimization conducted to design the shape of the mask, its performance might become comparable to currently available wavefront sensors. This would be very beneficial, because manufacturing a binary mask is simpler and less expensive than manufactirng e.g. a microlenslet array for Shack-Hartmann (SH) sensors.

In this work, we contribute to the development of the SAM sensor by formulating its design problem under a mathematical framework which allows us to determine its shape through nonlinear optimization. However, our discussion here is limited to the aspects of accelerating the speed of this optimization procedure by exploiting the Kronecker structures within system matrices. For a detailed analysis of the resulting SAM masks and their closed-loop performance compared to other sensors for a variety of coronagraph-integrated telescope systems, we refer to the collaborative paper [65] (in preparation).

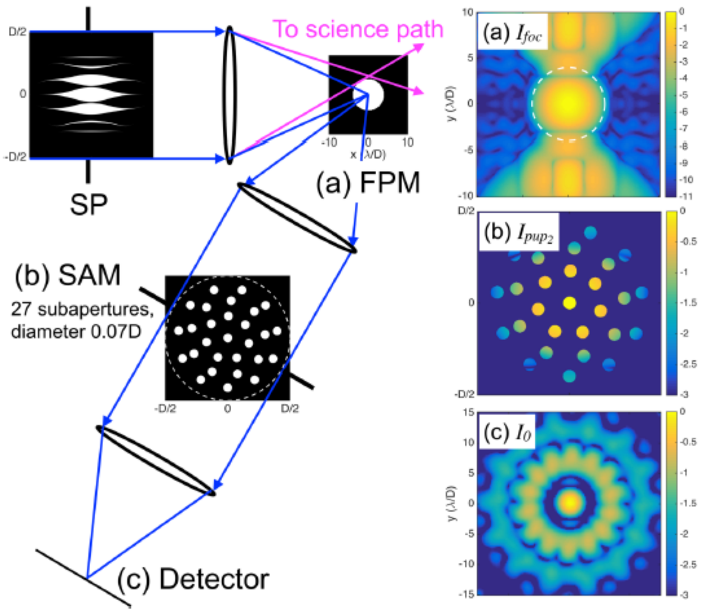
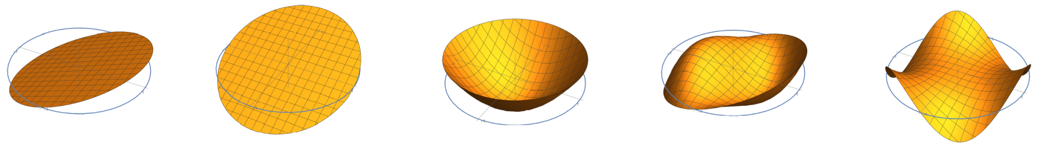


Figure 5-1: Simplified schematic diagram of high-contrast imaging and wavefront sensing using a shaped pupil and a sparse aperture mask. The imagse (a), (b), and (c) show the intensity distribution of a flat wavefront entering the SP at different planes of the optical tube [63].



(a) tip **(b)** tilt **(c)** defocus **(d)** coma **(e)** trefoil

Figure 5-2: Shapes of sample Zernike modes [1].

5-2 Problem formulation

Using Fourier optics, it can be shown that the sensing equation of the SAM takes the form:

$$\boldsymbol{t} = \mathbf{H}(\mathbf{M})\boldsymbol{x}. \tag{5-1}$$

Here $\boldsymbol{t} \in \mathbb{R}^{m^2}$ is the difference between the pixel intensity values of the nominal and perturbed m by m images obtained after allowing the distorted wavefront to pass through the SAM. The unknown $\boldsymbol{x} \in \mathbb{R}^p$ contains the coefficients for the p number of Zernike modes which will allow its reconstruction. Typical values for low-order wavefront sensors are around $p = 14$ and $m = 32$. The matrix $\mathbf{H} \in \mathbb{R}^{m^2 \times p}$ is the so-called *modal matrix* relating these two quantities as a function of the SAM. The kth column $\mathbf{H}_k \in \mathbb{R}^{m^2}$ describes the response of the system to the respective kth Zernike mode. Note that $\mathbf{H}$ is a very tall matrix. Finally, the SAM itself is represented by the binary matrix $\mathbf{M} \in \mathbb{R}^{n \times n}$ which describes its transmission at a grid of discretized points. A transmission of 0 corresponds to completely blocking, while a value of 1 corresponds to fully allowing light to pass through a given point of the mask. To make our optimization tractable, however, we will employ a convex relaxation and allow each element of $\mathbf{M}$ to take on any value between these two extremes. Let us denote the set of matrices whose values are within these bounds by $\mathcal{M}$:

$$\mathcal{M} := \left\{ \mathbf{M} \in \mathbb{R}^{n \times n} \;\middle|\; 0 \leq [M]_{ij} \leq 1, \ \forall i, j = 1, \ \ldots, \ n \right\}; \tag{5-2}$$

we must thus have $\mathbf{M} \in \mathcal{M}$. For the results presented in this chapter, we used a discretization of $n = 64$, though this resolution should be increased to allow finer manufacturing.

The challenge is to determine the mask $\mathbf{M}$ such that wavefront reconstruction can be performed in an optimal manner. To this end, we first give a brief analysis of the trade-off between distinguishing the Zernike modes and allowing a large light throughput of the mask.

The quality of our reconstructed wavefront can be measured by the error in the obtained Zernike coefficient vector. Due to the presence of measurement noise[2], the estimated $\hat{\boldsymbol{x}}$ coefficients are obtained as the solution to the following overdetermined linear system:

$$\hat{\boldsymbol{t}} = \boldsymbol{t} + \boldsymbol{v} = \mathbf{H}(\mathbf{M})\hat{\boldsymbol{x}}. \tag{5-3}$$

for some noise $\boldsymbol{v}$ corrupting the true measurements $\boldsymbol{t}$ for which the true $\boldsymbol{x}$ coefficients satisfy the sensing equation (5-1) explicitly. The sensitivity of the solution $\hat{\boldsymbol{x}}$ to this equation due to the presence of the measurement error can be characterized by the following inequality [28]:

$$\frac{\|\hat{\boldsymbol{x}} - \boldsymbol{x}\|_2}{\|\boldsymbol{x}\|_2} \leq \kappa(\mathbf{H}(\mathbf{M})) \cdot \frac{\|\boldsymbol{v}\|_2}{\|\boldsymbol{t}\|_2}. \tag{5-4}$$

Here $\kappa(\cdot)$ denotes the condition number of a given matrix, defined in terms of its maximum and minimum singular values as:

$$\kappa(\cdot) = \frac{\sigma_{\max}(\cdot)}{\sigma_{\min}(\cdot)}. \tag{5-5}$$

The condition number is a good indicator of how well we are able to distinguish the different Zernike modes; for different enough responses (i.e. columns of $\mathbf{H}$), it is expected to be low.

[2]Here we assume that the error in the modal matrix is negligible compared to the measurement noise.

As for the measurement error, here we assume the typical model of the form:

$$v_p = v_p^{\text{shot}} + v_p^{\text{read}}, \tag{5-6}$$

where each pixel p is independently affected by so-called *shot* and *read-out noise* [66]. The former is caused by quantum variations in the number of sensed photons and is modeled by a Poisson distribution of the quantity $t_p + v_p^{\text{shot}}$, which we've separated in terms of the average and zero-mean noise component. For N_p incoming photons, $t_p = N_p$ and v_p^{shot} has a variance of N_p according to the Poisson distribution:

$$E\left[\left(v_p^{\text{shot}}\right)^2\right] = N_p. \tag{5-7}$$

where $E[\cdot]$ denotes the expected value operator. The read-out noise is due to errors arising during image acquisition, and is typically modeled as a zero-mean Gaussian distribution, independent for each pixel and of the measured light intensity. We thus assume v_p^{read} is a constant variance s^2:

$$E\left[\left(v_p^{\text{read}}\right)^2\right] = s^2. \tag{5-8}$$

Combining these two effect, the variance of the entire noise v_p can be obtained by the simple additive rule of a small number of independent errors, i.e:

$$E\left[v_p^2\right] = N_p + s^2, \tag{5-9}$$

which yields the following relation between the expected relative error for the p pixel:

$$E\left[\frac{\|v_p\|_2}{\|t_p\|_2}\right] = \frac{\sqrt{N_p + s^2}}{N_p}. \tag{5-10}$$

This relation shows the importance of having a large amount of incoming photons available for our measurements; the relative noise of each pixel measurement diminishes with the amount of light, as can be intuitively expected. The results could be extended to obtain an explicit expression for the entire relative error in (5-4), though this leads to the same conclusions and we do not give the derivations here.

Our analysis and the reconstruction error bound given by (5-4) suggests the following observations:

(i) The modal matrix $\mathbf{H}(\mathbf{M})$ should be well-conditioned in order to minimize the sensitivity of the reconstructed Zernike modes to possible measurement errors.

(ii) Allowing more light to pass through the mask will decrease the relative error in the measurements, leading to better reconstruction as well.

We can expect a trade-off between the achievable conditioning and the light throughput of the SAM. The design problem is thus formulated as follows.

Problem 5.1. (SAM design) Determine the shape of the SAM mask $\mathbf{M} \in \mathcal{M}$ for which the trade-off between the conditioning of the modal matrix $\mathbf{H}(\mathbf{M})$ and the achieved light throughput allows wavefront reconstruction with minimal error.

Remark. *In this thesis work, however, we focus on the slightly different formulation of maximizing the light throughput given a bound on the conditioning. This will be better suited for our purposes as it is more well-defined mathematically. For the detailed analysis including the achieved reconstruction errors, we refer to our paper paper [65] (in preparation).*

5-2-1 Structure of the modal matrix

Before presenting our solution proposals and results for the SAM design problem, we briefly remark on the structure of the modal matrix $\mathbf{H}(\mathbf{M})$. The detailed derivations are outside the scope of this report and can be found in the original and upcoming papers [63] and [65]. An expression for each kth column of the modal matrix is obtained by propagating the kth Zernike mode through the optical system depicted in Figure 5-1, from the shaped pupil until the detector plane, and forming the difference with the nominal (plane) wavefront's image, which is computed in a similar manner. The propagations are done in the framework of Fourier optics, which relies heavily on 2D Fourier transforms. This is where Kronecker structures come into play; it can be shown that the Fourier transform $\hat{\mathbf{F}}$ of a quantity $\mathbf{F}$ from an n by n to an m by m grid can be computed using matrix formalism [67]:

$$\text{vec}(\hat{\mathbf{F}}) = (\mathbf{K} \otimes \mathbf{K})\, \text{vec}(\mathbf{F}) \tag{5-11}$$

which becomes:

$$\hat{\mathbf{F}} = \mathbf{K}\mathbf{F}\mathbf{K}^{\mathrm{T}}. \tag{5-12}$$

In the context of determining the modal matrix, the propagations until the SAM plane can be pre-calculated, and the kth column of $\mathbf{H}(\mathbf{M})$ can be expressed as:

$$\mathbf{H}_k(\mathbf{M}) = \boldsymbol{a}_1(\mathbf{M}) \odot \boldsymbol{a}_{2,k}(\mathbf{M}) + \boldsymbol{a}_3(\mathbf{M}) \odot \boldsymbol{a}_{4,k}(\mathbf{M}), \tag{5-13}$$

where $\boldsymbol{a}_1,\ \boldsymbol{a}_3 \in \mathbb{R}^{m^2}$ and each $\boldsymbol{a}_{2,k},\ \boldsymbol{a}_{4,k} \in \mathbb{R}^{m^2}$ for $k = 1, \dots, p$. They are given as:

$$\boldsymbol{a}_1(\mathbf{M}) = \text{vec}\left[\mathbf{K}_I\left(\mathbf{E}_0 \odot \mathbf{M}\right)\mathbf{K}_I^{\mathrm{T}} - \mathbf{K}_R\left(\mathbf{E}_0 \odot \mathbf{M}\right)\mathbf{K}_R^{\mathrm{T}}\right], \tag{5-14a}$$

$$\boldsymbol{a}_{2,k}(\mathbf{M}) = \text{vec}\left[\mathbf{K}_I\left(\mathbf{E}_k \odot \mathbf{M}\right)\mathbf{K}_R^{\mathrm{T}} + \mathbf{K}_R\left(\mathbf{E}_k \odot \mathbf{M}\right)\mathbf{K}_I^{\mathrm{T}}\right], \tag{5-14b}$$

$$\boldsymbol{a}_3(\mathbf{M}) = \text{vec}\left[\mathbf{K}_I\left(\mathbf{E}_0 \odot \mathbf{M}\right)\mathbf{K}_R^{\mathrm{T}} + \mathbf{K}_R\left(\mathbf{E}_0 \odot \mathbf{M}\right)\mathbf{K}_I^{\mathrm{T}}\right], \tag{5-14c}$$

$$\boldsymbol{a}_{4,k}(\mathbf{M}) = \text{vec}\left[\mathbf{K}_R\left(\mathbf{E}_k \odot \mathbf{M}\right)\mathbf{K}_R^{\mathrm{T}} - \mathbf{K}_I\left(\mathbf{E}_k \odot \mathbf{M}\right)\mathbf{K}_I^{\mathrm{T}}\right]. \tag{5-14d}$$

