Recovering structured low-rank operators using nuclear norms

Thesis by
John J. Bruer

In Partial Fulfillment of the Requirements for the
degree of
Doctor of Philosophy

CALIFORNIA INSTITUTE OF TECHNOLOGY
Pasadena, California

2017
Defended January 17, 2017

John J. Bruer
ORCID: 0000-0003-4590-3038

Acknowledgements

First, I need to thank my advisor Joel Tropp for the mentorship and support he has given me throughout my time at Caltech. It has been a true pleasure to learn from someone who has so much passion for both research and teaching. I greatly admire Joel's commitment to his work and the high standards he maintains.

I also appreciate the guidance Adam Wierman and John Doyle gave me early in my graduate career. I am grateful that Adam also agreed to serve on my thesis committee along with Babak Hassibi and Yisong Yue. Thank you all.

The administrative staff at Caltech is superlative. Many thanks to Jeri Chittum, Sydney Garstang, Maria Lopez, Carmen Nemer–Sirois, and Sheila Shull for all their efforts; their help was invaluable.

I have had the pleasure of working at Caltech alongside enthusiastic and friendly colleagues. These include my officemates Hyoung Jun Ahn, Henry Jacobs, Zhenua Liu, John Pang, and Xiaoqi Ren. I additionally want to recognize Brendan Ames, Richard Chen, Brendan Farrell, Alex Gittens, Mike McCoy, and Madeleine Udell from Joel's research group. Thank you for the stimulating conversations about research and your camaraderie. I especially want to thank Mike and his wife Anya Demianenko for the restaurant tours and beer festivals in addition to their friendship.

Trevor Fowler is a wonderful friend, and I throughly enjoyed the time we spent as roommates in Pasadena. I had always assumed that I would learn a little something about math at Caltech, but I didn't realize that I would learn how to brew beer and bet on horse racing as well. Thank you for making my time outside of Caltech that much more interesting and fun.

To Trevor, Andrew Payne, and Steve Trzesniewski: even though the general fund is no more, I will never forget pinhead and green shirt. Thank you to Mark

Giacomantonio and Yunia Lubega for all the bar crawls, brunches, and bad TV.

I am thankful for the close friendships that have endured despite my decision to pursue graduate school on the other side of the country. A special thank you to Phil Hennessey for his frequent trips out to LA and to his futon for weekends in New York. May there be many more adventures in the future.

Above all, thank you to my family. My grandfather helped foster my interest in math when I was young, and I think he has been looking forward to the completion of this dissertation as much as I have. My parents and brother have given me all the love and encouragement I could ask for. None of this would have been possible without their unwavering support.

Finally, thank you to Judy for your love and companionship. I am extremely fortunate to have you in my life, and I look forward to sharing whatever is to come.

Abstract

This work considers the problem of recovering matrices and operators from limited and/or noisy observations. Whereas matrices result from summing tensor products of vectors, operators result from summing tensor products of matrices. These constructions lead to viewing both matrices and operators as the sum of "simple" rank-1 factors.

A popular line of work in this direction is low-rank matrix recovery, i.e., using linear measurements of a matrix to reconstruct it as the sum of few rank-1 factors. Rank minimization problems are hard in general, and a popular approach to avoid them is convex relaxation. Using the trace norm as a surrogate for rank, the low-rank matrix recovery problem becomes convex.

While the trace norm has received much attention in the literature, other convexifications are possible. This thesis focuses on the class of nuclear norms—a class that includes the trace norm itself. Much as the trace norm is a convex surrogate for the matrix rank, other nuclear norms provide convex complexity measures for additional matrix structure. Namely, nuclear norms measure the structure of the factors used to construct the matrix.

Transitioning to the operator framework allows for novel uses of nuclear norms in recovering these structured matrices. In particular, this thesis shows how to lift structured matrix factorization problems to rank-1 operator recovery problems. This new viewpoint allows nuclear norms to measure richer types of structures present in matrix factorizations.

This work also includes a Python software package to model and solve structured operator recovery problems. Systematic numerical experiments in operator denoising demonstrate the effectiveness of nuclear norms in recovering structured operators. In particular, choosing a specific nuclear norm that corresponds to the underlying factor structure of the operator improves the performance

of the recovery procedures when compared, for instance, to the trace norm. Applications in hyperspectral imaging and self-calibration demonstrate the additional flexibility gained by utilizing operator (as opposed to matrix) factorization models.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

Chapter 1

Introduction

Matrices provide natural representations for data in many practical settings. Often, however, we only have partial or noisy observations of the true matrix underlying a data model. A central question in these cases is how to effectively recover this matrix from limited information. This endeavor requires the reconciliation of two competing interests: we prefer simple models to complex ones, and our models should agree with our observations. In other words, we seek the simplest explanation for what we see.

Matrices have a well-known complexity measure: *rank*. A low-rank matrix can be written as the sum of a small number of "simple" (i.e., rank-1) matrices. And while matrices of practical interest are typically low-rank—or at least approximately low-rank—optimization problems involving rank are usually computationally intractable.

Surrogate complexity measures, like the well-known trace norm [Faz02; RFP10], provide computationally efficient ways to promote low-rank solutions in optimization problems. But matrices may possess other types of structure besides low-rankedness. In particular, we consider factorization models where a matrix $\boldsymbol{A}$ may be written as $\boldsymbol{A} = \boldsymbol{X}\boldsymbol{Y}^{\mathsf{t}}$ such that the factors $\boldsymbol{X}$ and $\boldsymbol{Y}$ themselves have meaningful structure.

This thesis focuses on factorization models for data and a family of convex complexity measures called *nuclear norms* (of which the trace norm is but one example). These norms have a long history in functional analysis [Sch50; Gro53; Jam87; Rya02], and we can use them to promote *structured* low-rank solutions in matrix recovery problems. Instead of considering a matrix as simply the sum of

rank-1 matrices, we can consider it the sum of rank-1 matrices that themselves have special properties. In the factorization model, this corresponds to finding $\boldsymbol{A} = \boldsymbol{X}\boldsymbol{Y}^{\mathsf{t}}$ such that the columns of $\boldsymbol{X}$ and $\boldsymbol{Y}$ adhere to some particular structure.

A major contribution of this thesis is the extension of the structured matrix recovery problem to a class of objects we call *operators*. Whereas a rank-1 matrix is the tensor product of two vectors, a rank-1 operator is the tensor product of two *matrices*. These operators allow us to consider data with more than two dimensions, but they also have a connection with the matrix factorization model. Namely, the matrix $\boldsymbol{A} = \boldsymbol{X}\boldsymbol{Y}^{\mathsf{t}}$ can be viewed as the *action* of the operator $\boldsymbol{X} \otimes \boldsymbol{Y}$ on an identity matrix. Therefore, the problem of recovering a structured matrix may be lifted to the problem of recovering a rank-1 operator from an observation of its action. By moving to the operator setting, we can use nuclear norms to recover a larger variety of structured factorizations. We focus on demonstrating the numerical efficacy of nuclear norms in recovering structured operators using a Python package, `operfact`, that we developed.

This thesis is based on joint research with my advisor Joel Tropp and stems from his unpublished work on nuclear norms [Tro12]. In this chapter, we summarize our general approach to structured matrix recovery (Section 1.1), review the concept of nuclear norms (Section 1.2), and introduce our extension of this nuclear norm framework to operators (Section 1.3). We outline the remainder of the thesis in Section 1.4 and briefly discuss my other completed research project in Section 1.5.

1.1 An overview of the problem

Consider an $m \times n$ real matrix $\boldsymbol{A} \in \mathbb{M}^{m \times n}$, and assume that we observe the linear measurements

$$\boldsymbol{b} = \boldsymbol{\mu}(\boldsymbol{A}),$$

where $\boldsymbol{\mu} \colon \mathbb{M}^{m \times n} \to \mathbb{R}^p$ is a linear operator.

Given the measurements $\boldsymbol{b}$, we want to know:

- When can we find a factorization $\boldsymbol{A} = \boldsymbol{X}\boldsymbol{Y}^{\mathsf{t}}$ such that the factors $\boldsymbol{X}$ and $\boldsymbol{Y}$ have particular structure?
- When can we approximate $\boldsymbol{A}$ itself under the assumption that a factorization $\boldsymbol{A} = \boldsymbol{X}\boldsymbol{Y}^{\mathsf{t}}$ exists where the factors have particular structure?

The goal of this work is a numerical study of these problems. We develop a software package (`operfact`, see Chapter 4) to model structured factorization problems, and we employ it to demonstrate how a class of convex complexity measures called *nuclear norms* aid in their solution.

Finding a structured factorization is an interesting (and challenging problem) even when we have access to every entry of the matrix $\boldsymbol{A}$. In these cases, the factors themselves may have a useful interpretation, and we review several such models in Section 2.1.1. Uncovering the appropriate factorization may reveal an interesting underlying explanation for our data. Chapter 7 describes a problem in self-calibration where recovery of the factorization equates to recovery of signals sent over a channel with uncertain parameters.

Other times, however, we may seek to approximate $\boldsymbol{A}$ itself from limited and/or noisy measurements $\boldsymbol{b}$. In Section 1.1.1 we discuss several types of linear measurements that correspond to practical data acquisition situations. If we expect that $\boldsymbol{A}$ has a factorization $\boldsymbol{A} = \boldsymbol{X}\boldsymbol{Y}^{\mathsf{t}}$ where the factors $\boldsymbol{X}$ and $\boldsymbol{Y}$ have particular structure, we should use this information to inform our approximation procedure. Indeed, this knowledge is critical if we wish to accurately approximate $\boldsymbol{A}$ from partial observations. We give examples of such structure in Section 1.1.2. Chapters 5 and 6 show the benefit of incorporating prior beliefs on factor structure into denoising problems.

Note that approximating $\boldsymbol{A}$ does not necessarily require the recovery of a structured factorization but instead merely relies on the fact that such a factorization exists. This is a subtle yet important distinction.

In order to solve these recovery problems, we rely on the concept of regularization (Section 1.1.3). Our approach, however, does face some algorithmic challenges, and we outline those in Section 1.1.4.

1.1.1 Linear measurement models

In this work we consider linear measurement models. That is, given the matrix $\boldsymbol{A} \in \mathbb{M}^{m\times n}$, we observe a vector of measurements $\boldsymbol{b} \in \mathbb{R}^p$ with the ith entry

$$b_i = \langle \boldsymbol{M}_i, \boldsymbol{A} \rangle + z_i,$$

where $\boldsymbol{M}_i \in \mathbb{M}^{m\times n}$ is a known matrix and $z_i \in \mathbb{R}$ is additive noise.

It is convenient to write

$$\boldsymbol{b} = \boldsymbol{\mu}(\boldsymbol{A}) + \boldsymbol{z},$$

where $\boldsymbol{\mu} : \mathbb{M}^{m \times n} \to \mathbb{R}^p$ is the linear *measurement map* induced by the $\boldsymbol{M}_i$ and $\boldsymbol{z} \in \mathbb{R}^p$ is the additive noise.

While it may seem limiting to restrict ourselves to linear measurements, this model covers many data acquisition scenarios of practical interest.

Denoising. In the case where the map returns a vectorized version of the original matrix, i.e., $\boldsymbol{\mu} : \boldsymbol{A} \mapsto \operatorname{vec} \boldsymbol{A}$, the vector $\boldsymbol{b}$ is simply all entries of the matrix corrupted by additive noise. Given these measurements, a natural task is to recover the original matrix $\boldsymbol{A}$. This problem is called *denoising*, and we will consider it in more detail in Chapters 5 and 6.

Missing entries. Let $\boldsymbol{\mu}$ be the operator that returns a particular subset of p entries from a matrix.

This situation leads to the *matrix completion* problem where we wish to recover the original matrix by filling in the unobserved entries. It appears in applications such as collaborative filtering [SRJ05; KBV09] where a system makes predictions about users' preferences while having observed only a small subset of all possible preferences.

Compressed sensing. In *compressed sensing* [CRT06a; Don06] we attempt to recover a signal from few random measurements. That is, we take p random linear observations of the low-rank matrix $\boldsymbol{A} \in \mathbb{M}^{m \times n}$ where $p < mn$. At first blush this may appear impossible. If, however, we take the number p of measurements large enough, we can succeed with high probability [RFP10; CRPW12; ALMT14].

Phase retrieval et al. Lifting procedures [GW95; Nes98; BN01] can transform quadratic or bilinear vector measurements into linear measurements of rank-1 matrices. This technique has found applications in signal processing including phase retrieval [BBCE09; CMP11; CESV13], blind deconvolution [ARR14], and self-calibration [LS15b]. We use nuclear norms to perform self-calibration in Chapter 7.

Structure from action. We can also define the map $\boldsymbol{\mu}$ as the action of the matrix on a fixed vector. That is, $\boldsymbol{\mu} : \boldsymbol{A} \mapsto \boldsymbol{A}\boldsymbol{x}$, where $\boldsymbol{x} \in \mathbb{R}^n$ determines $\boldsymbol{\mu}$. This

occurs, for instance, in randomized linear algebra [HMT11] where the goal is to approximate low-rank matrices from their actions on vectors.

Given any of these linear measurements, the goal is to either factor or recover the partially observed matrix.

1.1.2 Factor structure

It is clear, however, that without additional assumptions on the underlying structure of the matrix we will not have enough information to do so. Indeed, consider the example of missing entries. Many matrices of a given size may agree on a subset of their entries. We apparently have too little information to reconstruct the original matrix.

The key realization is that an $m \times n$ matrix does not necessarily contain mn independent entries. In practice, matrices have structure, and this structure implies that these matrices contain limited information. Again, we consider matrices that admit factorizations

$$\boldsymbol{A} = \boldsymbol{X}\boldsymbol{Y}^{\mathsf{t}},$$

where the factors $\boldsymbol{X}$ and $\boldsymbol{Y}$ are themselves structured.

Examples of possible types of factor structure include

- **Sparsity:** The factor has few nonzero entries.
- **Repeated structure:** The factor is the linear image of a vector, such as a Toeplitz matrix.
- **Set membership:** The factor belongs to a convex set, such as the set of matrices with appropriate size and nonnegative entries.
- **Low complexity:** The factor has low complexity as measured by a convex function, such as a norm.

Additionally we often find that matrices have low rank. That is, a factorization $\boldsymbol{A} = \boldsymbol{X}\boldsymbol{Y}^{\mathsf{t}}$ exists where the inner dimension of the decomposition is small. While the rank of the matrix $\boldsymbol{A}$ is somewhat independent of factor structure, a low-rank assumption on $\boldsymbol{A}$ additionally serves to limit the amount of information contained in the matrix.

In this work we focus on factorizations $\boldsymbol{A} = \boldsymbol{X}\boldsymbol{Y}^{\mathsf{t}}$ where the columns of the factors $\boldsymbol{X}$ and $\boldsymbol{Y}$ have low complexity as measured by convex functions. In particular, we consider the cases where we measure this complexity using norms. This leads to the *nuclear norm framework* we introduce in Section 1.2. Chapter 3 covers the mathematical background of these norms in greater detail, while Section 2.3 discusses their historical development.

1.1.3 Regularization

We wish to use any prior knowledge regarding the factor structure of a matrix in order to recover it from incomplete observations. This goal creates a tension between remaining faithful to the measurements while seeking simpler (i.e., more structured) solutions. The key concept that enables us to balance these interests is *regularization*.

Consider a convex function f that assigns low values to matrices that have decompositions composed of "simple" factors. To recover a structured matrix from linear measurements $\boldsymbol{b}$, we can then attempt to solve one of the following three equivalent problems:

$$\begin{aligned} &\underset{\boldsymbol{A}}{\text{minimize}} \quad \text{loss}(\boldsymbol{A}; \boldsymbol{b}) \quad \text{subject to} \quad f(\boldsymbol{A}) \leq \gamma \\ &\underset{\boldsymbol{A}}{\text{minimize}} \quad f(\boldsymbol{A}) \quad \text{subject to} \quad \text{loss}(\boldsymbol{A}; \boldsymbol{b}) \leq \epsilon \\ &\underset{\boldsymbol{A}}{\text{minimize}} \quad \text{loss}(\boldsymbol{A}; \boldsymbol{b}) + \lambda f(\boldsymbol{A}), \end{aligned} \tag{1.1}$$

where $\text{loss}(\cdot; \boldsymbol{b})$ is a convex function measuring the agreement between the candidate matrix (under the linear measurement map) and the observations $\boldsymbol{b}$.

These problems illustrate our competing interests in maintaining fidelity to our observations while seeking simple solutions. The parameters γ, ϵ, and λ all serve to tune this balance. The complexity measurement f serves as a *regularizer*: it pushes us to find more regular, i.e., structured solutions.

1.1.4 Algorithmic challenges

Given our assumptions on the regularizer f and the loss function, the regularized recovery problems (1.1) are convex. In fact, they are regularized linear inverse problems owing to the fact that our measurement maps are linear. But even if we assume that we can construct appropriate convex regularizers f on the factor structure, we face significant challenges in solving these problems.