Here $\mathbf{K}_R \in \mathbb{R}^{m\times n}$ and $\mathbf{K}_I \in \mathbb{R}^{m\times n}$ are the real and imaginary parts of the Fourier transform matrix $\mathbf{K} \in \mathbb{C}^{m\times n}$ from the mask to the detector plane. The matrices $\mathbf{E}_0 \in \mathbb{R}^{n\times n}$ and each $\mathbf{E}_k$ represent the pre-calculated propagations of the nominal wavefront and the Zernike modes until the SAM. These matrices (along with $\mathbf{K}$) are dense and their exact expressions can be found in the cited papers. Note that due to the element-wise (Hadamard) products in Equation (5-13), $\mathbf{H}$ is a quadratic function of the mask $\mathbf{M}$.

In the following, we present two possible approaches to solving the SAM design problem. They differ in the way they aim to achieve a well-conditioned modal matrix. In both cases, we will highlight the advantages and disadvantages of the formulations, as well as discuss how the Kronecker structures within the modal matrix can be exploited for gains in computational efficiency. Due to the complexity of the problem, however, this will again be done in the context of matrix-vector multiplications, though here we argue that the sparse evaluation (A-11) could be slightly more effective than the dense. We conclude by proposing a solution framework which combines the strengths of our two approaches.

5-3 Solution approaches

5-3-1 Frobenius norm based approach

We first discuss a simple formulation based on the observation that the modal matrix is well-conditioned if its columns are orthogonal. Intuitively, this corresponds to the idea that these columns contain the responses to given Zernike modes and should be as different from one another as possible to allow them to be distinguished.

Mathematically, we can express how close the columns are to being orthogonal using the difference of the scaled modal matrix $\gamma\mathbf{H}(\mathbf{M})$ to some orthogonal matrix $\mathbf{Q} \in \mathbb{R}^{m^2 \times p}$ in the Frobenius-norm as:

$$\left\|\gamma\mathbf{H}(\mathbf{M}) - \mathbf{Q}\right\|_F, \tag{5-15}$$

which is a commonly used measure as it is analytically simple to work with. The parameter γ is introduced as a scaling factor as we do want to limit the throughput by constraining the columns of $\mathbf{H}$ to have unit norm. The problem of choosing the most appropriate $\mathbf{Q}$ matrix in the above measure is a special case of the so-called *orthogonal Procrustes problem.* The solution to this problem is well-known [28] and can be expressed as $\mathbf{Q} = \mathbf{U}\mathbf{V}^{\mathrm{T}}$ given the SVD decomposition $\mathbf{H}(\mathbf{M}) = \mathbf{U}\mathbf{S}\mathbf{V}^{\mathrm{T}}$. We thus have the measure:

$$\left\|\gamma\mathbf{U}\mathbf{S}\mathbf{V}^{\mathrm{T}} - \mathbf{U}\mathbf{V}\right\|_F = \left\|\gamma\mathbf{S} - \mathbf{I}_p\right\|_F. \tag{5-16}$$

An expression as a direct function of $\mathbf{H}(\mathbf{M})$ can be obtained if we instead go for the stricter formulation:

$$\left\|\gamma\mathbf{S}^2 - \mathbf{I}_p\right\|_F, \tag{5-17}$$

which, using the fact that $\mathbf{V}\mathbf{S}^2\mathbf{V}^{\mathrm{T}} = \mathbf{H}(\mathbf{M})^{\mathrm{T}}\mathbf{H}(\mathbf{M})$, can be rewritten as:

$$\left\|\gamma\mathbf{H}(\mathbf{M})^{\mathrm{T}}\mathbf{H}(\mathbf{M}) - \mathbf{I}_p\right\|_F. \tag{5-18}$$

Notice that the parameter γ actually gives a good indication of the amount of light passing through the SAM. A mask that allows a lot of light to pass through it will yield a modal matrix with large singular values, requiring a small scaling factor to minimize the above norm measure. Therefore, lower values of γ correspond to a higher average of singular values and thus a greater throughput. This observation leads to the following formulation of the SAM design problem:

$$\underset{\gamma\in\mathbb{R},\ \mathbf{M}\in\mathcal{M}}{\text{minimize}} \quad \gamma \tag{5-19}$$

$$\text{subject to} \quad \left\|\gamma\mathbf{H}(\mathbf{M})^{\mathrm{T}}\mathbf{H}(\mathbf{M}) - \mathbf{I}_p\right\|_F \leq c \cdot \left\|\mathbf{I}_p\right\|_F \tag{5-20}$$

for some constant c. We note that γ was introduced to scale $\mathbf{H}(\mathbf{M})^{\mathrm{T}}\mathbf{H}(\mathbf{M})$ as opposed to $\mathbf{I}_p$ in order to allow the constraint to be written in an intuitive manner; c expresses the maximum relative error we accept of the norm difference in a way such that the right-hand side of the constraint does not depend on $\mathbf{H}(\mathbf{M})$. This formulation does not perfectly fit our goal of tackling the SAM design problem because the Frobenius-norm constraint (5-20) only implies, but does not explicitly relate to the well-conditioning of the modal matrix. This is the main disadvantage of this approach and we can expect it to produce suboptimal results.

The constraint (5-20) is a non-convex function of the mask as can be seen e.g. by the presence of the scaling factor γ. However, it would remain non-convex even without this (needed) factor or by scaling $\mathbf{I}_p$, since the modal matrix is quadratic in $\mathbf{M}$ and the composition of convex functions is not necessarily convex, as in this case. The advantage of this formulation is still in its simplicity in the sense that the values and gradients of both the objective and the constraint can be easily derived and supplied to a nonlinear solver for an accelerated solution. As we will see, the Kronecker structures will allow these computations to be evaluated even faster than without them. Due to the large number of variables, we chose the interior-point solver of MATLAB's `fmincon` function to find the solution. To proceed, we now discuss how the sparse form (A-11) of applying Kronecker product multiplications can be effectively utilized in this optimization problem to allow fast computation of the constraint and its gradient.

Computation of constraint. When computing the columns of the modal matrix $\mathbf{H}(\mathbf{M})$, the Equations (5-14) already exploit the Kronecker structure. However, rewriting e.g. $\boldsymbol{a}_1(\mathbf{M})$ in the sparse formulation using the properties of Kronecker products, we have[3]:

$$\begin{aligned}
\boldsymbol{a}_1(\mathbf{M}) &= \operatorname{vec}\left[\mathbf{K}_I\left(\mathbf{E}_0 \odot \mathbf{M}\right)\mathbf{K}_I^{\mathrm{T}} - \mathbf{K}_R\left(\mathbf{E}_0 \odot \mathbf{M}\right)\mathbf{K}_R^{\mathrm{T}}\right] \\
&= \left(\mathbf{K}_I \otimes \mathbf{K}_I\right)\operatorname{vec}\left(\mathbf{E}_0 \odot \mathbf{M}\right) - \left(\mathbf{K}_R \otimes \mathbf{K}_R\right)\operatorname{vec}\left(\mathbf{E}_0 \odot \mathbf{M}\right) \\
&= \left(\mathbf{K}_I \otimes \mathbf{K}_I\right)\mathbf{E}_0^D \boldsymbol{m} - \left(\mathbf{K}_R \otimes \mathbf{K}_R\right)\mathbf{E}_0^D \boldsymbol{m} \\
&= \left(\mathbf{K}_I \otimes \mathbf{K}_I - \mathbf{K}_R \otimes \mathbf{K}_R\right)\mathbf{E}_0^D \boldsymbol{m} \\
&= \Big(\left(\mathbf{K}_I \otimes \mathbf{I}_m\right)\cdot\left(\mathbf{I}_n \otimes \mathbf{K}_I\right)\cdot\mathbf{E}_0^D - \left(\mathbf{K}_R \otimes \mathbf{I}_m\right)\cdot\left(\mathbf{I}_n \otimes \mathbf{K}_R\right)\cdot\mathbf{E}_0^D\Big)\boldsymbol{m},
\end{aligned} \tag{5-21}$$

where we have introduced $\mathbf{E}_0^D \in \mathbb{R}^{n^2 \times n^2}$ and $\boldsymbol{m} \in \mathbb{R}^{n^2}$ as:

$$\mathbf{E}_0^D = \operatorname{diag}\left(\operatorname{vec}(\mathbf{E}_0)\right) \qquad \text{and} \qquad \boldsymbol{m} = \operatorname{vec}(\mathbf{M}). \tag{5-22}$$

This formulation highlights a slight advantage of applying Kronecker product multiplications in their sparse form as opposed to their dense form. In the dense expression of $\boldsymbol{a}_1(\mathbf{M})$, we need to evaluate the Hadamard product $(\mathbf{E}_0 \odot \mathbf{M})$ before the left and right multiplications by the Kronecker pairs. On the other hand, the sparse form obtained as a result of the derivation allows us to pre-calculate the matrices

$$\left(\mathbf{I}_n \otimes \mathbf{K}_I\right)\cdot\mathbf{E}_0^D \qquad \text{and} \qquad \left(\mathbf{I}_n \otimes \mathbf{K}_R\right)\cdot\mathbf{E}_0^D$$

while maintaining the same sparse structure of the two Kronecker terms as before, i.e. without degrading the performance of evaluating multiplications with these terms. We would not be able to perform such a trick in the dense formulation. However, it must be noted that evaluating the Hadamard product would be an $\mathcal{O}(n^2)$ operation, while the Kronecker terms still require $\mathcal{O}(n^3)$ flops, so the gain is definitely not substantial, especially for increasingly large matrices. Another advantage of the sparse form is that we can easily accommodate cases where only parts of the entire square embedding $\mathbf{M}$ of the mask are unknown while the rest have values equal to 0. This is useful for fixing outer values near the edge to reduce the number of unknowns. With the sparse formulation, we can delete any element of $\boldsymbol{m}$ and the corresponding columns of the precalculated $(\mathbf{I}_n \otimes \mathbf{K}_R)\cdot\mathbf{E}_0^D$ with ease in most commercial software, e.g. MATLAB; however, although possible by treating $\mathbf{M}$ itself as a sparse matrix with large fill-in, this would be more cumbersome to implement using the dense formulation.

[3]Due to the similarity of the expressions for $\boldsymbol{a}_1$, $\boldsymbol{a}_3$ and each $\boldsymbol{a}_{2,k}$ $\boldsymbol{a}_{4,k}$ in terms of the unknown mask $\mathbf{M}$, we give the derivations for only one of them.

Computation of constraint gradient. The sparse formulation is also helpful in that the gradient with respect to the vectorized mask $\boldsymbol{m}$ can be easily derived. To lighten the notation, we will not show that the quantities $\mathbf{H}$, $\boldsymbol{a}_1$, etc... are explicitly functions of the mask.

To begin, the differential of the kth column $\mathbf{H}_k$ of the modal matrix in the direction of some $\delta\boldsymbol{m}$ can be expressed from the Equation (5-13) as:

$$\begin{aligned}\delta\mathbf{H}_k &= \left(\frac{\partial\boldsymbol{a}_1}{\partial\boldsymbol{m}}\delta\boldsymbol{m}\right)\odot\boldsymbol{a}_{2,k}+\boldsymbol{a}_1\odot\left(\frac{\partial\boldsymbol{a}_{2,k}}{\partial\boldsymbol{m}}\delta\boldsymbol{m}\right)+\left(\frac{\partial\boldsymbol{a}_3}{\partial\boldsymbol{m}}\delta\boldsymbol{m}\right)\odot\boldsymbol{a}_{4,k}+\boldsymbol{a}_3\odot\left(\frac{\partial\boldsymbol{a}_{4,k}}{\partial\boldsymbol{m}}\delta\boldsymbol{m}\right)\\ &:= \boldsymbol{a}^D_{2,k}\left(\frac{\partial\boldsymbol{a}_1}{\partial\boldsymbol{m}}\delta\boldsymbol{m}\right)+\boldsymbol{a}^D_1\left(\frac{\partial\boldsymbol{a}_{2,k}}{\partial\boldsymbol{m}}\delta\boldsymbol{m}\right)+\boldsymbol{a}^D_{4,k}\left(\frac{\partial\boldsymbol{a}_3}{\partial\boldsymbol{m}}\delta\boldsymbol{m}\right)+\boldsymbol{a}^D_3\left(\frac{\partial\boldsymbol{a}_{4,k}}{\partial\boldsymbol{m}}\delta\boldsymbol{m}\right), \qquad (5\text{-}23)\end{aligned}$$

where the introduced $\boldsymbol{a}^D_{i(,k)} = \operatorname{diag}(\boldsymbol{a}_{i(,k)})$ are diagonal $\mathbb{R}^{m^2\times m^2}$ matrices for each $i = 1,\ldots,4$ and $k = 1,\ldots,p$. The four summations in this expression all have similar form, and for the sake of evaluating an efficient computational scheme we only examine the first of these terms. The sparse expression (5-21) for $\boldsymbol{a}_1$ seen as a function of the vectorized mask allows us to readily find its gradient with respect to $\boldsymbol{m}$:

$$\frac{\partial\boldsymbol{a}_1}{\partial\boldsymbol{m}} = (\mathbf{K}_I\otimes\mathbf{K}_I-\mathbf{K}_R\otimes\mathbf{K}_R)\,\mathbf{E}^D_0. \qquad (5\text{-}24)$$

Similarly structured expressions can be obtained for the gradients of the other $\boldsymbol{a}_{i(,k)}$ terms with respect to the unknown $\boldsymbol{m}$.

Let us now derive the gradient of the constraint (5-20) with respect to both $\boldsymbol{m}$ and the variable γ. Denoting it by $g(\mathbf{H},\gamma)$, we can write:

$$\begin{aligned}g(\mathbf{H},\gamma) &= \left\langle\gamma\mathbf{H}^{\mathrm{T}}\mathbf{H}-\mathbf{I}_p,\ \gamma\mathbf{H}^{\mathrm{T}}\mathbf{H}-\mathbf{I}_p\right\rangle \qquad (5\text{-}25)\\ &= \left\langle\gamma^2\mathbf{H}\mathbf{H}^{\mathrm{T}}\mathbf{H},\ \mathbf{H}\right\rangle-2\left\langle\gamma\mathbf{H},\ \mathbf{H}\right\rangle+\left\langle\mathbf{I}_p,\ \mathbf{I}_p\right\rangle, \qquad (5\text{-}26)\end{aligned}$$

where the introduced $\mathbf{F}\in\mathbb{R}^{m^2,p}$. The change in the direction of $\delta\mathbf{H}$ is thus[4]:

$$\delta g(\mathbf{H},\gamma,\delta\mathbf{H}) = \left\langle4\gamma^2\mathbf{H}\mathbf{H}^{\mathrm{T}}\mathbf{H}-4\gamma\mathbf{H},\ \delta\mathbf{H}\right\rangle := \left\langle\mathbf{F},\ \delta\mathbf{H}\right\rangle. \qquad (5\text{-}27)$$

Examining only the kth column of this relation and only the contribution of the first term of the sum in (5-23) for $\delta\mathbf{H}_k$, substituting in our previous results (5-24) as well we have:

$$\begin{aligned}\delta g^{\mathrm{1st}}_k(\mathbf{H},\gamma,\delta\mathbf{H}_k) &= \left\langle\mathbf{F}_k,\ \delta\mathbf{H}_k\right\rangle \qquad (5\text{-}28)\\ &= \left\langle\mathbf{F}_k,\ \boldsymbol{a}^D_{2,k}(\mathbf{K}_I\otimes\mathbf{K}_I-\mathbf{K}_R\otimes\mathbf{K}_R)\,\mathbf{E}^D_0\delta\boldsymbol{m}\right\rangle \qquad (5\text{-}29)\\ &= \left\langle\mathbf{E}^D_0(\mathbf{K}_I\otimes\mathbf{K}_I-\mathbf{K}_R\otimes\mathbf{K}_R)^{\mathrm{T}}\boldsymbol{a}^D_{2,k}\mathbf{F}_k,\ \delta\boldsymbol{m}\right\rangle. \qquad (5\text{-}30)\end{aligned}$$

Using the sparse formulation of the Kronecker products, the effect of $\mathbf{E}^D_0$ can again be precalculated. The matrix-vector multiplications can then be carried out efficiently starting from the right with $\mathbf{F}_k\in\mathbb{R}^{m^2}$. Summing up all the contribution gives the gradient $\delta g(\mathbf{H},\gamma,\delta\mathbf{H})$.

Results A sample result of optimizing the mask for a relative error of $c = 0.75$ using the Frobenius-norm approach is depicted in Figure 5-3. As we will later see, while the optimization does run quite fast, the result is simply not optimal enough as we are only approximating the criteria for a well-conditioned modal matrix. Note that in this run, horizontal symmetry was explicitly enforced as the solution tended toward such a state and this allowed even faster computations. Studying why such symmetry arises is of future interest.

[4]The gradient with respect to γ can be easily derived from the quadratic expression (5-26).

5-3-2 Condition number based approach

In this section we present a direct approach of formulating the SAM design objectives highlighted in Section 5-2 through. It is our hope that although the resulting optimization problem will be more complex than in the previous case, the obtained masks will be much more optimal in terms our design goals. This is essential for space missions as the mask cannot be replaced on the operational telescope.

The most straightforward manner to express the first goal (i) of having a well-conditioned modal matrix is to directly use the condition number of $\mathbf{H}(\mathbf{M})$ as the measure. As for the second one (ii), the initial idea was that the amount of open area (i.e. sum of transmission values) in the mask could be a good indication of its light throughput and should be used as the objective function to maximize. However, here we argue that this is not the case - due to diffraction, the intensity of the light reaching the SAM is not evenly distributed along its plane and the transmission values should at least be weighted accordingly. An even better indicator for the throughput, as explained in the Frobenius norm formulation, are the singular values of $\mathbf{H}(\mathbf{M})$. A higher average $\bar{\sigma}(\mathbf{H}(\mathbf{M}))$ of these expresses a stronger response as it relates to how the responses to the Zernike modes scale up as a function of the coefficients. We can thus formulate the corresponding optimization problem for the SAM design problem as:

$$\max_{\mathbf{M}\in\mathcal{M}} \quad \bar{\sigma}(\mathbf{H}(\mathbf{M})) \tag{5-31}$$

$$\text{s.t.} \quad \kappa(\mathbf{H}(\mathbf{M}) \leq \kappa_0 \tag{5-32}$$

for some constant κ_0. In practice, we found that condition numbers of $\kappa_0 = 20$-25 yield quite acceptable closed-loop performance and constitutes a good balance between the two design trade-offs; however, as mentioned earlier, this analysis is not a subject of this discussion.

The advantage of the condition number based approach is that we are explicitly optimizing for the quantities of interest, i.e. the conditioning of $\mathbf{H}$ and its average singular values, which relate to the light throughput of the mask. The disadvantage, however, is that the resulting optimization problem is highly nonlinear with possibly many local minima. While there are formulations of optimizing for condition numbers using convex programming [68], [69], in general these methods are only applicable to a small class of matrices, e.g. positive definite ones. In our case, the matrix $\mathbf{H}$ whose condition number we are interested in is a quadratic function of the mask, and the techniques described in the referred papers are not applicable. A more general formulation based on a smooth approximation of the condition number as a function of a composite expression, such as $\mathbf{H}(\mathbf{M})$, is given in Chen et a. [70], which could serve as a good starting point for a more efficient solution to the above outlined optimization problem. In this study, however, we simply used the interior-point method of the `fmincon` function within MATLAB to find possible solution, without supplying a user-defined gradient for either the objective or the constraint. In order to speed up the optimization and possibly avoid local minima, horizontal symmetry was explicitly enforced on the mask after we noticed that the solution seemed to tend towards such a shape[5]. We note that the efficient calculation of the modal matrix can still be implemented using the sparse Kronecker formulation (5-21).

A sample result from optimizing the mask for a condition number of $\kappa_0 = 25$ using the conditioner number based approach is shown in Figure 5-4. Note that the result is much more optimal than the one obtained using the Frobenius-norm formulation.

[5]It remains of interest to see why this symmetry appears to yield optimal solutions.

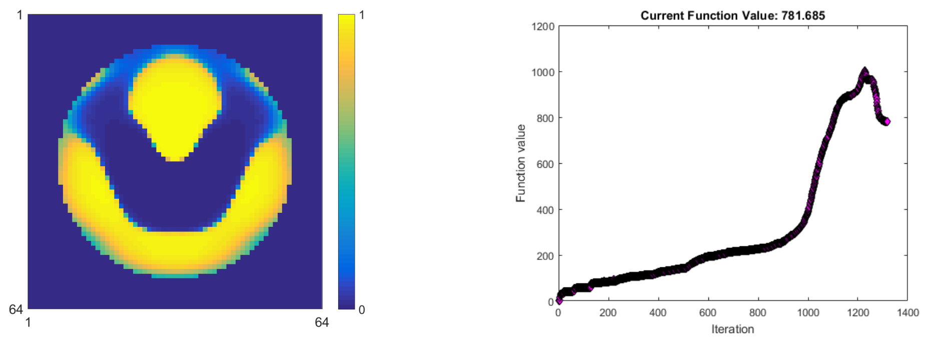


Figure 5-3: Results from the Frobenius-norm based approach to the SAM design problem with relative error constraint $c = 0.75$. The optimization was started from a random initial mask. (Left) Optimal mask $\kappa(\mathbf{H}) = 56.2$, $\bar{\sigma}(\mathbf{H}) = 0.533$; (Right) Objective curve from interior point optimization (scaled).