1.1.4.1 Convex, but intractable f

The convex approach in (1.1) seems simple and convenient, but our discussion of nuclear norms will show that expressing convex regularizers f for factorizations is not straightforward. Often there is no closed form for f, but even worse, we have examples of nuclear norms that are provably difficult to compute exactly.

Later we discuss two approaches for handling these situations. Our most versatile method relies on nonconvex alternating minimization (Section 4.2), while the other more narrowly applicable method finds semidefinite relaxations for some interesting cases (Section 3.6). While the shift to nonconvex methods introduces additional complexities, we take some solace in the fact that a truly convex problem underlies our efforts.

1.1.4.2 Factorization

If we seek a structured matrix factorization, the approach in (1.1) requires an additional step. After retrieving the approximation $\boldsymbol{A}$ we must then apply a factorization procedure. In a few cases where f is a nuclear norm, we will see that an appropriate factorization follows with almost no effort. This is the exception, however.

Generally, finding this factorization takes the form

$$\text{Find} \quad (\boldsymbol{X}, \boldsymbol{Y}) \quad \text{subject to} \quad \boldsymbol{X}\boldsymbol{Y}^{\mathsf{t}} = \boldsymbol{A} \quad \text{and} \quad (\boldsymbol{X}, \boldsymbol{Y}) \in \mathcal{C},$$

where $\mathcal{C}$ is the set of permissible structured factors. This *bilinear* inverse problem is nonconvex and presents optimization challenges.

1.1.4.3 Rank constraints

Rank is an important complexity measure for matrices, and we can define in it terms of a decomposition.

Definition 1.1.1 (Rank)**.** The rank of a matrix $\boldsymbol{A} \in \mathbb{M}^{m\times n}$ is the smallest inner dimension over all possible decompositions. That is,

$$\operatorname{rank}(\boldsymbol{A}) := \min \left\{ r : \boldsymbol{A} = \boldsymbol{X}\boldsymbol{Y}^{t}, \boldsymbol{X} \in \mathbb{M}^{m\times r}, \boldsymbol{Y} \in \mathbb{M}^{n\times r} \right\}.$$

Low-rank matrices admit decompositions with small inner dimension, and this dimension reduction leads to simple models. Furthermore, practical matrices

are often low-rank or approximately low-rank. In fact, we may believe that the underlying model of the data is truly low-rank, and any real instantiation of that model is noisy. Seeking low-rank models is therefore justified.

If, however, we incorporate a low-rank assumption into our algorithmic approach, we face a problem. Indeed, consider the rank minimization problem

$$\underset{\boldsymbol{A}}{\text{minimize}} \quad \operatorname{rank}(\boldsymbol{A}) \quad \text{subject to} \quad \operatorname{loss}(\boldsymbol{A}; \boldsymbol{b}) \leq \epsilon.$$

The rank constraint is *not* convex, and in general, this problem is difficult to optimize.

In her doctoral thesis, Fazel [Faz02] proposed replacing the difficult rank constraint with the trace norm. That is,

$$\underset{\boldsymbol{A}}{\text{minimize}} \quad \|\boldsymbol{A}\|_{S_1} \quad \text{subject to} \quad \operatorname{loss}(\boldsymbol{A}; \boldsymbol{b}) \leq \epsilon,$$

where the trace norm $\|\boldsymbol{A}\|_{S_1}$, also known as the Schatten 1-norm, sums the singular values of $\boldsymbol{A}$. By analogy with the ℓ_1 norm—a popular heuristic for promoting sparsity [DS89]—the trace norm promotes matrices with few nonzero singular values. Since the rank of a matrix corresponds exactly to its number of nonzero singular values, this approach proves effective in finding low-rank solutions.

As a norm, it is necessarily convex, and in particular it may be computed through a semidefinite program. Furthermore, this norm belongs to the class of nuclear norms[1] that we introduce momentarily. We will see that nuclear norms measure the complexity of matrices through the size and structure of their possible decompositions. Therefore we can use such norms to promote low-rank, as well as structured, solutions.

1.2 The nuclear norm framework

In this section we define *nuclear norms* and give some intuition for their behavior; we give a more complete development in Chapter 3. But first we introduce tensor notation for matrix decompositions.

1.2.1 Dyads

The discussion of nuclear norms is closely tied to matrix decompositions, and so we now introduce notation to assist us in working with such decompositions.

[1] In fact, the trace norm is often called "*the* nuclear norm".

Definition 1.2.1 (Dyad)**.** Given vectors $\boldsymbol{x} \in \mathbb{R}^m$ and $\boldsymbol{y} \in \mathbb{R}^n$, we define the *dyad* $\boldsymbol{x} \otimes \boldsymbol{y}$ to be the rank-1 matrix

$$\boldsymbol{x} \otimes \boldsymbol{y} := \boldsymbol{x}\boldsymbol{y}^{\mathsf{t}} = \begin{bmatrix} x_1 y_1 & \cdots & \cdots & x_1 y_n \\ x_2 y_1 & x_2 y_2 & \cdots & x_2 y_n \\ \vdots & \vdots & \ddots & \vdots \\ x_m y_1 & \cdots & \cdots & x_m y_n \end{bmatrix} \in \mathbb{M}^{m \times n}. \tag{1.2}$$

For any matrix decomposition $\boldsymbol{A} = \boldsymbol{X}\boldsymbol{Y}^{\mathsf{t}}$ with inner dimension r, we can write

$$\boldsymbol{A} = \sum_{i=1}^{r} \boldsymbol{x}_i \otimes \boldsymbol{y}_i,$$

where the $\boldsymbol{x}_i$ and $\boldsymbol{y}_i$ are the columns of $\boldsymbol{X}$ and $\boldsymbol{Y}$. Just as the matrix decomposition $\boldsymbol{A} = \boldsymbol{X}\boldsymbol{Y}^{\mathsf{t}}$ is not unique, neither is the dyadic decomposition.

We then have the following definition for rank.

Definition 1.2.2 (Rank)**.** The rank of a matrix is the minimal number of dyads required among all of its dyadic decompositions. That is,

$$\operatorname{rank}(\boldsymbol{A}) := \min \left\{ r : \boldsymbol{A} = \sum_{i=1}^{r} \boldsymbol{x}_i \otimes \boldsymbol{y}_i \right\}.$$

This corresponds exactly to the notion that a rank-r matrix can be written as the sum of r rank-1 matrices.

At first glance it appears that we have done nothing but introduce unusual notation for writing matrices and matrix factorizations. The benefits of this approach should become clearer as we continue, but for now we highlight the main advantages.

First, the notation itself suggests that the vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ have equal standing in the dyad $\boldsymbol{x} \otimes \boldsymbol{y}$. We consider their structures separately but consider their contributions to the dyad equally. Second, it provides a clean notation for writing factorizations that dispenses with using the matrix transpose. Third, the symbol $\otimes$ suggests the connection between this work and the theory of tensor products. (We discuss this briefly in Section 1.3.3 and survey some of the relevant historical developments in Section 2.3.) Finally, it allows for us to easily extend the work on nuclear norms in matrix factorizations to the operator setting. We introduce this extension in Section 1.3, and it is a major contribution of this thesis.

1.2.2 The nuclear norm

Say that we have normed spaces $X = (\mathbb{R}^m, \|\cdot\|_X)$ and $Y = (\mathbb{R}^n, \|\cdot\|_Y)$. We can now consider norms on $X \otimes Y$, and we insist that any such norm $|||\cdot|||$ satisfies the property

$$|||\boldsymbol{x} \otimes \boldsymbol{y}||| = \|\boldsymbol{x}\|_X \|\boldsymbol{y}\|_Y, \tag{1.3}$$

for all dyads $\boldsymbol{x} \otimes \boldsymbol{y}$ with $\boldsymbol{x} \in X$ and $\boldsymbol{y} \in Y$. This type of norm is called a *crossnorm*, and we discuss it more formally in Section 3.3.1. The crossnorm property (1.3) ensures that each factor of the dyad contributes to the norm symmetrically. We can see that scaling the dyad also scales the crossnorm, as expected.

For now, we want to consider how we can extend the crossnorm (1.3) to sums of dyads. This choice is not unique, but any such choice must obey the triangle inequality

$$\left|\left|\left| \sum_{i=1}^{r} \boldsymbol{x}_i \otimes \boldsymbol{y}_i \right|\right|\right| \leq \sum_{i=1}^{r} \|\boldsymbol{x}_i\|_X \|\boldsymbol{y}_i\|_Y. \tag{1.4}$$

We desire the largest such norm in order to maximally penalize deviation from structure, and so it is sensible to use the bound (1.4) to define the norm. Note, however, that the norm must also agree for equivalent sums of dyads (i.e., all possible dyadic decompositions of the equivalent matrix). So over all equivalent dyadic decompositions, we take the best possible value of (1.4). This leads to the definition of the *nuclear norm*. See Section 3.3 for a more detailed development.

Definition 1.2.3 (Nuclear norm)**.** Consider normed spaces $X = (\mathbb{R}^m, \|\cdot\|_X)$ and $Y = (\mathbb{R}^n, \|\cdot\|_Y)$. For every matrix $\boldsymbol{A} \in \mathbb{M}^{m \times n}$ we define $N_{X,Y}$, the nuclear norm on $X \otimes Y$, as

$$N_{X,Y}(\boldsymbol{A}) := \inf \left\{ \sum_i \|\boldsymbol{x}_i\|_X \|\boldsymbol{y}_i\|_Y : \boldsymbol{A} = \sum_i \boldsymbol{x}_i \otimes \boldsymbol{y}_i \right\}. \tag{1.5}$$

Alternatively, we have

$$N_{X,Y}(\boldsymbol{A}) := \inf \left\{ \|\boldsymbol{\lambda}\|_{\ell_1} : \boldsymbol{A} = \sum_i \lambda_i \boldsymbol{x}_i \otimes \boldsymbol{y}_i,\ \|\boldsymbol{x}_i\|_X = 1,\ \|\boldsymbol{y}_i\|_Y = 1 \right\}.$$

Both infima are taken over all (finite) decompositions of $\boldsymbol{A}$.

As desired, we can then interpret the nuclear norm as measuring the "cost" of constructing the matrix from a dyadic decomposition. Adding more dyads to the construction increases the cost as does increasing the magnitude of the

dyads. That is, adding the dyad $\boldsymbol{x} \otimes \boldsymbol{y}$ to the decomposition incurs cost $\|\boldsymbol{x}\|_X \|\boldsymbol{y}\|_Y$. We compute the nuclear norm using the least costly decomposition.

This suggests that we can use nuclear norms as regularizers to promote matrices that have structured factorizations. By choosing the norms $\|\cdot\|_X$ and $\|\cdot\|_Y$, we can control which types of factor structures the nuclear norm $N_{X,Y}$ favors. Also, the nuclear norm penalizes the addition of dyads to the factorization, and so we believe that the nuclear norm assists in promoting low-rank solutions. We will make the geometric intuition behind this more clear in Section 3.3.6, but for now we give a familiar example.

1.2.2.1 Example: The trace norm

The $\ell_2 \otimes \ell_2$ nuclear norm is the trace norm $\|\cdot\|_{S_1}$.

Proposition 1.2.4 (The trace norm)**.** *Let $\boldsymbol{A} \in \mathbb{M}^{m \times n}$ have the compact SVD $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^{\mathsf{t}}$. In dyadic notation, we write*

$$\boldsymbol{A} = \sum_{i=1}^{r} \sigma_i \boldsymbol{u}_i \otimes \boldsymbol{v}_i. \tag{1.6}$$

Then the $\ell_2 \otimes \ell_2$ nuclear norm, N_{ℓ_2,ℓ_2}, is

$$\begin{aligned} N_{\ell_2,\ell_2}(\boldsymbol{A}) &= \inf \left\{ \sum_i \|\boldsymbol{x}_i\|_{\ell_2} \|\boldsymbol{y}_i\|_{\ell_2} : \boldsymbol{A} = \sum_i \boldsymbol{x}_i \otimes \boldsymbol{y}_i \right\} \\ &= \sum_{i=1}^{r} \sigma_i, \end{aligned} \tag{1.7}$$

where the infimum runs over all decompositions of $\boldsymbol{A}$. We conclude that $N_{\ell_2,\ell_2}(\boldsymbol{A}) = \|\boldsymbol{A}\|_{S_1}$.

We prove the proposition in Section 3.4. If, however, the SVD (1.6) is the optimal decomposition in the infimum (1.7), then it is clear that the conclusion holds.

Now—if we continue to think of the nuclear norm as measuring the cost of constructing a matrix—the cost of adding the dyad $\boldsymbol{x} \otimes \boldsymbol{y}$ to the *optimal* decomposition is $\|\boldsymbol{x}\|_{\ell_2} \|\boldsymbol{y}\|_{\ell_2}$. In particular, the magnitude of the dyads is measured with respect to the ℓ_2 norms of their factors. We seek factors with low total energy. The trace norm results from searching over all decompositions and only considering the one with the lowest cost.

We provide additional examples of nuclear norms in Section 3.5.

1.2.3 The nuclear norm recovery problem

This thesis focuses on the use of nuclear norms to regularize structured matrix recovery problems. We modify the regularized recovery problem (1.1) accordingly.

Assume that we have access to noisy linear measurements $\boldsymbol{b}$ of an underlying true matrix $\boldsymbol{A}^\natural \in \mathbb{M}^{m\times n}$. That is,

$$\boldsymbol{b} = \boldsymbol{\mu}(\boldsymbol{A}^\natural) + \boldsymbol{z},$$

where $\boldsymbol{\mu}\colon \mathbb{M}^{m\times n} \to \mathbb{R}^p$ is a linear measurement map and $\boldsymbol{z} \in \mathbb{R}^p$ is additive noise.

Further assume that $\boldsymbol{A}^\natural$ admits a factorization

$$\boldsymbol{A}^\natural = \sum_{i=1}^{r} \boldsymbol{x}_i^\natural \otimes \boldsymbol{y}_i^\natural,$$

where the $\boldsymbol{x}_i^\natural \in (\mathbb{R}^m, \|\cdot\|_X)$ and the $\boldsymbol{y}_i^\natural \in (\mathbb{R}^n, \|\cdot\|_Y)$ have low complexity as measured by the X and Y norms.

We approximate $\boldsymbol{A}^\natural$ by solving

$$\underset{\boldsymbol{A}}{\text{minimize}} \quad \frac{1}{2}\|\boldsymbol{b} - \boldsymbol{\mu}(\boldsymbol{A})\|_{\ell_2}^2 + \lambda N_{X,Y}(\boldsymbol{A}), \tag{1.8}$$

where $N_{X,Y}$ is the nuclear norm on $X \otimes Y$ (1.5), and $\lambda > 0$ is a penalty constant controlling the balance between solution complexity and measurement fidelity. Here we have chosen the squared ℓ_2 norm as the convex loss function.

The key point is that (1.8) is a *convex* program! Nuclear norms allow us to convexify structural constraints on the factors of a matrix.

Solving (1.8) numerically while demonstrating the utility of nuclear norms is the focus of this thesis. One main contribution of this work is the development of a Python software package (`operfact`) to model and solve these problems. We discuss the details of the software in Chapter 4. Our other main contribution is the extension of the nuclear norm recovery problem to structured *operators*. We turn our attention to this now.

1.3 Operators

While we have used matrices to motivate nuclear norms, we will consider a more general set of tensors for the remainder of this thesis. We call these tensors

operators, and we construct them by generalizing our definition of dyad to also allow for matrix factors. Even though this modification may seem simple, it results in novel uses for nuclear norms.

1.3.1 Definition

The above definition of dyads (1.2) considers only the tensor product of two vectors. But what about the tensor product of two matrices? Or a matrix and a vector? These objects certainly exist, and we again choose to define them through a correspondence with rank-1 matrices.

Definition 1.3.1 (Dyad, redux)**.** Let $\mathrm{vec}(\cdot)$ be the mapping that takes a matrix in $\mathbb{M}^{d_1 \times d_2}$ and returns the vector in $\mathbb{R}^{d_1 d_2}$ created by stacking the columns of the matrix in order from left to right. Given $\boldsymbol{X} \in \mathbb{M}^{m \times n}$ and $\boldsymbol{Y} \in \mathbb{M}^{p \times q}$, let the dyad $\boldsymbol{X} \otimes \boldsymbol{Y}$ be the rank-1 matrix

$$\boldsymbol{X} \otimes \boldsymbol{Y} := \mathrm{vec}(\boldsymbol{X})\,\mathrm{vec}(\boldsymbol{Y})^{\mathsf{t}} \in \mathbb{M}^{mn \times pq}. \tag{1.9}$$

In the case where $n = q = 1$, this definition corresponds with Definition 1.2.1.

The operation $\otimes$ in (1.9) is *not* the Kronecker product. We are instead representing the dyad $\boldsymbol{X} \otimes \boldsymbol{Y} \in \mathbb{M}^{m \times n} \otimes \mathbb{M}^{p \times q}$ as a matrix in $\mathbb{M}^{mn \times pq}$ through the *Choi-Jamiołkowski isomorphism*. This mapping is prominent in quantum information theory; see Watrous's lecture notes [Wat11] for more details. We use this particular representation for its convenience in stating subsequent results. Note, however, that this Choi-Jamiołkowski representation and the Kronecker product are themselves related through an isomorphism.

An *operator* is then any finite linear combination of dyads

$$\mathcal{A} = \sum_i \lambda_i \boldsymbol{X}_i \otimes \boldsymbol{Y}_i.$$

For the same dimensions as in the definition above, we denote the linear space of operators as $\mathbb{O}^{m \times n \otimes p \times q}$.

1.3.2 The action of an operator

We call these objects operators because we can indeed regard them as linear mappings. First, we can define the action of the dyad $\boldsymbol{X} \otimes \boldsymbol{Y} \in \mathbb{M}^{m \times n} \otimes \mathbb{M}^{p \times q}$ on a tensor product of vectors as

$$(\boldsymbol{X} \otimes \boldsymbol{Y})(\boldsymbol{u} \otimes \boldsymbol{v}) := \boldsymbol{X}\boldsymbol{u} \otimes \boldsymbol{Y}\boldsymbol{v} = \boldsymbol{X}(\boldsymbol{u}\boldsymbol{v}^{\mathsf{t}})\boldsymbol{Y}^{\mathsf{t}},$$

where $\boldsymbol{u} \in \mathbb{R}^n$ and $\boldsymbol{v} \in \mathbb{R}^q$. In other words, we can view a dyad as the tensor product of two linear transformations.

We then extend this definition by linearity to obtain a linear mapping $\boldsymbol{X} \otimes \boldsymbol{Y} \colon \mathbb{M}^{n\times q} \to \mathbb{M}^{m\times p}$. For any matrix $\boldsymbol{M} = \sum_i \boldsymbol{u}_i \otimes \boldsymbol{v}_i$, we have that

$$(\boldsymbol{X} \otimes \boldsymbol{Y})\left(\sum_i \boldsymbol{u}_i \otimes \boldsymbol{v}_i \right) = \sum_i \boldsymbol{X}(\boldsymbol{u}_i \boldsymbol{v}_i^{\mathsf{t}})\boldsymbol{Y}^{\mathsf{t}} = \boldsymbol{X}\boldsymbol{M}\boldsymbol{Y}^{\mathsf{t}}.$$

Finally, another linear extension allows us to view the operator $\mathcal{A} = \sum_i \boldsymbol{X}_i \otimes \boldsymbol{Y}_i$ as the linear map from $\mathbb{M}^{n\times q}$ to $\mathbb{M}^{m\times p}$ given by

$$\mathcal{A}(\boldsymbol{M}) = \left(\sum_i \boldsymbol{X}_i \otimes \boldsymbol{Y}_i \right)(\boldsymbol{M}) = \sum_i (\boldsymbol{X}_i \otimes \boldsymbol{Y}_i)(\boldsymbol{M}) = \sum_i \boldsymbol{X}_i \boldsymbol{M} \boldsymbol{Y}_i^{\mathsf{t}}, \tag{1.10}$$

for any $\boldsymbol{M} \in \mathbb{M}^{n\times q}$.

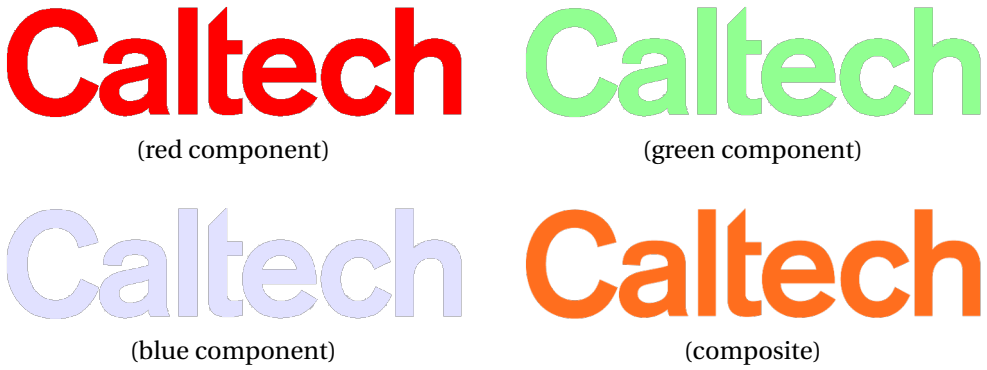
We point out that the matrix representation (1.9) of an operator given by the Choi-Jamiołkowski isomorphism is *not* used to compute (1.10). The Kronecker product, however, may be used to perform this computation. In any case, the representations are again isomorphic.

1.3.3 Why operators?

We have two main motivations for using operators.

A connection to multidimensional arrays. Even though we equate dyads with rank-1 matrices, we see above that we can also think of these operators as 4-dimensional arrays (or 3-dimensional arrays). Indeed, the entries of the matrix representation of $\boldsymbol{X} \otimes \boldsymbol{Y}$ contain all products $x_{ij}y_{kl}$ of the entries of $\boldsymbol{X}$ and $\boldsymbol{Y}$. We could therefore choose to index entries of the operator by using 4 numbers (instead of 2).

Note that in multilinear algebra it is common to construct tensors like $\boldsymbol{a}\otimes\boldsymbol{b}\otimes\boldsymbol{c}\otimes\boldsymbol{d}$ that are truly considered as 4-dimensional arrays. Such objects are said to have *tensor order* 4. They are the natural extensions of one-dimensional arrays (vectors) and two-dimensional arrays (matrices). While it may be convenient for us to consider the entries of our operators as four-dimensional arrays, note that we have defined our objects by correspondence to matrices (order-2 tensors). That is, our operators satisfy *bilinear* identities as opposed to more involved multilinear identities. In this way we rely solely on bilinear algebra, which is an important mathematical and computational distinction.



(red component) (green component)

(blue component) (composite)

Figure 1.1: **RGB Image.** Each pixel of the Caltech logotype (bottom right) is the sum of red, green, and blue components. This RGB image is therefore a hyperspectral image with 3 spectral bands.

Lifting matrices. Operators also allow us to consider additional structure in *matrix* factorizations by using nuclear norms. We note that the factorization $\boldsymbol{A} = \boldsymbol{X}\boldsymbol{Y}^{\mathsf{t}}$ with inner dimension r has a connection to the rank-1 operator $\boldsymbol{X} \otimes \boldsymbol{Y}$ when viewed as the linear mapping (1.10). That is,

$$\boldsymbol{A} = \boldsymbol{X}\boldsymbol{Y}^{\mathsf{t}} \quad \text{if and only if} \quad \boldsymbol{A} = (\boldsymbol{X} \otimes \boldsymbol{Y})(\mathbf{I}_r) = \boldsymbol{X}\mathbf{I}_r\boldsymbol{Y}^{\mathsf{t}}.$$

The key point is that we can view the matrix $\boldsymbol{A}$ as a linear image of the operator $\boldsymbol{X} \otimes \boldsymbol{Y}$. Therefore the problem of factoring $\boldsymbol{A}$ becomes a problem of finding (and factorizing) a rank-1 operator given its action on a matrix (i.e., the identity).

While nuclear norms for the matrix $\boldsymbol{A}$ only allow us to consider structure on the *columns* of $\boldsymbol{X}$ and $\boldsymbol{Y}$, nuclear norms on the lifted operator $\boldsymbol{X} \otimes \boldsymbol{Y}$ allow us to consider structure on the factors as a whole. This change of perspective enables us to consider nuclear norms that can account for additional types of two-dimensional structure present in the factors of these matrix decompositions.

We should note, however, that this lifting procedure carries a cost. In particular, the dimension of the search space rises and the number of measurements relative to that dimension falls. This suggests that the matrix factors $\boldsymbol{X}$ and $\boldsymbol{Y}$ must be highly structured in order for recovery to succeed.

1.3.3.1 Examples of operators

To get a sense for the utility of operators, let us consider some practical examples.

Hyperspectral images. Consider an $m \times n$ color image in RGB format. That is, for each pixel we have three intensity values: red, green, and blue. We could

store this image as the operator

$$\mathcal{A} = \sum_{i=1}^{3} \boldsymbol{X}_i \otimes \boldsymbol{e}_i \in \mathbb{O}^{m \times n \otimes 3 \times 1},$$

where $\boldsymbol{X}_1$, $\boldsymbol{X}_2$, and $\boldsymbol{X}_3$ are the matrices giving the red, green, and blue intensities for each pixel, and the $\boldsymbol{e}_i \in \mathbb{R}^3$ are standard basis vectors. Figure 1.1 illustrates this using the Caltech logotype. This notion extends to images storing intensities at any number of measured wavelengths, and such hyperspectral images have applications in remote sensing [Row+74; RGA77; VG88; VG93], astronomy [Heg+03], quality control [KCM01; Gow+07; Rod+05], and medical diagnosis [AKKT10]. We use the nuclear norm framework to denoise hyperspectral images in Chapter 6.

Two-dimensional time series. We can represent two-dimensional time series as three-dimensional arrays. The entry a_{ijk} of such an array $\mathcal{A}$ would denote the (i, j)th value at time k. A familiar concrete example is video, where a_{ijk} would be the intensity of light at pixel (i, j) of the kth frame.

Graphs with multiple linkages. Graphs have natural matrix representations through adjacency matrices. Assume that $G = (V, E)$ is a directed, unweighted graph on the ordered set of vertices V with edges in E. Let $\boldsymbol{X}$ be the $|V| \times |V|$ matrix such that $x_{ij} = 1$ if an edge exists from vertex i to vertex j and $x_{ij} = 0$ otherwise. Then $\boldsymbol{X}$ is the adjacency matrix for G.

We can imagine situations where we wish to represent various types of connections between the same set of vertices. For instance, Padgett [Pad94] cataloged the business and marital ties between prominent Renaissance-era Florentine families. Dunlavy et al. [DKK12] constructed a network between academic papers that not only contained citation links but also additional links such as authorship and keyword similarity. These situations may be modeled mathematically as a set of graphs $\{G_k = (V, E_k)\}_k$ with corresponding adjacency matrices $\{\boldsymbol{X}_k\}_k$. We may then combine this multiply-linked graph into the operator

$$\mathcal{A} = \sum_{k} \boldsymbol{X}_k \otimes \boldsymbol{e}_k.$$

If we regard $\mathcal{A}$ as a three-dimensional array, then the entry $a_{ijk} = 1$ if an edge of the kth kind exists from vertex i to vertex j. The above-mentioned work of Dunlavy et al., in fact, uses tensor decomposition methods to analyze a similarly constructed tensor.

Lifted matrices. As we just demonstrated, the matrix $\boldsymbol{A} = \boldsymbol{X}\boldsymbol{Y}^\intercal$ may be viewed as a linear image of the rank-1 operator $\boldsymbol{X} \otimes \boldsymbol{Y}$. This enables data models based on matrix factorization to be considered as low-rank operator recovery problems. In Section 2.1.1 we present a number of such models found in the literature.

1.3.4 Nuclear norms

The nuclear norm definition also generalizes to the operator setting with little modification.

Definition 1.3.2 (Nuclear norm)**.** Consider normed spaces $X = (\mathbb{M}^{m\times n}, \|\cdot\|_X)$ and $Y = (\mathbb{M}^{p\times q}, \|\cdot\|_Y)$. For every operator $\mathcal{A} \in \mathbb{O}^{m\times n\otimes p\times q}$, we define $N_{X,Y}$, the nuclear norm on $X \otimes Y$, as

$$N_{X,Y}(\mathcal{A}) := \inf\left\{ \sum_i \|\boldsymbol{X}_i\|_X \|\boldsymbol{Y}_i\|_Y : \mathcal{A} = \sum_i \boldsymbol{X}_i \otimes \boldsymbol{Y}_i \right\}.$$

Alternatively, we have

$$N_{X,Y}(\mathcal{A}) := \inf\left\{ \|\boldsymbol{\lambda}\|_{\ell_1} : \mathcal{A} = \sum_i \lambda_i \boldsymbol{X}_i \otimes \boldsymbol{Y}_i,\ \|\boldsymbol{X}_i\|_X = 1,\ \|\boldsymbol{Y}_i\|_Y = 1 \right\}.$$

Both infima are taken over all (finite) decompositions of $\mathcal{A}$.

This definition corresponds exactly to that of nuclear norms on matrices. It again measures the total cost of building an operator from dyads, penalizing the number and magnitude of dyads (as measured by the norms of their factors).

Similarly, this results in the nuclear norm recovery problem for operators. We approximate the operator $\mathcal{A}^\natural$ by solving

$$\underset{\mathcal{A}}{\text{minimize}} \quad \frac{1}{2}\|\boldsymbol{b} - \boldsymbol{\mu}(\mathcal{A})\|_{\ell_2}^2 + \lambda N_{X,Y}(\mathcal{A}), \tag{1.11}$$

where $\boldsymbol{\mu}$ is a linear measurement map, and $\boldsymbol{b} = \boldsymbol{\mu}(\mathcal{A}^\natural)$ are the observed measurements of the true operator $\mathcal{A}^\natural$. Here $N_{X,Y}$ is a nuclear norm on operators. This approach, again, results in a convex program that can promote the recovery of operators with distinguished factor structure.

1.4 Our contributions and roadmap

The main contributions of this thesis are an extension of nuclear norms to the setting of operators; a software package to model and solve the nuclear

norm recovery problem (1.11); and an empirical study of the effectiveness of nuclear norms as regularizers in operator recovery problems. We use examples in denoising and self-calibration for our numerical experiments.

In Chapter 2 we discuss the relationship between this work and the literature in bilinear models and decomposition norms. Chapter 3 presents relevant results of the *nuclear norm framework* for operator recovery problems. This work is largely based on the unpublished paper [Tro12] of Joel Tropp.

In Chapter 4 we describe the development and use of the Python package `operfact` that we created to enable the rapid prototyping of operator models and nuclear norm recovery problems. Our open-source software, available on GitHub[2], provides the versatility to experiment with nuclear norm models in an object-oriented fashion. We include various linear measurement models and nuclear norms in the base package but also allow for additional expansion.

In Chapter 5 we use our software package to investigate the performance of nuclear norms in synthetic denoising experiments. We show that incorporating prior information regarding factor structure indeed improves denoising performance versus other nuclear norms and, in particular, the trace norm. Chapter 6 extends these examples to an application in hyperspectral imaging incorporating real data.

Finally we consider a self-calibration application in Chapter 7. We use an operator lifting model to linearize a set of bilinear measurements. Our model can then incorporate the 2-dimensional structure of the underlying signals (as opposed to the 1-dimensional structure in the matrix lifting model).

1.5 Other contributions

In addition to the above, I completed a research project showing that a resource tradeoff exists in solving certain statistical problems via convex optimization. We summarize that work here.

The work of Chandrasekaran and Jordan [CJ13] proposed using a hierarchy of convex relaxations in constrained denoising problems to achieve a time–data tradeoff. In these problems, the accuracy of denoising depends on the number of data samples taken and the local geometry of the constraint set. They

[2] https://github.com/jbruer/operfact

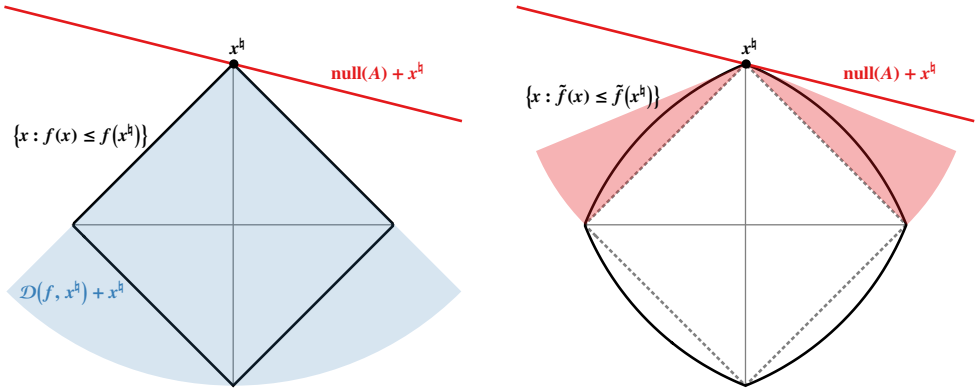


Figure 1.2: **Geometry of a time–data tradeoff.** The left panel shows the geometric exact recovery condition for the regularized linear inverse problem (1.13). The blue shaded area shows the convex cone of directions that decrease f at the true signal $\boldsymbol{x}^\natural$ (the descent cone), and the red line signifies the null space of the measurement matrix $\boldsymbol{A}$. Provided that these two sets intersect trivially, $\boldsymbol{x}^\natural$ is the unique solution to the regularized linear inverse problem. The right panel shows a relaxed regularizer $\tilde{f}$ and the growth of the descent cone. As the number of measurements of $\boldsymbol{x}^\natural$ grows, the null space shrinks. This provides more "room" to relax the regularizer while maintaining exact recovery. This figure originally appeared in [BTCB14, Fig. 1].

combined this geometric insight with a framework for generating hierarchies of relaxed constraint sets. These relaxed problems become more computationally efficient while providing sufficient accuracy in the presence of growing numbers of samples.