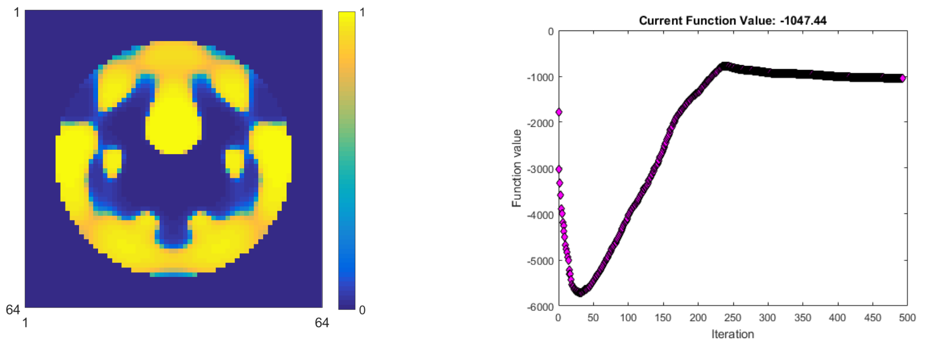


Figure 5-4: Results from the condition number based approach of maximizing the average singular values for the condition number $\kappa_0 = 25$. The optimization was initialized from the same random mask as in the figure above. (Left) Optimal mask $\kappa(\mathbf{H}) = 25.0$, $\bar{\sigma}(\mathbf{H}) = 0.419$; (Right) Interior point solution objective curve (scaled).

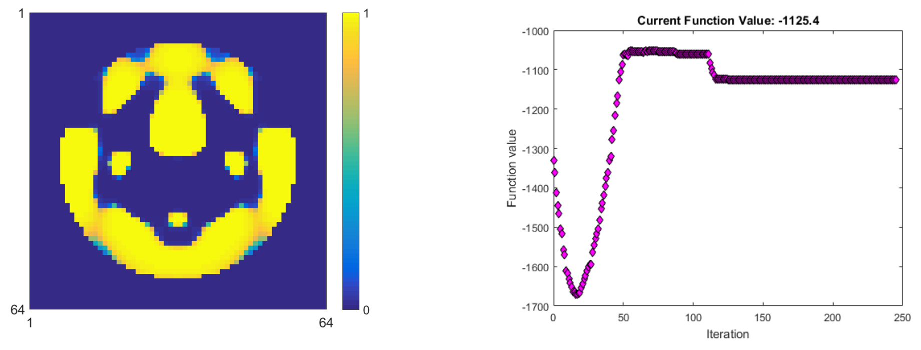


Figure 5-5: Results from maximizing the average singular values for the condition number $\kappa_0 = 25$ using the 100th iterate of the Frobenius-norm based solution as the initial mask. (Left) Optimal mask $\kappa(\mathbf{H}) = 25.0$, $\bar{\sigma}(\mathbf{H}) = 0.450$; (Right) Interior point solution objective curve (scaled).

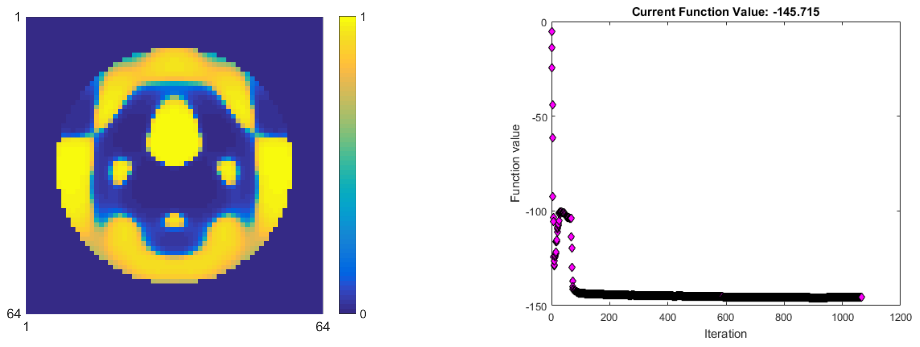


Figure 5-6: Results from maximizing the *minimum* singular value for the condition number $\kappa_0 = 25$ using a random the initial mask. (Left) Optimal mask $\kappa(\mathbf{H}) = 25.0$, $\bar{\sigma}(\mathbf{H}) = 0.391$; (Right) Interior point solution objective curve (scaled).

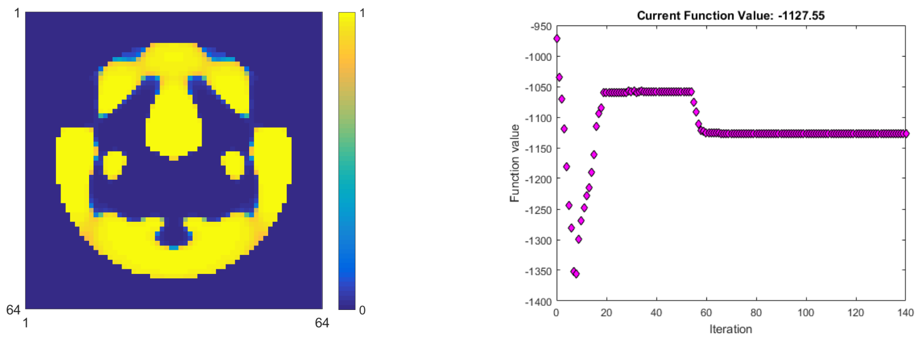


Figure 5-7: Results from from maximizing the average singular values for the condition number $\kappa_0 = 25$ using the 100th iterate of the result from above as the initial mask. (Left) Optimal mask $\kappa(\mathbf{H}) = 25.0$, $\bar{\sigma}(\mathbf{H}) = 0.451$; (Right) Interior point solution objective curve (scaled).

5-3-3 Proposed solution framework

The proposed solution to obtain a fully optimized SAM given a certain conditioning on the modal matrix, which allows the determination of the trade-off in the SAM design problem, is based on a combination of the two approaches outlined previously and our experiences gained while working with them.

The Frobenius-norm based approach runs about 6 time faster in our MATLAB implementation than the condition number based one[6]. Setting a small initial value for the parameter γ generally drives the solver towards a candidate solution with a large throughput. The results, however, are quite suboptimal; the achieved condition number are large because the Frobenius-norm difference to the identity is only an approximation to for a well-conditioned matrix. On the other hand, optimizing directly based on the spectral properties of $\mathbf{H}(\mathbf{M})$ yields much better solutions at the cost of being less computationally efficient. Due to the complexity of this formulation, our solver using `fmincon` only allows the gradient of both the objective and the constraint to be computed using finite differences[7], requiring many function evaluations for larger masks. The obtained solutions also seems more sensitive to the initial values of the unknowns than in the case of the Frobenius-norm solver.

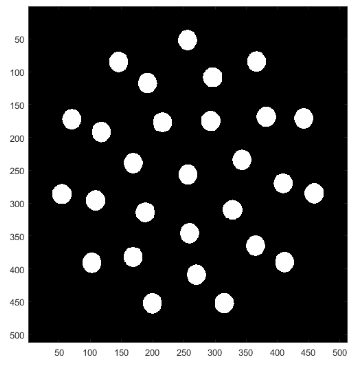
[6]Here the problem dimensions were $n = 64$, $m = 32$, and $p = 14$ for the mask size, image size, and number of Zernike modes in the low-order estimation, respectively.

[7]Note, however, that calculating these finite differences can be parallelized with suitable hardware.

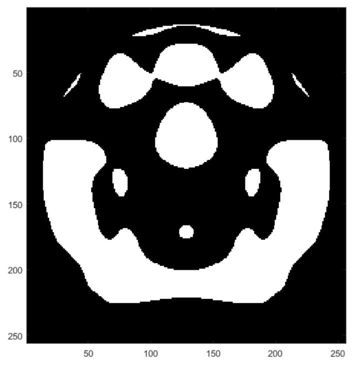
To combine the strengths of the two solution approaches, we propose to use the Frobenius-norm based solver to find an appropriate initial mask for the condition number based one. In our experience, the mask of Figure 5-3 starts to take shape within a few hundred iterations and there is no need to fully run the solver until convergence. The result would be suboptimal anyway, while an intermediate mask can also serve as a good initial estimate for the condition number based approach, as illustrated by Figure 5-5. We can see that in this case the latter solver takes much less iterations to find an optimal solution; the convergence using the default function settings is achieved in about 250 iterations as opposed to the 500 in Figure 5-4. We also noticed that maximizing the *minimum* singular value also leads to much faster initial convergence and can be effectively used to warm-start the solver aiming for the average. This observation is depicted in Figures 5-6 and 5-7[8].

We again emphasize that the optimal SAM for this particular optical system seems to exhibit horizontal symmetry; enforcing this yields leads to improved converge rates by a reduction of the problem size and the number of local minima. A further gain in efficiency can be obtained by fixing the outer perimeter of the mask and only optimizing within the depicted circles. This is possible because barely any light passes through the outer regions. Furthermore, it is worth noting that increasing the resolution of the SAM does not seem to impact the general form of the solution. Therefore, results from smaller problem sizes can be effective initial masks when increasing and solving the problem on a higher resolution. This also means that the quality of the mask is mostly retained if we binarize the obtained masks; this is an important if we wish for it to be manufactured.

A comparison of our fully optimized and binarized mask using a 50 % thresholding on a large 256 by 256 grid mask using the proposed solution framework is given by the figure below. It is clear that the optimized mask is superior in both the design goals of having (i) a well-conditioned matrix and (ii) allowing a large throughput of light. This also shows when implementing the wavefront sensor in a closed-loop control simulation; for more detailed study also comparing its effectiveness to other sensor, we refer to the paper [71] (in preparation).



(a) Original mask, $\kappa(\mathbf{H}) = 49.6$, $\bar{\sigma}(\mathbf{H}) = 0.0844$



(b) Optimized, $\kappa(\mathbf{H}) = 18.4$, $\bar{\sigma}(\mathbf{H}) = 0.297$

Figure 5-8: Initial and fully optimized masks. The condition number of the modal matrix has been reduced by over 60% while the average of its singular values increased by almost 350%.

[8]The minimum singular value is a concave function, while the average of the singular values (related to the nuclear norm) is a convex function of the modal matrix. During maximization, these become convex and concave, respectively; we suspect the observed behavior is related to the possibly more convex nature of optimizing for the minimum singular value. Of course, it is still not a convex problem as the optimization is done with respect to the mask, not the modal matrix.

5-4 Results and future work

In this chapter, we developed a framework which allows us to design the shape of an optimal sparse aperture mask for given coronagraphic telescope system. Our results serve as a contribution to the development of this novel wavefront sensor, which has potential application during future space missions aimed at the direct imaging of exoplanets.

The analysis of the design problem led to an interesting trade-off between the light throughput of the mask and its ability to distinguish Zernike modes for wavefront reconstruction. In the two proposed mathematical formulations for quantifying these trade-offs, Kronecker structures arise in the problem matrices because the propagation of Zernikes through the optical system can be efficiently computed using the matrix form of Fourier transforms. We argued that the structures can be exploited using the sparse formulation of evaluating Kronecker matrix multiplications during optimization. As opposed to the desnse formulation, this allows us to precompute products with diagonal matrices, although the gain here is marginal. A further advantage is that using the sparse form, we can readily retain only certain parts of the unknown mask in available optimization software. We discussed the strengths and weaknesses of the two potential problem formulations and presented an effective solution framework for the design problem by combining the advantages of each.

For the specific telescope system with the shaped pupil depicted in Figure 5-1, which served as a baseline for our study, a laboratory setup is available for realistic experimentation at the High Contrast Imaging Laboratory of Princeton University. The optimized SAM, as seen on Figure 5-8b, is currently being manufactured and its performance will be evaluated on the testbed. A further study of the competitiveness of the obtained SAMs compared to other wavefront sensors is also being conducted [71] (in preparation).

Chapter 6

Conclusions

In the following, we present the main results and observations from our work regarding the exploitation of Kronecker product structures both in theory and in practice. For more details and possible research directions on the individual topics discussed, please refer to the results summarized at the end of each chapter.

Our study of low Kronecker-rank matrices is motivated by their potential in compressing matrix representations and in reducing computational complexities associated with solution algorithms. Structure preservation plays an important role in their practical use regarding this area, as many algorithms rely on matrix operations or factorizations which might ruin these existing structures. The main contribution of this thesis is research into this topic, in particular results uncovered for the inverse approximation problem. Numerical applications include preconditioning Kronecker structured linear equations and the matrix sign iterations, as well as important practical engineering problems from the field of adaptive optics.

A theoretical study of the low Kronecker-rank inverse approximation problem has led to the development of a new solution approach where the subproblems of an ALS scheme are solved in a distributed least-squares like manner as opposed to by forming the normal equations. A detailed computational complexity analysis revealed that the resulting algorithm is slightly slower than the latter, although it is more robust in exchange. A comprehensive review of the two approaches was also presented in terms of how they allow imposing sparsity or symmetry on the unknowns Kronecker pairs, which is important for practical considerations as it can be very beneficial to preserve these addition structures.

We conducted a numerical analysis to examine the feasibility of preserving low Kronecker-rank structures during the inverse operation. Preliminary results suggest that this will remain an open and challenging issue in case we are working with general low Kronecker-rank structures. For situations where the Kronecker singular values exhibit a decaying rate, however, as often occurs in practice, low separation rank structures can still serve as good approximations for inverses. We uncovered the importance of fully optimizing all Kronecker pairs in this low separation rank representation at once as opposed to a progressive term-by-term approach employed to tackle higher-order tensors in the literature. Unfortunately, in structure preserving algorithms such as the matrix sign iterations, the accumulating approximation errors can quickly deteriorate the accuracy of the results, except for perhaps very special additionally structured cases which is a subject of future research.

A literature survey suggests that for more complicated problems, the most promising approach to exploiting Kronecker structures is through the accelerated matrix-vector multiplications they allow. The second part of our work offers solution proposals to two engineering problems of practical significance, in the context of exploiting the Kronecker structures appearing in them.

Examining possible efficient solutions to the adaptive optics wavefront control problem for turbulence correction, we found the simple first-order projected alternating Barzilai-Borwein (PABB) method to hold the most potential against the two other explored options, namely active sets (AS) and alternating direction method of multipliers (ADMM). Although low Kronecker-rank inverse approximations serve as good preconditioners for problems encountered in these latter two, the fact remains that we cannot use pre-calculated factorizations such as with a different approach exploiting sparsity without the Kronecker structures. This leads to their relatively worse performance compared to PABB as opposed to how the sparse variants compare to the sparse PABB in the literature. Whether PABB itself is more efficient than the in the sparse case will depend on the exact fill-in factors and Kronecker ranks of the identified system model. Carrying out matrix-vector multiplications on a real-time hardware also serves as a demonstrative example when the dense and small formulation of their evaluation is useful. A parallelized graphics processing unit (GPU) implementation of the most promising PABB approach is already under development. The three explored options should also be compared with more realistic data to re-evaluate the initial observations regarding their feasibility made herein.

In the other practical engineering application, we examined the shape design problem of a novel low-order wavefront sensor called a sparse aperture mask. Our analysis revealed that for an optimal mask, the trade-off between its light throughput and its ability to distinguish a certain basis set of low-order Zernike modes should be balanced. Results obtained from approximate solutions, such as using the Frobenius-norm difference to the identity matrix for well-conditioning, are not satisfactory enough to justify their use in important space missions. In case of the nonlinear optimization problems proposed to optimize the mask shapes, we argue that the large and sparse form of evaluating Kronecker matrix-vector is beneficial. The framework developed for optimizing masks is applicable for various coronagraph-integrated telescopes, and a comparison of these to other wavefront sensors is being conducted both using simulations as well as in a laboratory environment. The practical advantage of the mask would be in its simplicity and ease of manufacturing.

Appendix A

Kronecker products

We give a brief introduction to the main properties of Kronecker products used throughout this thesis, along with computational complexities in the context of exploiting their structure through matrix-vector multiplications.

A-1 Properties of Kronecker products

In the following we describe Kronecker products and their basic properties based on the review by Van Loan [2] and our Literature Survey [43]. The Kronecker product between two matrices $\mathbf{B} \in \mathbb{R}^{m\times n}$ and $\mathbf{C} \in \mathbb{R}^{p\times q}$ is defined as:

$$\mathbf{A} = \mathbf{B} \otimes \mathbf{C} = \begin{bmatrix} b_{11}\mathbf{C} & b_{12}\mathbf{C} & \cdots & b_{1n}\mathbf{C} \\ b_{21}\mathbf{C} & b_{22}\mathbf{C} & \cdots & b_{2n}\mathbf{C} \\ \vdots & \vdots & \ddots & \vdots \\ b_{m1}\mathbf{C} & b_{m2}\mathbf{C} & \cdots & b_{mn}\mathbf{C} \end{bmatrix}. \tag{A-1}$$

Essentially, every element in $\mathbf{B}$ is replaced by its multiple with $\mathbf{C}$. The result is thus a matrix of much larger dimension: $\mathbf{A} \in \mathbb{R}^{mp\times nq}$. Basic operations such as transpose and inverse, if it exists, can be efficiently computed using the smaller matrices by the properties:

$$(\mathbf{B} \otimes \mathbf{C})^{\mathrm{T}} = \mathbf{B}^{\mathrm{T}} \otimes \mathbf{C}^{\mathrm{T}}, \tag{A-2a}$$

$$(\mathbf{B} \otimes \mathbf{C})^{-1} = \mathbf{B}^{-1} \otimes \mathbf{C}^{-1}. \tag{A-2b}$$

In case of non-square matrices, the latter relation holds similarly with the pseudo-inverse as well. Distributive and associative properties are given by the relations:

$$\mathbf{D} \otimes (\mathbf{B} + \mathbf{C}) = \mathbf{D} \otimes \mathbf{C} + \mathbf{D} \otimes \mathbf{C}, \qquad (\mathbf{B} \otimes \mathbf{C}) \otimes \mathbf{D} = \mathbf{B} \otimes (\mathbf{C} \otimes \mathbf{D}). \tag{A-3}$$

The terms composing a Kronecker product are not unique in the sense that e.g. $\gamma\mathbf{A}$ can be expressed as:

$$\gamma\mathbf{A} = \gamma(\mathbf{B} \otimes \mathbf{C}) = (\gamma\mathbf{B}) \otimes (\gamma\mathbf{C}). \tag{A-4}$$

which allows trading scalar entries between the Kronecker pairs without changing the result.

Another important property is called the mixed-product property:

$$(\mathbf{B}_1 \otimes \mathbf{C}_1) \cdot (\mathbf{B}_2 \otimes \mathbf{C}_2) = (\mathbf{B}_1 \cdot \mathbf{B}_2) \otimes (\mathbf{C}_1 \cdot \mathbf{C}_2). \tag{A-5}$$

This is especially useful because it allows decompositions of $\mathbf{A}$ to be expressed using those of the smaller matrices $\mathbf{B}$ and $\mathbf{C}$. As an example, for the QR factorizations we have:

$$\mathbf{A} = \mathbf{B} \otimes \mathbf{C} = (\mathbf{Q}_B \cdot \mathbf{R}_B) \otimes (\mathbf{Q}_C \cdot \mathbf{R}_C) = (\mathbf{Q}_B \otimes \mathbf{Q}_C) \cdot (\mathbf{R}_B \otimes \mathbf{R}_C) = \mathbf{Q}_A \cdot \mathbf{R}_A. \tag{A-6}$$

Evaluating inner products can also be accomplished by working with the small matrices, for example:

$$\langle \mathbf{B}_1 \otimes \mathbf{C}_1,\ \mathbf{B}_2 \otimes \mathbf{C}_2 \rangle = \langle \mathbf{B}_1,\ \mathbf{B}_2 \rangle \cdot \langle \mathbf{C}_1,\ \mathbf{C}_2 \rangle . \tag{A-7}$$

Many properties of the composing Kronecker terms are translated and possessed by their Kronecker product as well. This includes non-singularity, symmetry, bandedness, orthogonality, positive definiteness, and so on [3].

Finally, we note a key relation between Kronecker ranks and traditional ranks. The elements of the matrix $\mathbf{A}$ can be suitably rearranged with an operator $\mathcal{R}(\cdot)$ such that we have the following relation:

$$\mathcal{R}(\mathbf{A}) = \text{vec}(\mathbf{B}) \cdot \text{vec}(\mathbf{C})^{\mathrm{T}}. \tag{A-8}$$

This also holds in the summation form:

$$\mathbf{A} = \sum_{j=1}^{M} \mathbf{B}_j \otimes \mathbf{C}_j \qquad \Longrightarrow \qquad \mathcal{R}(\mathbf{A}) = \sum_{j=1}^{M} \text{vec}(\mathbf{B}_j) \cdot \text{vec}(\mathbf{C}_j)^{\mathrm{T}}. \tag{A-9}$$

Kronecker ranks are thus translated into the traditional rank of this rearrangement matrix. Finding the nearest rank K Kronecker product approximation can also be done using this rearrangement in the form

$$\min \left\| \mathcal{R}(\mathbf{A}) - \sum_{j=1}^{K} \text{vec}(\mathbf{B}_j) \cdot \text{vec}(\mathbf{C}_j)^{\mathrm{T}} \right\|_F^2 , \tag{A-10}$$

whose solution is well known and can be obtained from the truncated SVD decomposition of the rearranged matrix by keeping the first K dominant left and right singular vectors and the corresponding singular values.

A-2 Structure exploitation in matrix-vector multiplications

Let us examine how the Kronecker structure can be exploited in the simple matrix vector multiplication

$$\boldsymbol{t} = \mathbf{A}\boldsymbol{x} = (\mathbf{B} \otimes \mathbf{C})\, \boldsymbol{x}, \tag{A-11}$$

where $\mathbf{B}, \mathbf{C} \in \mathbb{R}^{m \times n}$, $\boldsymbol{t} \in \mathbb{R}^{m^2}$, and $\boldsymbol{x} \in \mathbb{R}^{n^2}$. Note that carrying out the multiplication with the full matrix $\mathbf{A} = \mathbf{B} \otimes \mathbf{C}$ would take $2m^2n^2$ operations. In the following, we will look at two formulations which allow a more efficient computation by exploiting the Kronecker structure. In many complicated situations, the only way to exploit Kronecker structures seems to be through iterative methods which rely on matrix-vector multiplications, which motivates our study of this simple but instructive case in more detail.

Small and dense The first method of exploiting the Kronecker structure is to rewrite the operation in matrix form. With $\mathbf{T} \in \mathbb{R}^{m\times m}$ and $\mathbf{X} \in \mathbb{R}^{n\times n}$ defined such that $\boldsymbol{t} = \mathrm{vec}(\mathbf{T})$ and $\boldsymbol{x} = \mathrm{vec}(\mathbf{X})$, the product (A-11) can be rewritten as:

$$\mathbf{T} = \mathbf{C} \cdot \mathbf{X} \cdot \mathbf{B}^{\mathrm{T}}. \tag{A-12}$$

Here we must carry out two smaller matrix-matrix multiplications instead of one large matrix-vector multiplication without exploiting the structure. The total computational cost is $2mn^2 + 2m^2n = 2mn(m+n)$, which is much faster than $2m^2n^2$. The memory requirements are also minimal as it is enough to store the three matrices $\mathbf{B}, \mathbf{C}, \mathbf{X}$, and the result $\mathbf{T}$.