My collaborators and I harnessed the same geometric insight, albeit with a different relaxation method, to establish a resource tradeoff in sparse regression problems. Namely, we considered the situation where $\boldsymbol{x}^\natural \in \mathbb{R}^d$ is a signal of interest and we observe linear measurements

$$\boldsymbol{b} = \boldsymbol{A}\boldsymbol{x}^\natural + \boldsymbol{z}, \tag{1.12}$$

where $\boldsymbol{A} \in \mathbb{R}^{m \times d}$ is a known compressed sensing matrix ($m < d$), and $\boldsymbol{z}$ is noise.

In our first paper [BTCB14] we considered the noiseless case (i.e., $\boldsymbol{z} = \boldsymbol{0}$) and the regularized linear inverse problem

$$\underset{\boldsymbol{x}}{\text{minimize}} \quad f(\boldsymbol{x}) \quad \text{subject to} \quad \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}, \tag{1.13}$$

where f is a convex regularizer that promotes the structure of $\boldsymbol{x}^\natural$. The success of this program depends on the alignment of the null space of $\boldsymbol{A}$ and the convex

cone of directions that decrease f at $\boldsymbol{x}^\natural$. When these two sets intersect trivially, the regularized linear inverse problem recovers the true signal $\boldsymbol{x}^\natural$ exactly. The left panel of Figure 1.2 illustrates this condition.

Following Chandrasekaran and Jordan, we recognized that this recovery condition presents a geometric opportunity. In particular, as the number m of measurements grows, the size of the null space of $\boldsymbol{A}$ shrinks. This allows for increasingly relaxed regularizers—with larger descent cones—while maintaining exact recovery. Provided that these relaxed regularizers allow for faster computation, we can achieve a time–data tradeoff. The right panel of Figure 1.2 demonstrates this opportunity.

To achieve this tradeoff, we proposed a family of strongly convex majorizers $\{f_\mu : \mu > 0\}$ with

$$f_\mu(\boldsymbol{x}) := f(\boldsymbol{x}) + \frac{\mu}{2}\|\boldsymbol{x}\|_{\ell_2}^2. \tag{1.14}$$

Replacing the regularizer f in (1.13) with the strongly convex relaxation f_μ, results in a smooth dual problem that we can then solve using a Nesterov-style accelerated gradient descent technique [Nes05; AT06; BCG11]. As the parameter μ grows, the dual problem becomes smoother and allows for faster convergence.

On the other hand, increasing μ also increases the size of the descent cones of f_μ. To determine how much we can increase μ given a number m of measurements, we must be able to calculate the size of the descent cones of f_μ and compare them to the null space of $\boldsymbol{A}$. Amelunxen et al. [ALMT14] proposed the *statistical dimension* to measure the size of convex cones. They proved that for Gaussian $\boldsymbol{A}$, exact recovery in (1.13) occurs with high probability provided that the number m of measurements exceeds the statistical dimension of the regularizer's descent cone at $\boldsymbol{x}^\natural$.

We calculated the statistical dimension of the descent cones of the smoothed ℓ_1 norm and Schatten 1-norm (trace norm) at sparse vectors and low-rank matrices, respectively. This allowed us to compute the maximal value of μ as a function of the number m of available measurements. Our mathematical result suggests the existence of a time–data tradeoff. We restate that result here in the case where the true signal $\boldsymbol{x}^\natural \in \mathbb{R}^d$ is an s-sparse vector and the regularizer f is the ℓ_1 norm.

Proposition 1.5.1 (Error bound for dual-smoothed sparse vector recovery [BTCB14, Prop. 4.2])**.** *Let $\boldsymbol{x}^\natural$ in the measurement model (1.12) be an s-sparse*

vector in $\mathbb{R}^d$*, and let* $\boldsymbol{A} \in \mathbb{R}^{m\times d}$ *have independent standard Gaussian entries. Take the regularizer* f *in* (1.13) *to be the* ℓ_1 *norm and replace it with the smoothed version* f_μ *in* (1.14)*, where we set* $\mu := \mu(m)$ *to be the maximal value of the smoothing parameter* μ *as a function of the number* m *of measurements. The sequence* $\{\boldsymbol{x}_k\}_k$ *of primal iterates resulting from solving the smoothed version of* (1.13) *using a suitable Nesterov-style accelerated gradient method satisfies*

$$\|\boldsymbol{x}^\natural - \boldsymbol{x}_k\|_{\ell_2} \leq \frac{2d^{\frac{1}{2}}\kappa(\boldsymbol{A})\left[\rho\cdot(1+\mu(m)\|\boldsymbol{x}^\natural\|_{\ell_\infty})^2 + (1-\rho)\right]^{\frac{1}{2}}}{\mu(m)\cdot k},$$

where $\rho := s/d$ *is the sparsity of* $\boldsymbol{x}^\natural$*, and* $\kappa(\boldsymbol{A})$ *is the condition number of* $\boldsymbol{A}$*.*

The result suggests that the error at the kth iteration decreases roughly as $1/\mu(m)$, and so a time–data tradeoff exists. We confirmed the existence of the tradeoff through numerical experimentation.

Our subsequent work [BTCB15] considered the noisy case, i.e., where $\boldsymbol{z}$ in (1.12) is nonzero. The reasoning therein for the existence of a resource tradeoff is similar to the noiseless case. We utilized the work of Oymak and Hassibi [OH15] characterizing the stability of the phase transition in Amelunxen et al. [ALMT14] to estimate the accuracy of the (smoothed) regularized regression problem

$$\underset{\boldsymbol{x}}{\text{minimize}} \quad f_\mu(\boldsymbol{x}) \quad \text{subject to} \quad \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_{\ell_2} \leq \epsilon.$$

This again admits a dual problem solvable using Nesterov-style accelerated gradient methods [Nes07].

We again proved theoretical results suggesting a resource tradeoff. In this case, however, we may balance sample size, computational time, *and* statistical accuracy. Numerical results confirmed the existence of these tradeoffs with both synthetic data and an image interpolation problem.

Chapter 2

Bilinear modeling

In this chapter we review bilinear data models appearing in the literature. Section 2.1 highlights the use of bilinear models in various application areas. We survey two major techniques used to recover or factor bilinear models in Section 2.2. Finally, we discuss the history of nuclear norms—the focus of our efforts—in Section 2.3.

2.1 Bilinear models in practice

In this section, we examine several examples of bilinear models and discuss their historical development. We split the examples into two categories: matrix models and lifted models.

2.1.1 Matrix factorization

We call *matrix factorization models* those where the data naturally arise in matrix (or operator) form, and where structured factorizations of those matrices lead directly to interpretations of the models.

2.1.1.1 Sparse PCA

Consider an $m \times n$ data matrix $\boldsymbol{A}$ where the m rows represent observations over n variables. Principal component analysis (PCA) aims to find linear combinations of the original n variables that correspond to the directions of maximal variance in the data. These new "variables" are called *principal components*, and if a small number of them capture much of the variation in the original data set, we

can achieve dimension reduction by simply transforming our original variables to the linear space spanned by that small subset.

If we assume that the columns of $\boldsymbol{A}$ have zero mean, we can compute the principal components through the singular value decomposition (SVD)

$$\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^{\mathrm{t}} \quad \text{or} \quad \boldsymbol{U}\boldsymbol{\Sigma} = \boldsymbol{A}\boldsymbol{V},$$

where the columns of $\boldsymbol{U}\boldsymbol{\Sigma}$ are the principal components, and the columns of $\boldsymbol{V}$ are the *loadings*. In general we see that the principal components are linear combinations of all n original variables. This hinders interpretability. We might, instead, prefer that the principal components are linear combinations of a small subset of the original variables. That is, we would like the columns of $\boldsymbol{V}$ to have few nonzero entries.

Jeffers [Jef67] proposed thresholding smaller entries of $\boldsymbol{V}$, while Jackson [Jac91] opted for a rounding procedure. Jolliffe [Jol95] investigated applying rotations to achieve simpler loadings, but his work with Cadima [CJ95] highlighted some difficulties with all of these procedures. We instead look to procedures that attempt to construct suitable loadings $\boldsymbol{V}$ directly from the data, as opposed to post-processing the results of standard PCA.

The earlier work of Hausman [Hau82] considered restricting loadings to the set $S = \{0, -1, +1\}$. Kolda and O'Leary [KO00] noted the similarity between principal component analysis and *semidiscrete decomposition*—an SVD-like decomposition where the entries of the factors may only take values in S. Vines [Vin00] extended this approach to consider non-unit integer loadings as well.

Jolliffe et al. [JTU03], inspired by the sparsity-inducing LASSO [Tib96], proposed a PCA procedure called SCoTLASS that constrained the ℓ_1 norms of the loadings, i.e., the columns of $\boldsymbol{V}$. Finding the first column of $\boldsymbol{V}$ requires solving

$$\boldsymbol{v}_1 = \arg\max_{\boldsymbol{v}} \boldsymbol{v}^{\mathrm{t}}\boldsymbol{A}^{\mathrm{t}}\boldsymbol{A}\boldsymbol{v} \quad \text{subject to} \quad \|\boldsymbol{v}\|_{\ell_2} = 1, \|\boldsymbol{v}\|_{\ell_1} \leq s. \tag{2.1}$$

This nonconvex maximization problem adds an ℓ_1 constraint to the usual "maximal variance" approach to PCA. A semidefinite lifting and relaxation approach by d'Aspremont et al. [dEJL07] provides a convex approach to this problem, and we revisit it in Section 2.1.2.1.

Zou et al. [ZHT06] cast PCA as a ridge regression problem and promoted sparsity by using the elastic net [ZH05]. Their alternating minimization scheme to solve

the resulting nonconvex problem, termed *sparse PCA*, allowed for larger-scale computation.

Shen and Huang [SH08] and Witten et al. [WTH09] approached this problem by adding additional constraints to the SVD. In the case of finding the first principal component, this becomes

$$\begin{aligned} \underset{\sigma>0,\boldsymbol{u},\boldsymbol{v}}{\text{minimize}} \quad & \frac{1}{2}\|\boldsymbol{A}-\sigma\boldsymbol{u}\boldsymbol{v}^{\mathrm{t}}\|_{\ell_2}^2 \\ \text{subject to} \quad & \|\boldsymbol{u}\|_{\ell_2}^2 = \|\boldsymbol{v}\|_{\ell_2}^2 = 1 \\ & f_1(\boldsymbol{u}) \le \gamma_1,\, f_2(\boldsymbol{v}) \le \gamma_2, \end{aligned}$$

where the f_i are penalty functions chosen to induce structure on the $\boldsymbol{u}$, $\boldsymbol{v}$. Witten et al. considered this general form, which they called the *penalized matrix decomposition*. Shen and Huang called their approach *regularized SVD* and set $f_2 = \|\cdot\|_{\ell_1}$ with no penalty f_1. This more specific case bears resemblance to the $\ell_2 \otimes \ell_1$ nuclear norm.

We return to the problem of sparse PCA in Section 2.1.2.1 where we consider lifting models. Before moving on, however, we wish to highlight the appearance of nuclear norms in the related problem of *robust PCA*—principal component analysis in the presence of outliers. Under this model, the data matrix $\boldsymbol{A} \in \mathbb{M}^{m\times n}$ comprises m observations in $\mathbb{R}^n$. The observations are assumed to approximately lie in a low-dimensional subspace of $\mathbb{R}^n$ so that the data matrix $\boldsymbol{A}$ is the mixture

$$\boldsymbol{A} = \boldsymbol{L} + \boldsymbol{S},$$

where $\boldsymbol{L}$ is the true low-rank data and $\boldsymbol{S}$ are the corruptions known as outliers.

Candès et al. [CLMW11]—using an approach studied by Chandrasekaran et al. [CSPW11]—proposed solving robust PCA with the convex demixing problem

$$\text{minimize} \quad \|\boldsymbol{L}\|_{S_1} + \lambda\|\boldsymbol{S}\|_{\ell_1} \quad \text{subject to} \quad \boldsymbol{A} = \boldsymbol{L} + \boldsymbol{S}.$$

The trace norm ($\|\cdot\|_{S_1}$) is also the $\ell_2\otimes\ell_2$ nuclear norm (Section 3.4) and promotes low-rank $\boldsymbol{L}$. Meanwhile, the ℓ_1 norm is the $\ell_1 \otimes \ell_1$ nuclear norm (Section 3.5) and promotes sparse $\boldsymbol{S}$. The assumption under this model is that the outliers result as corruptions in a small fraction of all entries of $\boldsymbol{A}$.

McCoy and Tropp [MT11] and Xu et al. [XCS12] independently proposed a slightly different demixing method. They considered the data matrix $\boldsymbol{A}$ as arising

from the mixture

$$\boldsymbol{A} = \boldsymbol{L} + \boldsymbol{S},$$

where $\boldsymbol{L}$ is low-rank and $\boldsymbol{S}$ has few nonzero rows. Again, the matrix $\boldsymbol{L}$ is the true low-rank model, but now entire observations (rows of $\boldsymbol{A}$) may be outliers.

They then formulated robust PCA as the convex demixing problem

$$\text{minimize} \quad \|\boldsymbol{L}\|_{S_1} + \lambda \sum_i \|\boldsymbol{s}_i\|_{\ell_2} \quad \text{subject to} \quad \boldsymbol{A} = \boldsymbol{L} + \boldsymbol{S},$$

where the $\boldsymbol{s}_i$ are the *rows* of $\boldsymbol{S}$. Observe that the sum of the ℓ_2 norms of the rows of $\boldsymbol{S}$ is the $\ell_1 \otimes \ell_2$ nuclear norm (Section 3.5). The use of the $\ell_1 \otimes \ell_2$ nuclear norm to promote row-sparsity also appeared earlier in the literature on simultaneous sparse approximation (Section 2.1.1.4).

2.1.1.2 Dictionary learning

Consider the signal model

$$\boldsymbol{a} = \boldsymbol{X}\boldsymbol{y},$$

where $\boldsymbol{a} \in \mathbb{R}^m$ is a signal composed from linear combinations of the columns of $\boldsymbol{X} \in \mathbb{M}^{m \times r}$ with weights given by $\boldsymbol{y} \in \mathbb{R}^r$. In the theory of *sparse approximation*, the matrix $\boldsymbol{X}$ is a *dictionary*, and its columns are called *atoms*. The goal is to represent (or approximate) the signal $\boldsymbol{a}$ as a *sparse* linear combination of atoms, i.e., the vector $\boldsymbol{y}$ should have few nonzero entries. Benefits of sparse approximations include compressibility and interpretability.

A central question, however, is how to choose the dictionary $\boldsymbol{X}$. Choices include the Fourier transformation, wavelet bases, frames, unions of bases, and random atoms; see Mallat [Mal09] for more details. These designed dictionaries aim to provide sparse representations for many signals of interest, but we could instead consider learning a suitable dictionary from a corpus of known signals. This is exactly the *dictionary learning* problem.

Let us now consider the matrix $\boldsymbol{A} \in \mathbb{M}^{m \times n}$ of n signals in $\mathbb{R}^m$. We seek a factorization

$$\boldsymbol{A} = \boldsymbol{X}\boldsymbol{Y},$$

where $\boldsymbol{X} \in \mathbb{M}^{m \times r}$ is now the *unknown* dictionary, and $\boldsymbol{Y} \in \mathbb{M}^{r \times n}$ contains the unknown weights that generate the corpus. We ask that the columns of $\boldsymbol{Y}$ in this factorization be *sparse*.

Olshausen and Field, a pair of psychologists studying the mammalian visual cortex, ignited the field of dictionary learning with their 1997 paper [OF97]. They learned a dictionary from patches of nature photographs and showed that sparse coding in this dictionary mimicked the observed activity of certain sensory cells. Since then, dictionary learning has enjoyed successes in image denoising [EA06], edge detection [Mai+08], and super-resolution [YWHM10]. We use nuclear norms in a related example when denoising hyperspectral images in Chapter 6.

The last two decades have seen a growing interest in both improved numerical methods and theoretical guarantees for dictionary learning. The alternating methods of Engan et al. [EAH99] (method of optimal directions), Tropp [Tro04] (generalized k-means), and Aharon et al. [AEB06] (K-SVD) share the same general approach. Consider the dictionary learning problem

$$\underset{\boldsymbol{X},\boldsymbol{Y}}{\text{minimize}} \quad \frac{1}{2}\|\boldsymbol{A} - \boldsymbol{X}\boldsymbol{Y}\|_{\ell_2}^2 \quad \text{subject to} \quad \|\boldsymbol{y}_i\|_{\ell_0} \leq s, \ \text{for } i = 1, \ldots, n,$$

where the "ℓ_0 norm" returns the number of nonzero entries in a vector. These methods proceed by alternating between a sparse coding step (fixing $\boldsymbol{Y}$) and a dictionary update (fixing $\boldsymbol{X}$). Agarwal et al. [AAJN16] provide theoretical support for these techniques.

Clustering methods, including work by Agarwal et al. [AAN16] and Arora et al. [AGM14], instead find a subset of the corpus that shares a dictionary element and use that subset to estimate the element. Both papers include guarantees on the success of the clustering procedure, but the guarantees in the latter are stronger.