Large and sparse In the second method, we use a different approach and rewrite the Kronecker multiplication (A-11) as:

$$\boldsymbol{t} = (\mathbf{B} \otimes \mathbf{I}_m) \cdot (\mathbf{I}_n \otimes \mathbf{C})\, \boldsymbol{x}. \tag{A-13}$$

The fully dense Kronecker product is now separated into two very sparse matrices, the first one having m^2n, while the second one with mn^2 elements. Carrying out the multiplication takes a total of $2mn^2 + 2m^2n = 2mn(m+n)$ operations, as in the previous case. On the other hand, the memory requirements are much greater as we are storing $\mathbf{B}$ adn $\mathbf{C}$ a total of m and n times in a sparse manner. It is clear that the small and dense case is much more efficient in terms of memory and is thus more suitable for real-time implementations in embedded systems where memory bandwidth has essential limitation on achievable real-time performance [62].

Next, we examine how structures within the Kronecker pairs themselves can be exploited and how this relates to the efficiency of using the Kronecker structure. We will see that as opposed to the dense case, it is now always worthwhile to work with the Kronecker form.

Suppose that $\mathbf{B}$ and $\mathbf{C}$ have structure such that multiplying a vector with them takes $f(m,n)$ operations. For example, if they are dense, $f(m,n) = 2mn$, while for diagonal matrices $f(n) = n$. Furthermore, assume that the full matrix $\mathbf{A}$ has a similar structure (which is often the case by the translation of matrix structures) and the corresponding multiplication is thus $f(m^2,n^2)$. It is evident, that by exploiting the Kronecker structure we change the number of required operations from $f(m^2,n^2)$ to $(n \cdot f(m,n) + m \cdot f(m,n))$. In order to gain efficiency, we must have:

$$n \cdot f(m,n) + m \cdot f(m,n) < f(m^2,n^2). \tag{A-14}$$

Note that for special cases, this does not necessarily hold. If, for example, $\mathbf{B}$ and $\mathbf{C}$ are diagonal matrices with $m = n$ and $f(n) = n$, then we have:

$$n \cdot n + n \cdot n \not< n^2; \tag{A-15}$$

however, there is some gain in terms of memory. In general, the more structure $\mathbf{B}$ and $\mathbf{C}$ has, the less impact exploiting the Kronecker structure will make in computational efficiency. As an additional example, the following table compares costs of multiplications with full and sparse matrices, where the $\mathbf{B}$ and $\mathbf{C}$ terms are assumed to have p elements per column in the latter case.

Table A-1: Computational complexities of dense and sparse matrix-vector multiplications with and without exploiting the Kronecker structure of $\mathbf{A} = \sum_{j=1}^{M} \mathbf{B}_j \otimes \mathbf{C}_j$. In the sparse case, we assume each column of $\mathbf{B}_j$ and $\mathbf{C}_j$ has p nonzero elements, amounting to p^2 for the large matrix $\mathbf{A}$. It is clear that the Kronecker form can quickly become less efficient as the sparsity increases.

Operation	Dense matrix	Full $m \approx n$	Sparse matrix	Sparse $m \approx n$
$\mathbf{A}\boldsymbol{x}$	$2m^2n^2$	$\mathcal{O}(2n^4)$	$2n^2p^2$	$\mathcal{O}(2n^2p^2)$
$\mathbf{C}_j\mathbf{X}\mathbf{B}_j^{\mathrm{T}}$	$2(m^2n + mn^2)$	$\mathcal{O}(4n^3)$	$2n(m+n)p$	$\mathcal{O}(4n^2p)$
$\sum_{j=1}^{M} \mathbf{C}_j\mathbf{X}\mathbf{B}_j^{\mathrm{T}}$	$2M(m^2n + mn^2)$	$\mathcal{O}(4Mn^3)$	$2Mn(m+n)p$	$\mathcal{O}(4Mn^2p)$

Appendix B

ADMM

The following brief overview of alternating direction method of multipliers (ADMM) is based on the excellent discourse by Boyd et al. [29]. For a more detailed overview, please refer to the Literature Survey [43].

We focus on the application of ADMM to a class of optimization problems encountered through the course of this thesis, namely *global variable consensus with regularization.* Such problems take the general form:

$$\min_{\boldsymbol{x}\in\mathbb{R}^n} F(\boldsymbol{x}) = \sum_{i=1}^{N} f_i(\boldsymbol{x}) + g(\boldsymbol{x}); \tag{B-1}$$

terms $f_i(\cdot)$ constitute the decomposable objective function, while $g(\cdot)$ represents a form of regularization. These functions are real-valued and convex; however, we also let them extend their range to include $+\infty$, which allows them to encode indicator functions of constraints by taking on this value in case of violation.

The main idea behind ADMM is to solve the global variable consensus with regularization problem by breaking it into smaller, simpler subproblems. To this end, we introduce the so-called *local variables* $\boldsymbol{x}_i \in \mathbb{R}^n$ and the *global variable* $\boldsymbol{z} \in \mathbb{R}^n$, which allows us to rephrase (B-1) in the following (equivalent) formulation:

$$\begin{aligned} \underset{\boldsymbol{z},\, \boldsymbol{x}_i\in\mathbb{R}^n}{\text{minimize}} \quad & \sum_{i=1}^{N} f_i(\boldsymbol{x}_i) + g(\boldsymbol{z}) \\ \text{s. t.} \quad & \boldsymbol{x}_i - \boldsymbol{z} = \boldsymbol{0}, \quad i = 1, \ldots, N. \end{aligned} \tag{B-2}$$

Conditions for optimality are given by the stationarity of the *augmented Lagrangian*

$$L_\rho(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N, \boldsymbol{z}, \boldsymbol{y}_1, \ldots, \boldsymbol{y}_N) := g(\boldsymbol{z}) + \sum_{i=1}^{N} \left(f_i(\boldsymbol{x}_i) + \boldsymbol{y}_i^{\mathrm{T}}(\boldsymbol{x}_i - \boldsymbol{z}) + \frac{\rho}{2} \|\boldsymbol{x}_i - \boldsymbol{z}\|^2 \right), \tag{B-3}$$

where $\rho > 0$ is the so-called *penalty parameter* and $\boldsymbol{y}_i$ are the dual variables associated with the equality constraints. In ADMM, the optimal primal and dual variables are found in an iterative manner. The so-called *scaled form* of the method is presented below.

B-1 Scaled ADMM updates

It is often more convenient to work with the *scaled dual variables* $\boldsymbol{u}_i = \boldsymbol{y}_i/\rho$, which allow the augmented Lagrangian (B-3) to be expressed as:

$$L_\rho(\boldsymbol{x}_1,\dots,\boldsymbol{x}_N,\boldsymbol{z},\boldsymbol{u}_1,\dots,\boldsymbol{u}_N) = g(\boldsymbol{z}) + \sum_{i=1}^{N}\big(f_i(\boldsymbol{x}_i) + \frac{\rho}{2}\left\|\boldsymbol{x}_i - \boldsymbol{z} + \boldsymbol{u}_i\right\|^2 - \frac{\rho}{2}\left\|\boldsymbol{u}_i\right\|^2\big), \quad \text{(B-4)}$$

In this *scaled form*, the ADMM iterations are given by the following equations:

$$\boldsymbol{x}_i^{(l+1)} := \arg\min_{\boldsymbol{x}_i}\left(f_i(\boldsymbol{x}_i) + \frac{\rho}{2}\left\|\boldsymbol{x}_i - \boldsymbol{z}^{(l)} + \boldsymbol{u}_i^{(l)}\right\|^2\right) \quad \text{(B-5a)}$$

$$\boldsymbol{z}^{(l+1)} := \arg\min_{\boldsymbol{z}}\left(g(\boldsymbol{z}) + \sum_{i=1}^{M}\frac{\rho}{2}\left\|\boldsymbol{z} - \boldsymbol{u}_i^{(l)} - \boldsymbol{x}_i^{(l+1)}\right\|^2\right) \quad \text{(B-5b)}$$

$$\boldsymbol{u}_i^{(l+1)} := \boldsymbol{u}_i^{(l)} + \boldsymbol{x}_i^{(l+1)} - \boldsymbol{z}^{(l+1)}, \quad \text{(B-5c)}$$

which are referred to as the x-, z-, and dual updates. The advantage of the scaled representation is that the term $\|\boldsymbol{u}_i\|^2$ from the augmented Lagrangian does not need to be included in the equations as it does not depend on the variables $\boldsymbol{x}_i$ and $\boldsymbol{z}$.

B-2 Symmetric version

The convergence of ADMM has been widely studied and improved, leading to accelerated methods such as those described in Goldstein et al. [30]. In this thesis, we will use a simple yet effective scheme known as the symmetric version of ADMM to achieve a boost in performance. It is based on the work by Goldfarb et. al [72]: the dual updates are applied also as an intermediate step between the x- and z-updates. The algorithm serves as a baseline for using ADMM throughout this thesis and is summarized below.

Algorithm B.1 (S-ADMM) Symmetric ADMM

```
    Input:
        Problem description $f_i(\cdot)$, $g(\cdot)$, parameter $\rho$
        Initial estimates $\boldsymbol{u}^{(0)}$ and $\boldsymbol{x}_i^{(0)} = \boldsymbol{z}^{(0)}$
 1: for $l = 0, 1, 2, \dots$ do
 2:     $\boldsymbol{x}_i^{(l+1)} := \arg\min_{\boldsymbol{x}_i}\left(f_i(\boldsymbol{x}_i) + \frac{\rho}{2}\left\|\boldsymbol{x}_i - \boldsymbol{z}^{(l)} + \boldsymbol{u}_i^{(l)}\right\|^2\right)$
 3:     $\boldsymbol{u}_i^{(l+1/2)} := \boldsymbol{u}_i^{(l)} + \boldsymbol{x}_i^{(l+1)} - \boldsymbol{z}^{(l)}$
 4:     $\boldsymbol{z}^{(l+1)} := \arg\min_{\boldsymbol{z}}\left(g(\boldsymbol{z}) + \sum_{i=1}^{N}\frac{\rho}{2}\left\|\boldsymbol{z} - \boldsymbol{u}_i^{(l+1/2)} - \boldsymbol{x}_i^{(l+1)}\right\|^2\right)$
 5:     $\boldsymbol{u}_i^{(l+1)} := \boldsymbol{u}_i^{(l+1/2)} + \boldsymbol{x}_i^{(l+1)} - \boldsymbol{z}^{(l+1)}$
 6:     Check stopping criteria.
 7: end for
```

B-3 Stopping criteria

Commonly used stopping criteria for ADMM are based on bounds describing the residuals of the optimality conditions. They are formulated in terms of the *primal* and *scaled dual residuals*, which are defined as

$$\boldsymbol{r}_i^{(l)} := \boldsymbol{x}_i^{(l)} - \boldsymbol{z}^{(l)} \qquad \text{and} \qquad \boldsymbol{d}^{(l)} = \boldsymbol{z}^{(l)} - \boldsymbol{z}^{(l-1)}, \tag{B-6}$$

respectively. In this work, we use the following conditions for terminating the ADMM iterations, as recommended by Boyd et al. [29]:

$$\left\|\boldsymbol{r}_i^{(l)}\right\|_2 \leq \epsilon_i^{\text{pri}} \qquad \text{and} \qquad \left\|\boldsymbol{d}^{(l)}\right\|_2 \leq \epsilon_i^{\text{dual}}, \tag{B-7}$$

where the feasibility tolerances are chosen as a combination of absolute and relative criterion:

$$\epsilon_i^{\text{pri}} = \sqrt{n}\epsilon^{\text{abs}} + \epsilon^{\text{rel}} \max\left\{\left\|\boldsymbol{x}_i^{(l)}\right\|_2, \left\|\boldsymbol{z}^{(l)}\right\|_2\right\}, \tag{B-8a}$$

$$\epsilon_i^{\text{dual}} = \sqrt{n}\epsilon^{\text{abs}} + \epsilon^{\text{rel}} \left\|\boldsymbol{u}_i^{(l)}\right\|_2. \tag{B-8b}$$

Reasonable values are e.g. $\epsilon^{\text{rel}} = 10^{-3}$, while the absolute term ϵ^{abs} depends on the scale of typical variable values. These may also vary depending the required accuracy of the solution.