Note that the nuclear norm framework does not apply directly here as it works to constrain the columns of $\boldsymbol{X}$, and in this case, the *rows* of $\boldsymbol{Y}$. Here we wish the columns of $\boldsymbol{Y}$ to be sparse. Bach et al. [BMP08] do consider a nuclear norm-type approach but with mixed success. A possible alternative involves lifting this matrix problem to operator space as in Section 1.3.3. That is, we consider

$$\boldsymbol{A} = (\boldsymbol{X} \otimes \boldsymbol{Y}^{\mathsf{t}})(\boldsymbol{I}) = \boldsymbol{X}\boldsymbol{Y},$$

and solve a nuclear norm recovery problem on the space of operators. Note that the difficulty here arises from seeing the single action of the dictionary on the identity. While it may not be possible to identify the dictionary from this formulation, it may still provide benefits in denoising-type problems.

2.1.1.3 Nonnegative matrix factorization

The goal of nonnegative matrix factorization (NMF) is to decompose a matrix $\boldsymbol{A} \in \mathbb{M}^{m \times n}$ as

$$\boldsymbol{A} \approx \boldsymbol{X}\boldsymbol{Y},$$

where the nonnegative matrix $\boldsymbol{X} \in \mathbb{M}^{m \times k}$ is the *feature matrix*, and the nonnegative matrix $\boldsymbol{Y} \in \mathbb{M}^{k \times n}$ are the *weights*.

Paatero and Tapper [PT94] first proposed to find such decompositions in order to perform factor analysis on environmental data. In particular, they wised to improve on the interpretability of PCA. Indeed, the nonnegativity constraints ensure that features (columns of $\boldsymbol{X}$) may only be constructively added together to explain the data matrix $\boldsymbol{A}$. This leads to a more intuitive construction of the columns of the data matrix as a positive combination of nonnegative factors. To solve the problem, they used constrained alternating least squares.

A series of papers by Lee and Seung [LS97; LS99; LS01] independently developed the method of nonnegative matrix factorization in the context of unsupervised learning. In this setting, the goal is to write the columns of a data matrix $\boldsymbol{A}$ as linear combinations of the columns of a feature matrix $\boldsymbol{X}$. Again, restricting the features and weights to be positive prevents "cancellation" between the features. That is, we may interpret each column of $\boldsymbol{A}$ as being built up constructively from a library of parts (features). They applied NMF to images of faces (eigenfaces) to demonstrate this. Their multiplicative update method to find these factorizations can be regarded as a form of gradient descent. Tropp's literature review [Tro03] discusses both of these early approaches in more detail.

Notice the similarity in aims with the sparse dictionary learning problem. Indeed, Hoyer [Hoy02; Hoy04] proposed to add a sparseness constraint to the weights $\boldsymbol{Y}$. To solve the nonnegative sparse coding (NNSC) problem, he employed a projected gradient descent method. See also Lin [Lin07] for a more comprehensive study of these methods. The generalized k-means framework in Tropp's thesis [Tro04, Ch. 8] proposed solving NMF and NNSC using alternating minimization. Hyunsoo Kim and Park [KP07] provided an algorithm for sparse NMF using alternating least squares.

Ding et al. [DHS05] examined the connection between NMF and the k-means problem. Namely, the symmetric NMF with an additional orthogonality con-

straint

$$\underset{\boldsymbol{X}\geq 0, \boldsymbol{X}^{\mathsf{t}}\boldsymbol{X}}{\text{minimize}} \quad \|\boldsymbol{A} - \boldsymbol{X}\boldsymbol{X}^{\mathsf{t}}\|_{\mathrm{F}}^2,$$

is equivalent to solving

$$\underset{\boldsymbol{X}\geq 0, \boldsymbol{X}^{\mathsf{t}}\boldsymbol{X}}{\text{maximize}} \quad \operatorname{tr}(\boldsymbol{X}^{\mathsf{t}}\boldsymbol{A}\boldsymbol{X}),$$

the spectral relaxation to k-means [Zha+02].[1] Of particular interest, the symmetric NMF problem without the orthogonality constraint still returns $\boldsymbol{X}$ with nearly orthogonal columns. Jingu Kim and Park [KP08] used an alternating least squares method for sparse NMF to solve the clustering problem.

More recent work provides recovery guarantees for these nonconvex approaches [AGM12; XY13; AGKM16; RRTB12; BGKP16; LLR16].

The work of Bach [Bac13], discussed in Section 2.3.3.3, considered nuclear norm-type problems where the individual factor structures may be penalized using gauges on compact sets (instead of just norms). Creating a gauge on the intersection of ℓ_p-norm balls with the positive orthant results in nuclear norm-type regularizers for NMF.

2.1.1.4 Simultaneous sparse approximation

Consider a set of signals $\boldsymbol{b}_i \in \mathbb{R}^m$, $i = 1, \ldots, k$, each taking the form

$$\boldsymbol{b}_i = \boldsymbol{\Phi}\boldsymbol{x}_i^\natural + \boldsymbol{z}_i,$$

where the columns of $\boldsymbol{\Phi} \in \mathbb{M}^{m\times d}$ are elementary signals, the sparse vectors $\boldsymbol{x}_i^\natural \in \mathbb{R}^d$ are weights, and the $\boldsymbol{z}_i$ are additive noise. The simple sparse approximation problem seeks to recover $\boldsymbol{x}_i^\natural$ given $\boldsymbol{\Phi}$ and $\boldsymbol{b}_i$. Simultaneous sparse approximation concerns the situation where all of the $\boldsymbol{x}_i^\natural$ have the same sparsity pattern. That is, each signal $\boldsymbol{b}_i$ is a noisy linear combination of the same elementary signals.

We can write this model in matrix form as

$$\boldsymbol{B} = \boldsymbol{\Phi}\boldsymbol{X}^\natural + \boldsymbol{Z},$$

where the k columns of $\boldsymbol{B}$, $\boldsymbol{X}^\natural$, and $\boldsymbol{Z}$ comprise the $\boldsymbol{b}_i$,$\boldsymbol{x}_i$, and $\boldsymbol{z}_i$ from above. The goal now is to recover the coefficients $\boldsymbol{X}^\natural$.

[1] In k-means, one attempts to create k clusters of vectors such that each vector belongs to the cluster whose centroid is closest to it.

This problem appears in medical imaging [GGR95; CREK05; Phi+05], communications [CR02], source localization [MCW03; MCW05; Oll15], sensor networks [LGLS06], multidimensional signal processing [SDB13; SDBC17], blind source separation [Gri02], and image processing [FR08].

Tropp et al. [TGS06] developed a greedy algorithm based on orthogonal matching pursuit [PRK93; DMA97] to solve the problem. More relevant to the present work is his convex relaxation method [Tro06]. This formulation proposes relaxing the hard problem

$$\underset{\boldsymbol{X}}{\text{minimize}} \quad \#\text{ of nonzero rows in } \boldsymbol{X} \quad \text{subject to} \quad \|\boldsymbol{B} - \boldsymbol{\Phi}\boldsymbol{X}\|_{\text{F}} \leq \epsilon,$$

by replacing the objective to obtain a convex program

$$\underset{\boldsymbol{X}}{\text{minimize}} \quad \sum_i \max_j |x_{ij}| \quad \text{subject to} \quad \|\boldsymbol{B} - \boldsymbol{\Phi}\boldsymbol{X}\|_{\text{F}} \leq \epsilon.$$

The objective is now the sum of the ℓ_∞ norms of the rows of $\boldsymbol{X}$, and this serves to promote the row-sparsity of $\boldsymbol{X}$. We will see in Section 3.5 that this is exactly the $\ell_1 \otimes \ell_\infty$ nuclear norm!

While this may not seem like a matrix factorization model, consider the structure of $\boldsymbol{X}^\natural$. In the case where the $\boldsymbol{x}_i^\natural$ are identically equal to some vector $\boldsymbol{x}^\natural$, we can write

$$\boldsymbol{X}^\natural = \sum_{i=1}^{d} \boldsymbol{e}_i \otimes x_i^\natural \mathbf{1}_k,$$

where $\mathbf{1}_k \in \mathbb{R}^k$ is the all-ones vector. We now see that the left factors of $\boldsymbol{X}^\natural$ are sparse, and the right factors are constant. The $\ell_1 \otimes \ell_\infty$ nuclear norm is a natural fit. In the case where the $\boldsymbol{x}_i$ are allowed to vary, the $\ell_1 \otimes \ell_2$ nuclear norm may be a better choice.

2.1.1.5 Matrix completion

Consider the problem of recovering a low-rank matrix $\boldsymbol{A}^\natural \in \mathbb{M}^{m \times n}$ while observing only a subset of its entries. Namely, we assume that

$$\boldsymbol{A}^\natural = \boldsymbol{X}\boldsymbol{Y}^{\mathsf{t}},$$

where $\boldsymbol{X} \in \mathbb{M}^{m \times r}$ and $\boldsymbol{Y} \in \mathbb{M}^{n \times r}$.

Say that Ω is the set of indices of $\boldsymbol{A}^\natural$ that we observe. Then we can attempt to find $\boldsymbol{A}^\natural$ by solving the rank minimization problem

$$\underset{\boldsymbol{A}}{\text{minimize}} \quad \text{rank}(\boldsymbol{A}) \quad \text{subject to} \quad a_{ij} = a_{ij}^\natural \text{ for } (i, j) \in \Omega.$$

The work of Srebro [SRJ05; SS05], motivated by the example of collaborative filtering, suggested replacing the hard rank constraint with the max-norm. We show in Example 3.6.10 of Section 3.6.5 that the max-norm is a semidefinite relaxation of the $\ell_\infty \otimes \ell_\infty$ nuclear norm. This relaxation effectively penalizes the largest squared ℓ_2 norms of the rows of $\boldsymbol{X}$ and $\boldsymbol{Y}$.

Candès and Recht [CR09] used the familiar trace norm relaxation for the rank [Faz02] to convexify the matrix completion problem. Using techniques from compressed sensing they proved that $\mathcal{O}(n^{1.2} r \log n)$ observations suffice to recover a rank-r $n \times n$ matrix. Their subsequent works [CT10; Rec11] refined this result. Candès and Plan [CP10] addressed the stability of this procedure in the presence of noise.

Nonconvex algorithms for solving this problem include alternating minimization [JNS12; Har14] and gradient descent [KMO10a; KMO10b]. In fact, alternating least squares is popular in large-scale applications like collaborative filtering [ZWSP08; Kor09; KBV09]. Stochastic gradient algorithms also achieve great speedups on large-scale problems through parallelization [RR13]. Recent work by Ge et al. [GLM16] showed that the nonconvex matrix completion problem for positive semidefinite matrices has no spurious local minima.

Note that the matrix completion problem is a special form of the matrix sensing problem [RFP10] wherein we wish to recover structured matrices from linear measurements. Indeed, we can extract the ij-entry of $\boldsymbol{A}^\natural$ through the inner product $\langle \boldsymbol{E}_{ij}, \boldsymbol{A}^\natural \rangle$, where $\boldsymbol{E}_{ij}$ is a standard basis matrix. Therefore we may instead frame the matrix completion problem as

$$\underset{\boldsymbol{A}}{\text{minimize}} \quad f(\boldsymbol{A}) \quad \text{subject to} \quad \boldsymbol{\mu}(\boldsymbol{A}) = \boldsymbol{b},$$

where $\boldsymbol{\mu}$ is the linear measurement map returning a particular set of entries from a matrix, and $\boldsymbol{b} = \boldsymbol{m}\boldsymbol{u}(\boldsymbol{A}^\natural)$ is the vector of observed entries. This is simply a regularized linear inverse problem. Instead of restricting f to the trace norm, we can choose any other nuclear norm that might better encode the factor structure of $\boldsymbol{A}^\natural$. Our Python package, `operfact`, models these problems using the `SubsampleMeasurement` object (Section 4.5.4).

2.1.2 Lifting models

In this section we consider approaches where the bilinear (or quadratic) measurement model on vectors is lifted to a linear model on a rank-1 matrix.

2.1.2.1 Sparse PCA, redux

We revisit the sparse PCA model (2.1) from Section 2.1.1.1. For simplicity we assume that $\boldsymbol{A} \in \mathbb{M}^{n\times n}$ is the covariance matrix of the n observed zero-mean variables. Finding the first sparse principal component in $\mathbb{R}^n$ corresponds to the maximization problem

$$\underset{\boldsymbol{x}}{\text{maximize}} \quad \boldsymbol{x}^{\mathrm{t}}\boldsymbol{A}\boldsymbol{x} \quad \text{subject to} \quad \|\boldsymbol{x}\|_{\ell_2} = 1 \quad \text{and} \quad \|\boldsymbol{x}\|_{\ell_0} \leq s, \tag{2.2}$$

where $s \leq n$ and the "ℓ_0 norm" returns the number of nonzero entries of the vector.

The approach of d'Aspremont et al. [dEJL07] requires lifting the problem to

$$\begin{aligned} \underset{\boldsymbol{X}}{\text{maximize}} \quad & \operatorname{tr}(\boldsymbol{A}\boldsymbol{X}) \\ \text{subject to} \quad & \operatorname{tr}(\boldsymbol{X}) = 1 \quad \text{and} \quad \|\boldsymbol{X}\|_{\ell_0} \leq s^2, \\ & \boldsymbol{X} \succeq \boldsymbol{0} \quad \text{and} \quad \operatorname{rank}(\boldsymbol{X}) = 1, \end{aligned} \tag{2.3}$$

where we now optimize over $\boldsymbol{X} \in \mathbb{M}^{n\times n}$. The second set of constraints guarantees that any solution $\boldsymbol{X} = \boldsymbol{x}\boldsymbol{x}^{\mathrm{t}}$ for some $\boldsymbol{x}$, and the first set of constraints ensures that $\boldsymbol{x}$ satisfies the same conditions as in (2.2).

The lifted problem (2.3) has turned the quadratic objective into a linear one, and it has replaced the nonconvex norm equality with a linear one. The ℓ_0 and rank constraints still make (2.3) a hard, nonconvex problem. But by dropping the rank constraint and relaxing the ℓ_0 constraint, we may obtain the semidefinite program

$$\begin{aligned} \underset{\boldsymbol{X}}{\text{maximize}} \quad & \operatorname{tr}(\boldsymbol{A}\boldsymbol{X}) \\ \text{subject to} \quad & \operatorname{tr}(\boldsymbol{X}) = 1, \\ & \|\boldsymbol{X}\|_{\ell_1} \leq s, \\ & \boldsymbol{X} \succeq \boldsymbol{0}. \end{aligned}$$

This follows the standard technique of relaxing a hard sparsity constraint to an ℓ_1 constraint. Here we use the fact that if $\operatorname{tr}(\boldsymbol{X}) = 1$ with $\boldsymbol{X} = \boldsymbol{x}\boldsymbol{x}^{\mathrm{t}}$, then $\|\boldsymbol{X}\|_{\ell_0} \leq s^2$ implies that $\|\boldsymbol{X}\|_{\ell_1} \leq s$. This is truly a relaxation.

As we will see in Section 3.5, the ℓ_1 norm of a matrix (i.e., the sum of its absolute entries) coincides with the $\ell_1 \otimes \ell_1$ nuclear norm. So we can think of this semidefinite program as encouraging the recovery of a PSD matrix with a short dyadic decomposition comprising sparse factors. Indeed, this is the goal.

2.1.2.2 Phase retrieval

Phase retrieval is the problem of recovering a signal while having access only to measurements of the magnitude of its Fourier coefficients—and not their phase. This problem has applications in areas such as optics [Wal63], astronomy [DF87], crystallography [Mil90; Har93], microscopy [MCKS99; Mia+02; MISE08], and diffractive imaging [Bun+07].

Mathematically, let $\boldsymbol{x}^\natural \in \mathbb{C}^n$ be a signal, and suppose that we observe quadratic measurements

$$b_k = |\langle \boldsymbol{a}_k, \boldsymbol{x}^\natural \rangle|^2, \quad k = 1, 2, \dots, m, \tag{2.4}$$

where the $\boldsymbol{a}_k$ are known (designed). To solve the phase retrieval problem, we must find a vector $\boldsymbol{x}$ that agrees with the observed measurements.

The classical algorithms for phase retrieval sprout from the alternating projection work of Gerchberg and Saxton [GS72] and Fienup [Fie78; Fie82]. We focus, however, on more recent work that lifts this vector recovery problem to a matrix recovery problem. Balan et al. [BBCE09] recast the quadratic measurements (2.4) of the vector $\boldsymbol{x}^\natural$ into linear measurements of a matrix $\boldsymbol{X}^\natural$ using certain tight frames. Chai et al. [CMP11] used a different linearization scheme that resulted in the lifted matrix $\boldsymbol{X}^\natural$ having rank one. The phase retrieval problem then becomes the familiar one of low-rank matrix recovery given linear measurements. This idea was further promoted by Candès and coauthors [CESV13; CSV13].