Appendix C

Iterative solvers

C-1 SQMR

The following summary of the SQMR algorithm for right preconditioning is given based on the presentation by Freund and Nachtigal [39]. It is used to solve symmetric indefinite linear equations of the form

$$\mathbf{A}\boldsymbol{x} = \boldsymbol{b}$$

with $\mathbf{A} = \mathbf{A}^{\mathrm{T}} \in \mathbb{R}^{n\times n}$ and $\boldsymbol{x},\ \boldsymbol{b} \in \mathbb{R}^{n}$ using an arbitrary symmetric preconditioner given as $\mathbf{P} = \mathbf{P}^{\mathrm{T}} \in \mathbb{R}^{n\times n}$. A reasonable criteria for convergence might be $\left\|\boldsymbol{d}^{(l)}\right\|_2 \leq \epsilon^{\mathrm{rel}} \left\|\boldsymbol{x}^{(l)}\right\|_2$.

Algorithm C.1 (SQMR) Iterative solution to symmetric indefinite linear equations.

```
    Input:
        Problem description $\mathbf{A},\ \boldsymbol{b}$; preconditioner $\mathbf{P}$; initial estimate $\boldsymbol{x}_0$
    Initialize:
        $\boldsymbol{r}^{(0)} = \boldsymbol{b} - \mathbf{A}\boldsymbol{x}_0$, $\boldsymbol{t} = \boldsymbol{r}^{(0)}$, $\tau_0 = \|\boldsymbol{t}\|_2$, $\boldsymbol{q}^{(0)} = \mathbf{P}\boldsymbol{t}$
        $\vartheta_0 = 0$, $\rho_0 = \left\langle \boldsymbol{r}^{(0)},\ \boldsymbol{q}^{(0)} \right\rangle$
1:  for $l = 0, 1, 2, \dots$ do
2:      $\boldsymbol{t} := \mathbf{A}\boldsymbol{q}^{(l)}$,    $\sigma_l := \left\langle \boldsymbol{q}^{(l)},\ \boldsymbol{t} \right\rangle$
3:      Stop if $\sigma_l = 0$
4:      $\alpha_l := \rho_l / \sigma_l$,    $\boldsymbol{r}^{(l+1)} := \boldsymbol{r}^{(l)} - \alpha_l \boldsymbol{t}$
5:      $\boldsymbol{t} := \boldsymbol{r}^{(l+1)}$,    $\vartheta_{l+1} := \|\boldsymbol{t}\|_2 / \tau_l$,    $c_{l+1} := 1/\sqrt{1 + \vartheta_{l+1}^2}$,    $\tau_{l+1} := \tau_l \vartheta_{l+1} c_{l+1}$
6:      $\boldsymbol{d}^{(l+1)} := c_{l+1}^2 \vartheta_l^2 \boldsymbol{d}^{(l)} + c_{l+1}^2 \alpha_l^2 \boldsymbol{q}^{(l)}$,    $\boldsymbol{x}^{(l+1)} := \boldsymbol{x}^{(l)} + \boldsymbol{d}^{(l+1)}$
7:      Stop if $\boldsymbol{x}^{(l+1)}$ has converged or $\rho_l = 0$
8:      $\boldsymbol{u}^{(l+1)} := \mathbf{P}\boldsymbol{t}$,  $\rho_{l+1} := \left\langle \boldsymbol{r}^{(l+1)},\ \boldsymbol{u}^{(l+1)} \right\rangle$,  $\beta_{l+1} := \frac{\rho_{l+1}}{\rho_l}$,  $\boldsymbol{q}^{(l+1)} := \boldsymbol{u}^{(l+1)} + \beta_{l+1} \boldsymbol{q}^{(l)}$
9:  end for
```

C-2 MLSQR

Here we present the matrix-free variant of LSQR from the works of Arridge et al. [42]. The algorithm aims to solve least-squares problems of the form:

$$\min_{\boldsymbol{x}} \|\mathbf{A}\boldsymbol{x} - \boldsymbol{b}\|_2$$

with $\mathbf{A} \in \mathbb{R}^{m\times n}$ and $\boldsymbol{x},\ \boldsymbol{b} \in \mathbb{R}^n$ using an arbitrary positive definite preconditioner given as $\mathbf{M} \in \mathbb{R}^{n\times n}$ as an approximation of the inverse of $\mathbf{A}^{\mathrm{T}}\mathbf{A}$. A reasonable criteria for convergence might be based on an absolute tolerance ϵ^{abs} on the gradient norm $\mathbf{A}^{\mathrm{T}}(\boldsymbol{b} - \mathbf{A}\boldsymbol{x})$; at step l, this is available as $\left\|\boldsymbol{g}^{(l)}\right\|_2 = \bar{\phi}_l \alpha_l |c|$, see [41].

Algorithm C.2 (MLSQR) Matrix-free iterative solution to linear least squares problems.

```
    Input:
        Problem description $\mathbf{A},\ \boldsymbol{b}$; preconditioner $\mathbf{M}$; initial estimate $\boldsymbol{x}_0$
    Initialize:
        $\beta_0 = \|\boldsymbol{b}\|_2,\quad \boldsymbol{u}^{(0)} = \boldsymbol{b}/\beta_0,\ \tilde{\boldsymbol{p}}^{(0)} = \mathbf{A}^{\mathrm{T}}\boldsymbol{u}^{(0)},\ \tilde{\boldsymbol{v}}^{(0)} = \mathbf{M}\tilde{\boldsymbol{p}}$
        $\alpha_0 = \sqrt{\left\langle \tilde{\boldsymbol{v}}^{(0)},\ \tilde{\boldsymbol{p}} \right\rangle},\ \tilde{\boldsymbol{p}}^{(0)} = \tilde{\boldsymbol{p}}^{(0)}/\alpha_0,\ \tilde{\boldsymbol{v}}^{(0)} = \tilde{\boldsymbol{v}}^{(0)}/\alpha_0,\ \tilde{\boldsymbol{w}}^{(0)} = \tilde{\boldsymbol{v}}^{(0)}$
        $\boldsymbol{x}^{(0)} = \boldsymbol{x}_0,\ \bar{\phi}_0 = \beta_0,\ \bar{\rho}_0 = \alpha_0$
1:  for $l = 0, 1, 2, \dots$ do
2:      $\boldsymbol{u}^{(l+1)} := \mathbf{A}\tilde{\boldsymbol{v}}^{(l)} - \alpha_l \boldsymbol{u}^{(l)},\quad \beta_{l+1} := \left\|\boldsymbol{u}^{(l+1)}\right\|,\quad \boldsymbol{u}^{(l+1)} := \boldsymbol{u}^{(l+1)}/\beta_{l+1}$
3:      $\tilde{\boldsymbol{p}} := \mathbf{A}^{\mathrm{T}}\boldsymbol{u}^{(l+1)} - \beta_{l+1}\tilde{\boldsymbol{p}},\quad \tilde{\boldsymbol{v}}^{(l+1)} := \mathbf{M}\tilde{\boldsymbol{p}}$
4:      $\alpha_{l+1} := \sqrt{\left\langle \tilde{\boldsymbol{v}}^{(l+1)},\ \tilde{\boldsymbol{p}} \right\rangle},\quad \tilde{\boldsymbol{p}} := \tilde{\boldsymbol{p}}/\alpha_{l+1},\quad \tilde{\boldsymbol{v}}^{(l+1)} := \tilde{\boldsymbol{v}}^{(l+1)}/\alpha_{l+1}$
5:      $\rho := \sqrt{\bar{\rho}_l^2 + \beta_{l+1}^2},\quad c := \bar{\rho}/\rho,\quad s := \beta_{l+1}/\rho$
6:      $\theta := s\alpha_{l+1},\quad \bar{\rho}_{l+1} = -c\alpha_{l+1},\quad \phi := c\bar{\phi}_{l+1},\quad \bar{\phi}_{l+1} := s\bar{\phi}_l$
7:      Check stopping criteria
8:      $\boldsymbol{x}^{(l+1)} := \boldsymbol{x}^{(l)} + \frac{\phi}{\rho}\tilde{\boldsymbol{w}}^{(l)},\quad \tilde{\boldsymbol{w}}^{(l+1)} := \tilde{\boldsymbol{v}}^{(l+1)} - \frac{\theta}{\rho}\tilde{\boldsymbol{w}}^{(l)}$
9:  end for
```

Bibliography

[1] M. Verhaegen, G. Vdovin, and O. Soloviev, “Control for high resolution imaging,” *TU Delft, Delft Center for Systems and Control, lecture notes for SC4045*, April 2016.

[2] C. F. Van Loan, “The ubiquitous Kronecker product,” *Journal of computational and applied mathematics*, vol. 123, no. 1, pp. 85–100, 2000.

[3] C. F. Van Loan and N. Pitsianis, “Approximation with Kronecker products,” in *Linear algebra for large scale and real-time applications*, pp. 293–314, Springer, 1993.

[4] C. T. Fulton and L. Wu, “Parallel algorithms for large least squares problems involving Kronecker products,” *Nonlinear Analysis: Theory, Methods & Applications*, vol. 30, no. 8, pp. 5033–5040, 1997.

[5] A. Barrlund, “Efficient solution of constrained least squares problems with Kronecker product structure,” *SIAM Journal on Matrix Analysis and Applications*, vol. 19, no. 1, pp. 154–160, 1998.

[6] J. M. Bardsley, S. Knepper, and J. Nagy, “Structured linear algebra problems in adaptive optics imaging,” *Advances in Computational Mathematics*, vol. 35, no. 2-4, pp. 103–117, 2011.

[7] M. Rezghi, S. M. Hosseini, and L. Eldén, “Best Kronecker product approximation of the blurring operator in three dimensional image restoration problems,” *SIAM Journal on Matrix Analysis and Applications*, vol. 35, no. 3, pp. 1086–1104, 2014.

[8] K. Greenewald, T. Tsiligkaridis, and A. O. Hero, “Kronecker sum decompositions of space-time data,” in *Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP), 2013 IEEE 5th International Workshop on*, pp. 65–68, IEEE, 2013.

[9] T. Tsiligkaridis and A. O. Hero, “Covariance estimation in high dimensions via Kronecker product expansions,” *IEEE Transactions on Signal Processing*, vol. 61, no. 21, pp. 5347–5360, 2013.

[10] B. Sinquin and M. Verhaegen, "A Kronecker-based model of adaptive optics systems," tech. rep., TU Delft, September 2016.

[11] F. Ding, P. X. Liu, and J. Ding, "Iterative solutions of the generalized Sylvester matrix equations by using the hierarchical identification principle," *Applied Mathematics and Computation*, vol. 197, no. 1, pp. 41–50, 2008.

[12] F. Ding and T. Chen, "Iterative least-squares solutions of coupled Sylvester matrix equations," *Systems & Control Letters*, vol. 54, no. 2, pp. 95–107, 2005.

[13] F. Ding and T. Chen, "On iterative solutions of general coupled matrix equations," *SIAM Journal on Control and Optimization*, vol. 44, no. 6, pp. 2269–2284, 2006.