To linearize the quadratic measurements (2.4), we use the fact that

$$|\langle \boldsymbol{a}_k, \boldsymbol{x} \rangle|^2 = \operatorname{tr}(\boldsymbol{a}_k^* \boldsymbol{x} \boldsymbol{x}^* \boldsymbol{a}_k) = \operatorname{tr}(\boldsymbol{a}_k \boldsymbol{a}_k^* \boldsymbol{x} \boldsymbol{x}^*).$$

If we let $\boldsymbol{A}_k = \boldsymbol{a}_k \boldsymbol{a}_k^*$, then we have the *linear* measurements

$$b_k = \operatorname{tr}(\boldsymbol{A}_k \boldsymbol{X}^\natural), \tag{2.5}$$

where $\boldsymbol{X}^\natural = \boldsymbol{x}^\natural (\boldsymbol{x}^\natural)^*$. Furthermore, the lifted matrix $\boldsymbol{X}^\natural$ is rank-1 and positive semidefinite.

Define the measurement map $\boldsymbol{\mu} \colon \mathbb{C}^{n \times n} \to \mathbb{C}^m$ such that $[\boldsymbol{\mu}(\boldsymbol{X})]_k = \operatorname{tr}(\boldsymbol{A}_k \boldsymbol{X})$. We can then state the lifted phase retrieval problem as

$$\underset{\boldsymbol{X}}{\text{minimize}} \quad \operatorname{rank}(\boldsymbol{X}) \quad \text{subject to} \quad \boldsymbol{\mu}(\boldsymbol{X}) = \boldsymbol{b} \quad \text{and} \quad \boldsymbol{X} \succeq \boldsymbol{0},$$

where the kth entry of $\boldsymbol{b}$ is the measurement b_k (2.5). By replacing the rank objective with the trace norm, we obtain a convex program for phase retrieval.

In addition to the convex methods, Candès et al. [CLS15] proposed a nonconvex gradient descent method, while Netrapalli et al. [NJS15] used alternating minimization.

Researchers, inspired by the application of low-rank matrix recovery techniques to phase retrieval, have turned to lifting approaches to solve other signal processing problems. We discuss two such problems here.

2.1.2.3 Blind deconvolution

Consider two real signals $\boldsymbol{g}, \boldsymbol{h} \in \mathbb{R}^L$ and their (circular) convolution

$$\boldsymbol{b} = \boldsymbol{g} * \boldsymbol{h}, \quad \text{or} \quad b_l = \sum_{l'=1}^{L} g_{l'} h_{l-l'+1},$$

where the index on h runs modulo the set $\{1, 2, \ldots, L\}$. An important problem in signal processing asks to recover $\boldsymbol{g}$ and $\boldsymbol{h}$ from their convolution $\boldsymbol{b}$. This task has a long history in applications such as astronomy [Bat82; JC93], communications [TXHK95; WP98; WBSJ15], and medical imaging [Dro89; FKL89; WH90; Kri+92].

We focus on a recent approach to convexify this problem. Inspired by the phase retrieval work in [CSV13], Ahmed et al. [ARR14] proposed a lifting approach to formulate blind deconvolution as a rank-1 matrix recovery problem with incomplete measurements. They employed trace-norm minimization to relax the rank constraint while providing conditions that guaranteed recovery of the true low-rank solution. In order to achieve this with incomplete measurements, however, they needed to impose subspace constraints on the signals. Namely, they assumed that $\boldsymbol{g}$ and $\boldsymbol{h}$ belong to M and N dimensional subspaces, respectively. Thus,

$$\begin{cases} \boldsymbol{g} = \boldsymbol{S}\boldsymbol{x}, & \boldsymbol{x} \in \mathbb{R}^M, \boldsymbol{S} \in \mathbb{M}^{L\times M} \\ \boldsymbol{h} = \boldsymbol{T}\boldsymbol{y}, & \boldsymbol{y} \in \mathbb{R}^N, \boldsymbol{T} \in \mathbb{M}^{L\times N}. \end{cases}$$

Then by the convolution theorem [Mal09, Thm. 3.9],

$$\begin{aligned}\hat{\boldsymbol{b}} = \boldsymbol{F}\boldsymbol{y} = \boldsymbol{F}(\boldsymbol{g} * \boldsymbol{h}) &= \boldsymbol{F}\boldsymbol{g} \odot \boldsymbol{F}\boldsymbol{h} \\ &= (\boldsymbol{F}\boldsymbol{S}\boldsymbol{x}) \odot (\boldsymbol{F}\boldsymbol{T}\boldsymbol{y}) \\ &= \hat{\boldsymbol{S}}\boldsymbol{x} \odot \hat{\boldsymbol{T}}\boldsymbol{y} \\ &= \operatorname{diag}(\hat{\boldsymbol{S}}\boldsymbol{x})\hat{\boldsymbol{T}}\boldsymbol{y},\end{aligned}$$

where $\boldsymbol{F}$ is the Fourier transform, and the operation $\odot$ is elementwise vector multiplication.

Then the lth entry of $\hat{\boldsymbol{b}}$ is

$$\begin{aligned}\hat{b}_l &= \langle \hat{\boldsymbol{s}}_l, \boldsymbol{x}\rangle\langle \boldsymbol{y}, \hat{\boldsymbol{t}}_l\rangle \\ &= (\hat{\boldsymbol{s}}_l)^* \boldsymbol{x}\boldsymbol{y}^* \hat{\boldsymbol{t}}_l \\ &= \operatorname{tr}((\hat{\boldsymbol{s}}_l)^* \boldsymbol{x}\boldsymbol{y}^* \hat{\boldsymbol{t}}_l) \\ &= \operatorname{tr}(\hat{\boldsymbol{t}}_l (\hat{\boldsymbol{s}}_l)^* \boldsymbol{x}\boldsymbol{y}^*) \\ &= \operatorname{tr}(\boldsymbol{M}_l^* (\boldsymbol{x}\boldsymbol{y}^*)) = \langle \boldsymbol{M}_l, \boldsymbol{x}\boldsymbol{y}^*\rangle,\end{aligned}$$

where $\boldsymbol{M}_l = \hat{\boldsymbol{s}}_l \hat{\boldsymbol{t}}_l^*$, $\hat{\boldsymbol{s}}_l$ is the lth column of $\hat{\boldsymbol{S}}^*$, and $\hat{\boldsymbol{t}}_l$ is the lth row of $\hat{\boldsymbol{T}}$. We can therefore view the measurements $\hat{b}_l$ as the inner products of the matrices $\boldsymbol{M}_l$ with the rank-1 matrix $\boldsymbol{x}\boldsymbol{y}^*$. Convolution is thereby represented as a *linear* operator on this rank-1 matrix.

We can perform the deconvolution by solving

$$\underset{\boldsymbol{A}}{\text{minimize}} \quad \|\boldsymbol{A}\|_{S_1} \quad \text{subject to} \quad \operatorname{tr}(\boldsymbol{M}_l^* \boldsymbol{A}) = \hat{b}_l.$$

If the resulting $\boldsymbol{A}$ is rank-1, a simple factorization will find candidate vectors $\boldsymbol{x}$, $\boldsymbol{y}$, and we can compute candidate signals $\boldsymbol{g}$ and $\boldsymbol{h}$. As mentioned in Section 1.2.2.1 (and proved in Section 3.4), the trace norm is the $\ell_2 \otimes \ell_2$ nuclear norm. Ahmed et al. used low-rank matrix recovery techniques to provide guarantees on the success of this program for blind deconvolution. In the case where the Fourier coefficients $\boldsymbol{y}$ are sparse, Flinth [Fli16] showed that the $\ell_2 \otimes \ell_1$ nuclear norm is a superior choice.

There has also been recent work on nonconvex techniques to solve this problem. Lee et al. [LLJB15] provided performance guarantees for an alternating minimization solver. Li et al. [LLSW16] used a nonconvex gradient descent method (Wirtinger flow) to improve performance computationally and in terms of required data.

2.1.2.4 Self-calibration

Consider an array of sensors used to measure and report signals. In practical settings, however, sensors do not give exact measurements. Their outputs are better viewed as a combination of the inputs we wish to measure *and* some additional parameters that we may not control. The problem of *self-calibration* is to estimate both the true signal and the calibration parameters of our sensor(s) from these uncalibrated outputs.

This problem arises in distributed sensing, where the sensors themselves have inaccuracies in gain (or phase) [BN07; BN08; WRS08; LB14]. Direction-of-arrival estimation [FW88; FW91; See94; NS96; LY06; Liu+11] seeks to estimate a signal and the direction from where it came using sensor arrays. Source localization in acoustics aims to identify the location of sound sources using microphonic arrays [MDO11]. Autofocusing is used to increase range resolution in radar imaging [ZWBY14]

Mathematically, consider a signal $\boldsymbol{y} \in \mathbb{R}^N$, and assume that we have measurements of that signal

$$\boldsymbol{b} = \boldsymbol{T}(\boldsymbol{x})\boldsymbol{y} + \boldsymbol{z},$$

where the measurement matrix $\boldsymbol{T}(\boldsymbol{x}) \in \mathbb{M}^{L\times N}$ depends on a vector of parameters $\boldsymbol{x} \in \mathbb{R}^M$ and $\boldsymbol{z} \in \mathbb{R}^L$ is noise. Even if we assume that $\boldsymbol{T}(\boldsymbol{x})$ depends linearly on $\boldsymbol{x}$, solving for $\boldsymbol{y}$ is a bilinear inverse problem. Furthermore, the problem still may be underdetermined even if we know $\boldsymbol{T}(\boldsymbol{x})$ exactly.

Note the similarity to the blind deconvolution problem above. Indeed, we can regard blind deconvolution as a specific type of self-calibration problem. The distinguishing feature is that here the measurements do not necessarily result from a convolution.

Our primary motivation to consider this problem is a lifting model proposed by Ling and Strohmer [LS15b]. They were inspired by the previously-discussed lifting procedures to convexify phase retrieval [CSV13; CESV13] and blind deconvolution [ARR14]. Their method, termed SparseLift, transforms the bilinear self-calibration problem into an ℓ_1-minimization problem on a rank-1 matrix.

Earlier work of Balzano and Nowak [BN07; BN08] provided an approach to use multiple snapshots (observations from a sensor array at different times) to reformulate self-calibration as a solvable linear system. Ling and Strohmer [LS16]

recently considered a linear least squares approach with performance guarantees.

Gribonval et al. [GCD12] also considered a convex approach for calibrating compressed sensing problems with unknown gains on the measurements. The work of Bilen et al. [BPGD14] extended this method to handle phase uncertainty in measurements as well. A new paper by Wang and Chi [Wan16] provides theoretical guarantees.

We describe each of these approaches in more detail in Section 7.2. We also note that Cambareri and Jacques [CJ16a] proposed a new gradient descent method that mirrors the work of Li et al. [LLSW16] in blind deconvolution. In Chapter 7 we use our `operfact` package to conduct numerical experiments on self-calibration problems.

2.2 Numerical techniques for bilinear models

In the previous section we examined several examples of bilinear data models. The numerical techniques used to solve these problems fall into several broad categories. Here, we briefly summarize a few such methods.

2.2.1 Convexification

While not a numerical method *per se*, we have seen convexification as an approach to solve simultaneous sparse regression [Tro06], sparse PCA [dEJL07], phase retrieval [CSV13; CESV13], blind deconvolution [ARR14; ACD15], and self-calibration [GCD12; BPGD14; FS14; LS15b; LS15a]. These examples lift the bilinear (or quadratic) problems on vectors into linear measurements of a low-rank matrix. The solution then proceeds via convex relaxation of the low-rank constraint using, for instance, the trace norm [Faz02; RFP10]. Standard semidefinite solvers (or frontends such as CVX [GB14] and CVXPY [DB16]) can then solve the problem.

It is increasingly common, however, in large-scale applications to use nonconvex techniques such as alternating minimization [JNS12; Har14] or gradient descent [KMO10a; KMO10b] (see also below). In these cases, the nonconvex approaches can provide great advantages in terms of storage and parallelization. While convex programs are desirable for their convergence guarantees, these

nonconvex approaches work well empirically. A growing body of research aims to explain this phenomenon [ZL15; ZWL15; CW15; SL15; GLM16].

Note that our nuclear norm framework also works to convexify bilinear data models. Indeed, the nuclear norms serve as convex regularizers on operators that promote structure within their factors. We also resort to nonconvex methods to solve these problems, but we do so for their flexibility in modeling.

2.2.2 Alternating minimization

The idea behind alternating minimization is simple: we optimize over sets of variables in turn while holding the others fixed.[2] To see its benefit with factorization models, consider the general low-rank matrix sensing problem

$$\underset{\boldsymbol{A}}{\text{minimize}} \quad \text{rank}(\boldsymbol{A}) \quad \text{subject to} \quad \boldsymbol{\mu}(\boldsymbol{A}) = \boldsymbol{b},$$

where $\boldsymbol{A} \in \mathbb{M}^{m\times n}$ is the decision variable, and $\boldsymbol{b} \in \mathbb{R}^p$ are measurements resulting from applying the linear operator $\boldsymbol{\mu}$ to an unobserved low-rank matrix $\boldsymbol{A}^\natural$.

We have already discussed how to solve this problem via convex relaxation and the trace norm. Instead, let us explicitly factorize $\boldsymbol{A} = \boldsymbol{X}\boldsymbol{Y}^\mathsf{t}$, where $\boldsymbol{X} \in \mathbb{M}^{m\times k}$, $\boldsymbol{Y} \in \mathbb{M}^{n\times k}$, and k is a target rank. Then we reformulate our optimization problem as

$$\underset{\boldsymbol{X}\in\mathbb{M}^{m\times k},\boldsymbol{Y}\in\mathbb{M}^{n\times k}}{\text{minimize}} \quad \frac{1}{2}\|\boldsymbol{b} - \boldsymbol{\mu}(\boldsymbol{X}\boldsymbol{Y}^\mathsf{t})\|_{\ell_2}^2 =: f(\boldsymbol{X}, \boldsymbol{Y}). \tag{2.6}$$

This problem is easily solvable when holding either $\boldsymbol{X}$ or $\boldsymbol{Y}$ fixed and optimizing over the other. Given an initialization $\boldsymbol{Y}_0$, the updating procedure at each iteration t is then

$$\begin{aligned}\boldsymbol{X}_{t+1} &\leftarrow \arg\min_{\boldsymbol{X}} f(\boldsymbol{X}, \boldsymbol{Y}_t)\\ \boldsymbol{Y}_{t+1} &\leftarrow \arg\min_{\boldsymbol{Y}} f(\boldsymbol{X}_{t+1}, \boldsymbol{Y}),\end{aligned}$$

where the subscript on the matrices indicates the iteration.

Alternating minimization is used to solve sparse PCA [ZHT06; SH08; WTH09], dictionary learning [EAH99; Tro04; AEB06; AAJN16; XY13], nonnegative matrix factorization [PT94; Tro04; KP07; KP08], matrix completion [JNS12; Har14], phase retrieval [NJS15], and blind deconvolution [LLJB15].

[2] This method is also referred to as *block coordinate descent* in the literature.

The explicit factorization allows for a large reduction in storage requirements and processing with rank $k \ll \min\{m, n\}$. Additionally, simple subproblems may admit opportunities for parallelization to take advantage of multiple computational cores in large-scale settings. Lastly, the alternating approach provides flexibility in modeling. This last reason, in particular, led us to implement an alternating minimization approach to solving nuclear norm problems (see Section 4.2). It allows us to easily prototype many different nuclear norms.

On the other hand, alternating minimization generally does not guarantee convergence to a global minimum. Furthermore, it can exhibit great sensitivity to its initialization. We discuss initialization below in Section 2.2.4, and some of the previously-cited papers do manage to provide guarantees for their specific applications. Despite these potential shortcomings, however, alternating methods have still shown empirical success and remain a staple of nonconvex optimization.

2.2.3 Gradient methods

Gradient methods work similarly to alternating minimization. Consider again the low-rank matrix sensing problem (2.6) from above. Assume that we have an initialization $[\, X_0 \; Y_0 \,]^t$. At each iteration t, we update the decision variables through the gradient step

$$\begin{bmatrix} \boldsymbol{X}_{t+1} \\ \boldsymbol{Y}_{t+1} \end{bmatrix} \leftarrow \begin{bmatrix} \boldsymbol{X}_t - \eta \nabla f(\boldsymbol{X}, \boldsymbol{Y}_t) \\ \boldsymbol{Y}_t - \eta \nabla f(\boldsymbol{X}_t, \boldsymbol{Y}) \end{bmatrix},$$

where $\eta > 0$ is the step size.

We also have gradient methods to solve dictionary learning [AGMM15], nonnegative matrix factorization [LS01; Hoy04; Lin07], matrix completion [KMO10a; KMO10b; RR13], phase retrieval [CLS15], blind deconvolution [LLSW16], and self-calibration [CJ16b].

Despite its relatively slow convergence, gradient descent has grown more popular with the advent of parallelized stochastic gradient descent methods [BT97; ZLS09; ZWLS10; RRWN11] that can take advantage of multicore and distributed computing systems for large-scale optimization.

Applying gradient descent to nonconvex problems raises questions about initialization and convergence. As opposed to convex optimization, we cannot

guarantee that gradient descent on the nonconvex problem converges to the global optimum. In fact, gradient descent may fail to converge to even a local minimum given an unfavorable initialization [Nes04, Sec. 1.2.3]. Recent work, however, showed that for objectives satisfying the *strict saddle property*[3], gradient descent (with an appropriate step size) converges almost surely to a local minimum [LSJR16]. A line of work by Sun and coauthors demonstrated that problems in dictionary learning [SQW17a; SQW17b] and phase retrieval [SQW16] indeed satisfy this property.

For our work, however, the flexibility afforded by using alternating minimization in conjunction with existing convex solver packages proves most valuable. The extensibility of our software package `operfact` certainly allows for future incorporation of various gradient solvers if applications demand it.

2.2.4 Initialization for nonconvex methods

Both the nonconvex alternating minimization and gradient descent algorithms require careful initializations. As opposed to convex optimization problems, the nonconvex versions may have spurious local minima. That is, these local minima are not globally optimal solutions. Furthermore, the convergence to any particular point in the nonconvex case may depend on the starting point provided to the algorithm.

Consider again the nonconvex formulation (2.6) of the matrix sensing problem. Observe that the point $(\boldsymbol{X}, \boldsymbol{Y}) = (\mathbf{0}, \mathbf{0})$ is always a local minimum, but clearly this is not a global minimum in general. This also shows why initialization to zero does not work in these settings. Indeed, if we apply gradient descent starting at this point, we will find that we have already converged. Therefore we need to examine initialization schemes more closely.

One method for initialization simply chooses a random point. After solving the nonconvex minimization problem for several different random initializations, we can then choose the solution that has the lowest objective value. In generating these random points, we can also take into account problem-specific constraints. For instance, if we seek a nonnegative factorization we may force our random initial factors to have only nonnegative entries [Hoy04; Lin07; KP07;

[3] A twice continuously differentiable function $f : \mathbb{R}^d \to \mathbb{R}$ possesses the strict saddle property if every critical point $\boldsymbol{x}$ is either a local minimum or $\nabla^2 f(\boldsymbol{x})$ has at least one eigenvalue strictly less than zero.

KP08]. Random initialization may also rely on observed data. Tropp's alternating minimization method for matrix factorization initializes the columns of the factor $\boldsymbol{X}$ with random columns from the observed matrix $\boldsymbol{A}$ [Tro04].

A large downside of random initialization is the possibility of having to solve the optimization problem multiple times. It would be preferable to construct a deterministic initialization that would hopefully result in good convergence. A *spectral initialization* does precisely this. Generally these methods rely on a spectral decomposition, the SVD, to find an initial point given the observed data. In sparse PCA, for example, one can use classical PCA as an initial guess [ZHT06; SH08; WTH09]. This is simply the SVD.

The measurements in the low-rank matrix sensing problem (2.6), however, are incomplete. Indeed, we observe the vector $\boldsymbol{b} = \boldsymbol{\mu}(\boldsymbol{A}^\natural) + \boldsymbol{z}$ instead of the matrix $\boldsymbol{A}^\natural$ itself. To perform a spectral initialization, we can first compute $\boldsymbol{T} = \boldsymbol{\mu}^*(\boldsymbol{b})$, where $\boldsymbol{\mu}^*$ is the adjoint of the measurement map $\boldsymbol{\mu}$. Then we can compute the SVD of $\boldsymbol{T}$ and initialize $\boldsymbol{X}$ and $\boldsymbol{Y}$ with the top-r left and right singular vectors of $\boldsymbol{T}$, where r is the rank of the explicit factorization $\boldsymbol{A} = \boldsymbol{X}\boldsymbol{Y}^\mathsf{t}$.

Keshavan, Montanari, and Oh [KMO10a; KMO10b] used this initialization for solving matrix completion with gradient descent. Their method additionally involves trimming and rescaling the adjoint—here, simply the matrix of observed entries. Jain et al. [JNS12] and Hardt [Har14] use this same initialization in their analyses of alternating minimization algorithms for solving matrix completion. Spectral initializations also show success in phase retrieval [NJS15; CLS15], blind deconvolution [LLJB15; LLSW16], and self-calibration [CJ16b].

Given the similarity of our structured operator recovery problems to matrix sensing, we adopt spectral initialization for our work as well. We discuss our implementation in Section 4.2.3.

2.3 Development of the nuclear norm

The mathematical development of the nuclear norm in Chapter 3 follows the unpublished work [Tro12] of my advisor Joel Tropp. What we refer to as the nuclear norm, however, has a long history in functional analysis. In this section, we review that history and discuss some more recent works that are more similar in character to our own.

2.3.1 The emergence of cross spaces

In 1927, Hitchcock proposed writing multidimensional arrays (tensors) as sums of elementary, rank-1 tensors [Hit27; Hit28]. His work, along with that of Oldenburger in the 1930s [Old34; Old36], established the algebraic character of tensor product spaces. They considered how rank, an important algebraic complexity measure of matrices, extended to tensors. These types of polyadic decompositions gained popularity in the 1970s with CANDECOMP [CC70] and PARAFAC [Har70]. See the comprehensive review of Kolda and Bader [KB09] for more details on this line of research.

Our discussion of nuclear norms naturally depends on the *topology* of linear spaces created from sums of elementary tensors. Namely, what can be said about the topology of $X \otimes Y$, the linear space of dyads $\boldsymbol{x} \otimes \boldsymbol{y}$ with $\boldsymbol{x} \in X$ and $\boldsymbol{y} \in Y$. Francis Murray and John von Neumann [MN36] considered the case where both X and Y were Hilbert spaces.

Robert Schatten, Murray's doctoral student, focused on the more general case where X and Y were Banach spaces. His 1950 monograph *A Theory of Cross-Spaces* [Sch50] collected his work (partly in collaboration with von Neumann) from throughout the 1940s [Sch43; Sch46; SN46; SN48] devoted to addressing the question of norms on $X \otimes Y$.

A *cross-space* results from first forming the linear space of products from X and Y taking the form

$$\sum_{i=1}^{r} \boldsymbol{x}_i \otimes \boldsymbol{y}_i,$$

where the product $\otimes$ observes bilinear identities (see Section 3.2.1). Schatten viewed such objects as operators from Y^*, the dual of Y, to X, and attached norms to this space. He considered a particular class of norms, however, called crossnorms (hence the name cross-space). Such norms (which we denote generally using the notation $|||\cdot|||$) satisfied the additional condition that

$$|||\boldsymbol{x} \otimes \boldsymbol{y}||| = \|\boldsymbol{x}\|_X \|\boldsymbol{y}\|_Y,$$

for all pairs $\boldsymbol{x} \in X$, $\boldsymbol{y} \in Y$.[4]

[4] Schatten used α to indicate a general crossnorm, whereas we use $|||\cdot|||$.

Of particular interest to us, Schatten proved the existence of a largest (and smallest) crossnorm. This largest crossnorm, which he denoted as γ, is

$$\gamma\left(\sum_i \boldsymbol{x}_i \otimes \boldsymbol{y}_i\right) = \inf\left\{\sum_j \|\boldsymbol{x}'_j\|_X \|\boldsymbol{y}'_j\|_Y : \sum_j \boldsymbol{x}'_j \otimes \boldsymbol{y}'_j = \sum_i \boldsymbol{x}_i \otimes \boldsymbol{y}_i\right\},$$

where the infimum extends over all equivalent representations of the tensor. This largest crossnorm, also called the *projective tensor norm*, corresponds exactly to our Definition 1.2.3 of the nuclear norm.[5] A key contribution by Schatten is the notion that this largest crossnorm is *universal*. That is, it exists for the tensor product of any pair of Banach spaces.

The interpretation that holds the most meaning for the present work—and which we expand on through our discussion in Chapter 3—is that this largest crossnorm is the strongest. That is, its unit ball is contained inside that of all other crossnorms. Since we wish to use this norm as a regularizer to find operators with structured factors, it is advantageous that the norm assigns the greatest penalty (among crossnorms) for deviating from that structure.

The smallest crossnorm, known as the *injective tensor norm* and denoted as λ by Schatten, appears in our work as the dual of the nuclear norm (Definition 3.3.7). While Schatten primarily dealt with these two crossnorms, there is in fact a rich structure of crossnorms that we now examine briefly.

2.3.2 The fundamental theorem of Grothendieck

Alexandre Grothendieck published his famous *Résumé* [Gro53] in 1953. Within he also formed the product space of Banach spaces X and Y as the linear span of dyads. On these spaces he consider $\otimes$-norms (tensor norms), crossnorms with an extra condition bounding their size in combination with linear operators on the factors.[6] These include the greatest (projective) norm from above, denoted in Grothendieck's work as $\|\cdot\|_\wedge$, and the least (injective) norm, denoted $\|\cdot\|_\vee$.

In addition, he showed that there exist 12 other "natural" $\otimes$-norms. These are the only 14 $\otimes$-norms (up to equivalencies), and they form a complete lattice [Gro53, p. 37]. Of particular interest to us are two famous theorems that

[5] The term *nuclear norm* has a precise definition, and while it coincides with the projective tensor norm in finite dimensions, this is *not* the case in infinite dimensions. We deal with finite dimensional spaces in this work, and so we adopt to term nuclear norm to emphasize the connection with current low-rank matrix recovery literature. See Ryan [Rya02, Sec. 2.6] for a fuller explanation.

[6] Schatten and von Neumann [SN48] called these *uniform* crossnorms.

bound the equivalence of certain tensor norms. What Grothendieck [Gro53, Thm. 1 (p. 59), Coro. 2 (p. 60)] called the "fundamental theorem in the metric theory of tensor products", now known as Grothendieck's Theorem, compares the $\ell_\infty \otimes \ell_\infty$ nuclear norm (in our notation) and its semidefinite relaxation known as the max-norm in the machine learning literature [SRJ05; SS05]. We discuss this in Section 3.6.6.1.[7] Additionally the "little" Grothendieck theorem [Gro53, Thm. 4, Coro. 1 (p. 51)] compares the $\ell_\infty \otimes \ell_2$ nuclear norm with its semidefinite relaxation. See our discussion in Section 3.6.6.2.

Even though the results of Grothendieck are widely celebrated today, they were not immediately recognized. Possible reasons include the difficulty of reading his *Résumé* [Pie07, Sec. 6.3.11.8] or its publication in an obscure Brazilian journal [Pis12, p. 238]. The work, 15 years later, of Lindenstrauss and Pełczyński [LP68] revived the main result of Grothendieck's work and reformulated it in terms of matrices instead of tensor product spaces.

Many references exist for further reading on the topology of tensor product spaces. Ryan [Rya02] provides a more accessible introduction to the material. Meanwhile Pietsch [Pie07] details the history of Banach spaces. Diestel et al. [DGFS08] revisit Grothendieck's *Résumé*, following its structure while expanding and modernizing the content. Pisier [Pis12] reviews the development and applications of Grothendieck's theorems.

2.3.3 With an eye towards convex optimization

We now turn our attention to more recent work that applies the nuclear norm to recovering structured matrix factorizations.

2.3.3.1 A familiar example

In Section 1.2.2.1 we already saw an example of a nuclear norm used in matrix recovery problems: the trace norm. In our notation, this is the $\ell_2 \otimes \ell_2$ nuclear norm, but it is often referred to as *the* nuclear norm in the literature. As we discussed, Fazel [Faz02] used the trace norm as a convex relaxation for matrix rank. The motivation to do so is that the trace norm is the convex envelope of the rank, the tightest convex relaxation for the rank. We address the creation of nuclear norms from sets of structured atoms in Section 3.3.6. Her work with

[7] In the tensor product literature, the relaxation is known as the *Hilbertian norm*.

Recht et al. [RFP10] provided guarantees on the effectiveness of the trace norm in solving the *matrix sensing* problem

$$\underset{\boldsymbol{A}}{\text{minimize}} \quad \|\boldsymbol{A}\|_{S_1} \quad \text{subject to} \quad \boldsymbol{\mu}(\boldsymbol{A}) = \boldsymbol{b},$$

where $\boldsymbol{\mu}$ is a random, linear measurement map, and the entries $\boldsymbol{b}$ are the corresponding linear measurements of the true low-rank matrix we wish to recover.

Their results extended the compressed sensing approach [CRT06a; Don06] where the goal is to recover sparse vectors from undersampled random linear measurements. They generalized the restricted isometry property (RIP) of Candès and Tao [CRT06a] to matrices, showing that for certain random measurement maps , $\mathcal{O}(r(m+n)\log(mn))$ measurements suffice to recover a rank-r $m \times n$ matrix using trace-norm minimization. Recent work in convex geometry provides exact characterizations of the number of required Gaussian measurements for matrix sensing to succeed [CRPW12; ALMT14].

2.3.3.2 Rediscovering the projective tensor norm

Meanwhile, Francis Bach et al. [BMP08] considered the problem of dictionary learning (see also Section 2.1.1.2). Recall that in this problem we wish to find a factorization of a signal corpus $\boldsymbol{A}$,

$$\boldsymbol{A} = \boldsymbol{X}\boldsymbol{Y}^{\mathsf{t}},$$

where the matrix $\boldsymbol{X}$ is a dictionary, and the matrix $\boldsymbol{Y}$ of coefficients has sparse rows (i.e., $\boldsymbol{Y}^{\mathsf{t}}$ has sparse columns). They aimed to convexify the nonconvex approach of optimizing directly over the $\boldsymbol{X}$ and $\boldsymbol{Y}$ subject to constraints on their structure.

Their solution, termed *decomposition norms*, was the projective tensor norm. They considered the function (restated to match our notation)

$$f_r(\boldsymbol{A}) = \min\left\{ \sum_{i=1}^{r} \|\boldsymbol{x}_i\|_X \|\boldsymbol{y}_i\|_Y : \boldsymbol{A} = \sum_{i=1}^{r} \boldsymbol{x}_i \otimes \boldsymbol{y}_i \right\}.$$

This closely resembles the definition of the nuclear norm (1.5) except for the fact that the minimization occurs only over decompositions of length r versus all finite-length decompositions. By taking the limit $f_\infty(\boldsymbol{A}) = \lim_{r\to\infty} f_r(\boldsymbol{A})$, they recovered the nuclear norm on $X \otimes Y$ and thus obtained a convex regularizer for imposing structure on the factors of a matrix decomposition.

To solve the dictionary learning problem, they considered a combination of the ℓ_1 and ℓ_2 norms on the coefficient matrix with the ℓ_2 norm on the dictionary elements. That is,

$$\|\boldsymbol{x}\|_X^2 = \|\boldsymbol{x}\|_{\ell_2}^2 \quad \text{and} \quad \|\boldsymbol{y}\|_Y^2 = (1-\nu)\|\boldsymbol{y}\|_{\ell_1}^2 + \nu\|\boldsymbol{y}\|_{\ell_2}^2.$$

They then defined the function

$$F(\boldsymbol{y}\boldsymbol{y}^{\mathsf{t}}) := (1-\nu)\sum_{ij}(|\boldsymbol{y}\boldsymbol{y}^{\mathsf{t}}|)_{ij} + \nu\,\mathrm{tr}(\boldsymbol{y}\boldsymbol{y}^{\mathsf{t}}) = \|\boldsymbol{y}\|_Y^2.$$

This allowed them to rework the dictionary learning problem as one over positive semidefinite matrices, but the convex formulation required allowing the rank to be any finite number. By applying a smoothing procedure (to obtain a twice-differentiable approximation of F), they were able to establish that any rank-deficient local minimum of the rank-constrained version of this problem was in fact a global minimum [BMP08, Prop. 4]. The argument followed that of Burer and Monteiro [BM04].

We should point out, however, that just because no spurious local minima exist does not mean that all stationary points of the nonconvex problem are global minima. Furthermore, finding local minima (as opposed to critical points) may still be challenging. Even so, their work demonstrated both the promise of nuclear norms and some of the computational difficulties in their application.

Haeffele et al. [HYV14] further extended the above result to allow for some nondifferentiable F. They proposed a rank-constrained block coordinate descent procedure [XY13] for the factor structures

$$\|\cdot\|_X = \nu_X\|\cdot\|_{\tilde{X}} + \|\cdot\|_{\ell_2} \quad \text{and} \quad \|\cdot\|_Y = \nu_Y\|\cdot\|_{\tilde{Y}} + \|\cdot\|_{\ell_2},$$

where ν_X and ν_Y are user-chosen parameters. Their simulations showed success in performing image segmentation and compressed sensing on hyperspectral images. (We consider a hyperspectral image denoising problem in Chapter 6.)

2.3.3.3 Generalizing the results

In a subsequent work [Bac13], Bach revisited the notion of decomposition norms. This time, however, he formulated them as a gauge functions on sets of structured dyads. See Section 3.3.6 for our similar construction and a discussion of its connection to sparse approximation and atomic norms [DT96;

CRPW12]. Bach considered the more general case of using gauges to measure the complexity of each factor in dyadic decompositions, whereas we restrict our discussion to norms. He also formed the semidefinite relaxation to nuclear norms we discuss in Section 3.6. Furthermore, Bach characterized the quality of the semidefinite relaxations using a technique from Nesterov [Nes98], itself a rediscovery of Grothendieck's Theorem. See the discussion of Grothendieck's work above and our statement of the result in Section 3.6.6.1. Note that Tropp [Tro12] independently characterized these semidefinite relaxations.