[14] M. Dehghan and M. Hajarian, "An efficient algorithm for solving general coupled matrix equations and its application," *Mathematical and Computer Modelling*, vol. 51, no. 9, pp. 1118–1134, 2010.

[15] J. Kamm and J. G. Nagy, "Kronecker product and SVD approximations in image restoration," *Linear Algebra and its Applications*, vol. 284, no. 1, pp. 177–192, 1998.

[16] A. N. Langville and W. J. Stewart, "A Kronecker product approximate preconditioner for SANs," *Numerical Linear Algebra with Applications*, vol. 11, no. 8-9, pp. 723–752, 2004.

[17] A. Touzene, "A tensor sum preconditioner for stochastic automata networks," *INFORMS Journal on Computing*, vol. 20, no. 2, pp. 234–242, 2008.

[18] H. Xiang and L. Grigori, "Kronecker product approximation preconditioners for convection–diffusion model problems," *Numerical Linear Algebra with Applications*, vol. 17, no. 4, pp. 691–712, 2010.

[19] E. Ullmann, "A Kronecker product preconditioner for stochastic Galerkin finite element discretizations," *SIAM Journal on Scientific Computing*, vol. 32, no. 2, pp. 923–946, 2010.

[20] E. K. Zander, *Tensor approximation methods for stochastic problems.* Shaker, 2012.

[21] L. Giraldi, A. Nouy, and G. Legrain, "Low-rank approximate inverse for preconditioning tensor-structured linear systems," *SIAM Journal on Scientific Computing*, vol. 36, no. 4, pp. A1850–A1870, 2014.

[22] I. V. Oseledets and S. Dolgov, "Solution of linear systems and matrix inversion in the TT-format," *SIAM Journal on Scientific Computing*, vol. 34, no. 5, pp. A2718–A2739, 2012.

[23] E. D. Denman and A. N. Beavers, "The matrix sign function and computations in systems," *Applied mathematics and Computation*, vol. 2, no. 1, pp. 63–94, 1976.

[24] J. K. Rice, *Efficient Algorithms for Distributed Control: A Structured Matrix Approach.* PhD thesis, Delft University of Technology, Netherlands, 2010.

[25] MATLAB, *version 8.6.0.267246 (R2015b).* Natick, Massachusetts: The MathWorks Inc., 2015.

[26] G. Beylkin and M. J. Mohlenkamp, "Algorithms for numerical analysis in high dimensions," *SIAM Journal on Scientific Computing*, vol. 26, no. 6, pp. 2133–2159, 2005.

[27] B. Sinquin and M. Verhaegen, "Kronecker-based modeling of networks with unknown communication links," *arXiv preprint arXiv:1609.07518*, 2016.

[28] G. H. Golub and C. F. Van Loan, *Matrix computations*, vol. 3. JHU Press, 2012.

[29] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends® in Machine Learning*, vol. 3, no. 1, pp. 1–122, 2011.

[30] T. Goldstein, B. O'Donoghue, S. Setzer, and R. Baraniuk, "Fast alternating direction optimization methods," *SIAM Journal on Imaging Sciences*, vol. 7, no. 3, pp. 1588–1623, 2014.

[31] R. H. Bartels and G. W. Stewart, "Solution of the matrix equation AX+XB=C [F4] (Algorithm 432).," *Commun. ACM*, vol. 15, no. 9, pp. 820–826, 1972.

[32] R. Martin, G. Peters, and J. Wilkinson, "The QR algorithm for real Hessenberg matrices," in *Linear algebra*, pp. 359–371, Springer, 1971.

[33] L. Xie, J. Ding, and F. Ding, "Gradient based iterative solutions for general linear matrix equations," *Computers & Mathematics with Applications*, vol. 58, no. 7, pp. 1441–1448, 2009.

[34] Y. Nesterov, "A method of solving a convex programming problem with convergence rate $\mathcal{O}(1/k^2)$," *Soviet Mathematics Doklady*, vol. 27, no. 2, pp. 372–376, 1983.

[35] A. P. Liavas, G. Kostoulas, G. Lourakis, K. Huang, and N. D. Sidiropoulos, "Nesterov-based alternating optimization for nonnegative tensor factorization: algorithm and parallel implementations," submitted to the IEEE Transactions on Signal Processing.

[36] E. Tyrtyshnikov, "Tensor ranks for the inversion of tensor-product binomials," *Journal of computational and applied mathematics*, vol. 234, no. 11, pp. 3170–3174, 2010.

[37] B. Sinquin, "Sign iterations with Kronecker modeling," tech. rep., TU Delft, May 2017.

[38] R. Barrett, M. W. Berry, T. F. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, and H. Van der Vorst, *Templates for the solution of linear systems: building blocks for iterative methods*, vol. 43. Siam, 1994.

[39] R. W. Freund and N. M. Nachtigal, "A new Krylov-subspace method for symmetric indefinite linear systems," in *Proceedings of the 14th IMACS World Congress on Computational and Applied Mathematics*, vol. 1253, 1994.

[40] J. H. Johansson and A. Hansson, "A tailored inexact interior-point method for systems analysis," in *Decision and Control, 2008. CDC 2008. 47th IEEE Conference on*, pp. 3071–3076, IEEE, 2008.

[41] C. C. Paige and M. A. Saunders, "LSQR: An algorithm for sparse linear equations and sparse least squares," *ACM transactions on mathematical software*, vol. 8, no. 1, pp. 43–71, 1982.

[42] S. R. Arridge, M. M. Betcke, and L. Harhanen, "A priorconditioned LSQR algorithm for linear ill-posed problems with edge-preserving regularization," *arXiv preprint arXiv:1308.6634*, 2013.

[43] P. Varnai, "Exploiting Kronecker product structures in the constrained real-time control of large-scale adaptive optics systems," tech. rep., TU Delft, 2016.

[44] R. Gilmozzi and J. Spyromilio, "The European Extremely Large Telescope (E-ELT)," *The Messenger*, vol. 127, no. 11, p. 3, 2007.

[45] C. Kulcsár, H.-F. Raynaud, C. Petit, and J.-M. Conan, "Minimum variance prediction and control for adaptive optics," *automatica*, vol. 48, no. 9, pp. 1939–1954, 2012.

[46] M. Konnik, *Online constrained receding horizon control for astronomical adaptive optics.* PhD thesis, Ph. D. dissertation, School of Electrical Engineering and Computer Science, the University of Newcastle, Australia, 2013.

[47] M. Verhaegen and V. Verdult, *Filtering and system identification: a least squares approach.* Cambridge university press, 2007.

[48] C. Kulcsár, H.-F. Raynaud, C. Petit, and J.-M. Conan, "Minimum variance control in presence of actuator saturation in adaptive optics," in *SPIE Astronomical Telescopes+ Instrumentation*, pp. 70151G–70151G, International Society for Optics and Photonics, 2008.

[49] D. Gavel and D. Wiberg, "Toward Strehl-optimizing adaptive optics controllers," tech. rep., Lawrence Livermore National Lab., CA (US), 2002.

[50] M. Konnik and J. De Doná, "Hot-start efficiency of quadratic programming algorithms for fast model predictive control: A comparison via an adaptive optics case study," in *Control Conference (AUCC), 2014 4th Australian*, pp. 95–100, IEEE, 2014.

[51] M. Grant, S. Boyd, and Y. Ye, "Cvx: Matlab software for disciplined convex programming," 2008.

[52] Y.-H. Dai and R. Fletcher, "Projected Barzilai-Borwein methods for large-scale box-constrained quadratic programming," *Numerische Mathematik*, vol. 100, no. 1, pp. 21–47, 2005.

[53] J. Barzilai and J. M. Borwein, "Two-point step size gradient methods," *IMA journal of numerical analysis*, vol. 8, no. 1, pp. 141–148, 1988.

[54] M. Raydan, "On the Barzilai and Borwein choice of steplength for the gradient method," *IMA Journal of Numerical Analysis*, vol. 13, no. 3, pp. 321–326, 1993.

[55] J. J. Moré and G. Toraldo, "On the solution of large quadratic programming problems with bound constraints," *SIAM Journal on Optimization*, vol. 1, no. 1, pp. 93–113, 1991.

[56] A. Björck, *Numerical methods for least squares problems.* Siam, 1996.

[57] J. Nocedal and S. Wright, *Numerical optimization.* Springer Science & Business Media, 2006.

[58] M. Benzi, G. H. Golub, and J. Liesen, “Numerical solution of saddle point problems,” *Acta numerica*, vol. 14, pp. 1–137, 2005.

[59] E. L. S. Wong, *Active-set methods for quadratic programming.* PhD thesis, University of California, San Diego, 2011.

[60] C. M. Maes, *A regularized active-set method for sparse convex quadratic programming.* PhD thesis, Citeseer, 2010.

[61] L. Lukšan and J. Vlček, “Indefinitely preconditioned inexact Newton method for large sparse equality constrained nonlinear programming problems,” *Numerical linear algebra with applications*, vol. 5, no. 3, pp. 219–247, 1998.

[62] C. Vuik and C. Lemmens, “Programming on the GPU with CUDA (version 6.5),” *TU Delft, course material*, 2015.

[63] H. Subedi, N. T. Zimmerman, N. J. Kasdin, K. Cavanagh, and A. E. Riggs, “Coronagraph-integrated wavefront sensing with a sparse aperture mask,” *Journal of Astronomical Telescopes, Instruments, and Systems*, vol. 1, no. 3, pp. 039001–039001, 2015.

[64] E. E. Bloemhof and J. K. Wallace, “Phase contrast techniques for wavefront sensing and calibration in adaptive optics,” 2003.

[65] H. Subedi, P. Varnai, and J. Kasdin, “Low-order wavefront sensing for space-based coronagraphic missions,” in preparation, 2017.

[66] J. Nakamura, *Image sensors and signal processing for digital still cameras.* CRC press, 2016.

[67] R. J. Vanderbei, “Fast Fourier optimization,” *Mathematical Programming Computation*, vol. 4, no. 1, pp. 53–69, 2012.

[68] Z. Lu and T. K. Pong, “Minimizing condition number via convex programming,” *SIAM Journal on Matrix Analysis and Applications*, vol. 32, no. 4, pp. 1193–1211, 2011.

[69] P. Maréchal and J. J. Ye, “Optimizing condition numbers,” *SIAM Journal on Optimization*, vol. 20, no. 2, pp. 935–947, 2009.

[70] X. Chen, R. S. Womersley, and J. J. Ye, “Minimizing the condition number of a Gram matrix,” *SIAM Journal on optimization*, vol. 21, no. 1, pp. 127–148, 2011.

[71] H. Subedi, P. Varnai, and J. Kasdin, “Low-order wavefront sensing for space-based coronagraphic missions,” 2017.

[72] D. Goldfarb, S. Ma, and K. Scheinberg, “Fast alternating linearization methods for minimizing the sum of two convex functions,” *Mathematical Programming*, vol. 141, no. 1-2, pp. 349–382, 2013.

Glossary

List of Acronyms

ALS	alternating least squares
SAM	sparse aperture mask
AO	adaptive optics
DM	deformable mirror
WFS	wavefront sensor
SH	Shack-Hartmann
VARX	vector autoregressive with exogenous inputs
ADMM	alternating direction method of multipliers
AMA	alternating minimization algorithm
AS	active sets
IP	interior point
PG	projected gradient
PABB	projected alternating Barzilai-Borwein
MVC	minimum variance control
E-ELT	European Extremely Large Telescope
KKT	Karush-Kuhn-Tucker
SQMR	symmetric quasi-minimum residual
MLSQR	matrix-free LSQR
GPU	graphics processing unit
SP	shaped pupil
FPM	focal plane mask