Haeffele and Vidal [HV15] returned to consider general nuclear norms on $X \otimes Y$ as well as applications with higher-order tensors. In our context of regularized matrix factorization problems, their results state that any local minimizer of the rank-constrained problem

$$\text{minimize} \quad \frac{1}{2}\left\| \boldsymbol{b} - \sum_{i=1}^{r} \boldsymbol{x}_i \otimes \boldsymbol{y}_i \right\|_{\ell_2}^2 + \lambda \sum_{i=1}^{r} \|\boldsymbol{x}_i\|_X \|\boldsymbol{y}_i\|_Y,$$

is a global minimizer provided that $\boldsymbol{x}_i = \boldsymbol{0}$ and $\boldsymbol{y}_i = \boldsymbol{0}$ for some i. Furthermore, if the $\boldsymbol{x}_i \in \mathbb{R}^m$ and the $\boldsymbol{y}_i \in \mathbb{R}^n$, then the global minimizer requires at most mn dyads.

They additionally provided a "meta-algorithm" that guarantees finding the global minimizer of the recovery problem via local descent from any initializer. Critically, this "meta-algorithm" requires the ability to determine that the local descent procedure reaches a local minimum of the rank-constrained problem. This is non-trivial, and alternating minimization algorithms can generally only guarantee convergence to a stationary point. In practice, this makes the simple condition for global optimality an unverifiable one.

2.3.4 Our work

Our work builds on the literature in three main ways. First, we extend the use of nuclear norms to structured *operator* recovery problems. We can view these operators, introduced in Section 1.3, as liftings for matrices. This enables us to use nuclear norms to promote a larger variety of structured matrix factorizations.

Second, we provide a flexible Python package (`operfact`) to allow for rapid prototyping of nuclear norm models. This, combined with an alternating minimization solver built upon CVXPY, enable us to systematically test the performance of nuclear norms in structured low-rank recovery problems. We release

this software to the community as an open-source project [8], and this thesis describes its design (Chapter 4).

Third, we use this software package to perform a systematic numerical study of nuclear norms in denoising problems (Chapter 5). We provide solid evidence that nuclear norms succeed in recovery problems over a variety of different factor structures. This knowledge enables us to model problems in hyperspectral imaging (Chapter 6) and self-calibration (Chapter 7) as convex operator recovery problems. Using nuclear norms to match the factor structures of our models, we then solve these problems numerically.

[8] https://github.com/jbruer/operfact

Chapter 3

The nuclear norm

The nuclear norm has a history in Banach space theory that we surveyed in Section 2.3. This chapter gives a mathematical introduction to nuclear norms and presents results relevant to this thesis. Our treatment is based on the unpublished work [Tro12] of my advisor Joel Tropp. We restate many of his ideas and results here in the interest of making this document self-contained, but we develop the framework for the more general case of operators (Section 1.3) instead of matrices.

3.1 Notation

Vectors. We let $\mathbb{R}^m$ be the space of real m-dimensional vectors, and we denote vectors with bold, lowercase, italic letters, e.g., $\boldsymbol{a}$. We write entries of $\boldsymbol{a}$ as the scalars a_i. This space is endowed with the usual operations of vector addition and scalar multiplication, and we write the standard basis $\{\boldsymbol{e}_i\}_{i=1}^m$. Additionally, it has the inner product

$$\langle \boldsymbol{a}, \boldsymbol{b} \rangle = \boldsymbol{a}^\mathsf{t} \boldsymbol{b} = \sum_{i=1}^m a_i b_i,$$

where t indicates the transpose.

Matrices. We let $\mathbb{M}^{m \times n}$ be the space of $m \times n$ matrices with real entries. We write matrices using bold, uppercase letters, e.g., $\boldsymbol{A}$, with scalar entries a_{ij}. This space has the usual operations of matrix addition and scaling, and we write the standard basis as $\{\boldsymbol{E}_{ij}\}$, where $\boldsymbol{E}_{ij} = \boldsymbol{e}_i \otimes \boldsymbol{e}_j = \boldsymbol{e}_i \boldsymbol{e}_j^\mathsf{t}$ for $i = 1, \ldots, m$ and

$j = 1, \ldots, n$. It has the inner product

$$\langle \boldsymbol{A}, \boldsymbol{B} \rangle = \operatorname{tr}(\boldsymbol{A}^{\mathsf{t}}\boldsymbol{B}) = \sum_{i=1}^{m} \sum_{j=1}^{n} a_{ij} b_{ij},$$

where t is again the transpose, and tr is the trace operator.

We will sometimes refer to rows and columns of matrices using colon notation. For instance the vectors $\boldsymbol{a}_{i:}$ and $\boldsymbol{a}_{:j}$ refer respectively to the ith row and jth column of the matrix $\boldsymbol{A}$.

Operators. We write operators with bold, uppercase, script letters, e.g., $\boldsymbol{\mathcal{A}}$. In the next section, we discuss additional notational conventions for operators.

Vectorization. For a matrix $\boldsymbol{A} \in \mathbb{M}^{m \times n}$, we define the vectorization operator $\operatorname{vec}\colon \mathbb{M}^{m \times n} \to \mathbb{R}^{mn}$ as the map returning the columns of the matrix stacked in order from left to right.

Norms. We use the standard ℓ_p norms on vectors. We adopt the convention that applying a vector norm on a matrix is equivalent to vectorizing the matrix and applying the norm.

The Schatten p-norm of the matrix $\boldsymbol{A}$ is defined as

$$\|\boldsymbol{A}\|_{S_p} := \left(\sum_{i=1}^{r} \sigma_i(\boldsymbol{A})^p \right)^{\frac{1}{p}},$$

where $\sigma_1(\boldsymbol{A}) \geq \sigma_2(\boldsymbol{A}) \geq \cdots \geq \sigma_r(\boldsymbol{A}) \geq 0$ are the singular values of $\boldsymbol{A}$. In particular, the Schatten 1-norm is the trace norm, the Schatten 2-norm is the Frobenius (Euclidean) norm, and the Schatten ∞-norm is the spectral norm.

3.2 Dyads and operators

Since decompositions are at the heart of nuclear norms we review the notation introduced in Section 1.3.1 and discuss additional properties of operators.

3.2.1 Dyads

First let us restate the definition of a *dyad* from Section 1.3.1.

Definition 3.2.1 (Dyad)**.** Given $\boldsymbol{X} \in \mathbb{M}^{m \times n}$ and $\boldsymbol{Y} \in \mathbb{M}^{p \times q}$, let the dyad $\boldsymbol{X} \otimes \boldsymbol{Y}$ be the rank-1 matrix

$$\boldsymbol{X} \otimes \boldsymbol{Y} := \operatorname{vec}(\boldsymbol{X}) \operatorname{vec}(\boldsymbol{Y})^{\mathsf{t}} \in \mathbb{M}^{mn \times pq}. \tag{3.1}$$

In the case where $n = q = 1$, this definition corresponds to the usual notion of a rank-1 matrix as the outer product of two vectors.

We emphasize that we are defining dyads—elementary tensors in the tensor product space $\mathbb{M}^{m\times n} \otimes \mathbb{M}^{p\times q}$—through a particular correspondence with matrices in $\mathbb{M}^{mn\times pq}$. In particular, we are *not* using the operation $\otimes$ to represent the Kronecker product. The mapping (3.1) is known as the *Choi-Jamiołkowski isomorphism*, and it is prominent in quantum information theory. See, for example, Watrous's lecture notes [Wat11] for more details. We choose this specific representation as it proves more convenient for subsequent results.

Bilinear relationships. Two dyads are equal if and only if their corresponding rank-1 matrices are equal. This makes it easy to see that dyads indeed obey the bilinear equivalencies of tensor products:

- $(\boldsymbol{X} \otimes \boldsymbol{Y}) + (\boldsymbol{X}' \otimes \boldsymbol{Y}) = (\boldsymbol{X} + \boldsymbol{X}') \otimes \boldsymbol{Y}$,
- $(\boldsymbol{X} \otimes \boldsymbol{Y}) + (\boldsymbol{X} \otimes \boldsymbol{Y}') = \boldsymbol{X} \otimes (\boldsymbol{Y} + \boldsymbol{Y}')$,
- $(\lambda \boldsymbol{X}) \otimes \boldsymbol{Y} = \lambda(\boldsymbol{X} \otimes \boldsymbol{Y}) = \boldsymbol{X} \otimes (\lambda \boldsymbol{Y})$,

where $\boldsymbol{X}, \boldsymbol{X}' \in \mathbb{R}^{m\times n}$, $\boldsymbol{Y}, \boldsymbol{Y}' \in \mathbb{R}^{p\times q}$, and $\lambda \in \mathbb{R}$ is a scalar. In particular $\boldsymbol{X} \otimes \boldsymbol{Y} = \boldsymbol{0}$, the zero dyad, when $\boldsymbol{X} = \boldsymbol{0}$ or $\boldsymbol{Y} = \boldsymbol{0}$.

Tensor product of linear maps. A dyad may also represent the tensor product of linear maps. For $\boldsymbol{X} : \mathbb{R}^n \to \mathbb{R}^m$ and $\boldsymbol{Y} : \mathbb{R}^q \to \mathbb{R}^p$, we define the action of the dyad $\boldsymbol{X} \otimes \boldsymbol{Y}$ on the tensor product of vectors as

$$(\boldsymbol{X} \otimes \boldsymbol{Y})(\boldsymbol{u} \otimes \boldsymbol{v}) := \boldsymbol{X}\boldsymbol{u} \otimes \boldsymbol{Y}\boldsymbol{v} = (\boldsymbol{X}\boldsymbol{u})(\boldsymbol{Y}\boldsymbol{v})^{\mathsf{t}} = \boldsymbol{X}(\boldsymbol{u}\boldsymbol{v}^{\mathsf{t}})\boldsymbol{Y}^{\mathsf{t}},$$

where $\boldsymbol{u} \in \mathbb{R}^n$ and $\boldsymbol{v} \in \mathbb{R}^q$.

We can extend this definition linearly to obtain a mapping $\boldsymbol{X} \otimes \boldsymbol{Y} : \mathbb{M}^{n\times q} \to \mathbb{M}^{m\times p}$. For any matrix $\boldsymbol{M} = \sum_i \boldsymbol{u}_i \otimes \boldsymbol{v}_i \in \mathbb{M}^{n\times q}$, we have that

$$(\boldsymbol{X} \otimes \boldsymbol{Y})(\boldsymbol{M}) = \sum_i (\boldsymbol{X} \otimes \boldsymbol{Y})(\boldsymbol{u}_i \otimes \boldsymbol{v}_i) = \sum_i \boldsymbol{X}(\boldsymbol{u}_i \boldsymbol{v}_i^{\mathsf{t}})\boldsymbol{Y}^{\mathsf{t}} = \boldsymbol{X}\boldsymbol{M}\boldsymbol{Y}^{\mathsf{t}}. \tag{3.2}$$

In this context $\boldsymbol{X} \otimes \boldsymbol{Y}$ is a linear map, but note that its action does *not* correspond to matrix multiplication using the definition (3.1).

3.2.2 Operators

We define the vector space of *operators* through linear combinations of dyads. That is,

$$\mathbb{O}^{m\times n\otimes p\times q} = \left\{ \sum_{i=1}^{r} \lambda_i \boldsymbol{X}_i \otimes \boldsymbol{Y}_i : \boldsymbol{X}_i \in \mathbb{M}^{m\times n}, \boldsymbol{Y}_i \in \mathbb{M}^{p\times q}, \lambda_i \in \mathbb{R}, r \in \mathbb{N} \right\}.$$

When the number of dyads in an operator decomposition is irrelevant to the discussion we may, for instance, write

$$\mathcal{A} = \sum_i \boldsymbol{X}_i \otimes \boldsymbol{Y}_i,$$

but we emphasize that the sum is *always* finite.

The reason that we call these objects *operators* is because we can regard them as linear mappings over matrices. Let $\mathcal{A} = \sum_{i=1}^{r} \boldsymbol{X}_i \otimes \boldsymbol{Y}_i \in \mathbb{O}^{m\times n\otimes p\times q}$. By linearly extending the action of dyads (3.2) we see that $\mathcal{A}\colon \mathbb{M}^{n\times q} \to \mathbb{M}^{m\times p}$ is the linear map

$$\mathcal{A}(\boldsymbol{M}) := \sum_{i=1}^{r} (\boldsymbol{X}_i \otimes \boldsymbol{Y}_i)(\boldsymbol{M}) = \sum_{i=1}^{r} \boldsymbol{X}_i \boldsymbol{M} \boldsymbol{Y}_i^{\mathsf{t}}, \tag{3.3}$$

for every $\boldsymbol{M} \in \mathbb{M}^{n\times q}$.

As we saw in Section 1.3.3, this viewpoint allows us to treat operators as liftings of matrices. In particular, we have that

$$\boldsymbol{A} = \boldsymbol{X}\boldsymbol{Y}^{\mathsf{t}} \quad \text{if and only if} \quad \boldsymbol{A} = (\boldsymbol{X} \otimes \boldsymbol{Y})(\mathbf{I}_r) = \boldsymbol{X}\mathbf{I}_r\boldsymbol{Y}^{\mathsf{t}},$$

where r is the inner dimension of the matrix factorization. Therefore we can regard the matrix $\boldsymbol{A} = \boldsymbol{X}\boldsymbol{Y}^{\mathsf{t}}$ as a linear image of the rank-1 operator $\boldsymbol{X} \otimes \boldsymbol{Y}$. In particular, we can use the nuclear norm framework on *operators* to promote structure in *matrix* factorizations. This approach leads to novel applications of nuclear norms for structured matrix recovery.

In the remainder of this section, we introduce additional notation and properties of operators.

The standard basis. If the families $\{\boldsymbol{E}_{ij} : 1 \le i \le m, 1 \le j \le n\}$ and $\{\boldsymbol{E}_{kl} : 1 \le k \le p, 1 \le l \le q\}$ are the standard bases for $\mathbb{M}^{m\times n}$ and $\mathbb{M}^{p\times q}$, then the dyad $\boldsymbol{E}_{ij} \otimes \boldsymbol{E}_{kl}$ corresponds to the standard basis element $\mathcal{E}_{ijkl}$ of $\mathbb{O}^{m\times n\otimes p\times q}$.

Therefore we can represent any operator $\mathcal{A} \in \mathbb{O}^{m\times n\otimes p\times q}$ entrywise as the linear combination

$$\mathcal{A} = \sum_{i=1}^{m}\sum_{j=1}^{n}\sum_{k=1}^{p}\sum_{l=1}^{q} a_{ijkl}(\boldsymbol{E}_{ij}\otimes\boldsymbol{E}_{kl}). \tag{3.4}$$

The a_{ijkl} are the *entries* of the operator $\mathcal{A}$. Even though operators are rank-2 tensors, this shows how we may model 4-dimensional data through the use of operators.

Just as with matrices, we use colon notation to retrieve slices of operators. For instance, given an operator $\mathcal{A} \in \mathbb{O}^{m\times n\otimes p\times q}$ the slice $\boldsymbol{A}_{::kl} \in \mathbb{M}^{m\times n}$ is such that $(\boldsymbol{A}_{::kl})_{ij} = \mathcal{A}_{ijkl}$.

Matrix representations. In particular, we can combine the entrywise representation (3.4) with Definition 3.2.1 to explicitly write the operator $\mathcal{A}$ as a matrix

$$\begin{aligned}\mathcal{A} &= \sum_{i=1}^{m}\sum_{j=1}^{n}\sum_{k=1}^{p}\sum_{l=1}^{q} a_{ijkl}(\boldsymbol{E}_{ij}\otimes\boldsymbol{E}_{kl})\\ &= \sum_{i=1}^{m}\sum_{j=1}^{n}\sum_{k=1}^{p}\sum_{l=1}^{q} a_{ijkl}\operatorname{vec}(\boldsymbol{E}_{ij})\operatorname{vec}(\boldsymbol{E}_{kl})^{\mathsf{t}} \in \mathbb{M}^{mn\times pq}.\end{aligned} \tag{3.5}$$

When we wish to emphasize that we are treating the operator $\mathcal{A}$ as a matrix, we use the notation $\operatorname{mat}(\mathcal{A})$.

Alternatively, for any dyadic decomposition $\mathcal{A} = \sum_i \boldsymbol{X}_i \otimes \boldsymbol{Y}_i$, we can write

$$\operatorname{mat}(\mathcal{A}) = \sum_i \operatorname{vec}(\boldsymbol{X}_i)\operatorname{vec}(\boldsymbol{Y}_i)^{\mathsf{t}}. \tag{3.6}$$

This representation always exists, as we can always find a dyadic decomposition of $\mathcal{A}$ using (3.4).

Again, this particular representation of objects in the tensor product space $\mathbb{M}^{m\times n}\otimes\mathbb{M}^{p\times q}$ as matrices in $\mathbb{M}^{mn\times pq}$ is the Choi-Jamiołkowski isomorphism. Furthermore, note that this matrix representation (3.6) of an operator is *not* used to compute the action of the operator (3.3). The Kronecker product may be used for this computation instead, and it is again related to (3.6) through an isomorphism.

Dyadic SVD. A useful decomposition of $\mathcal{A}$ comes from the singular value decomposition (SVD) of its matrix representation (3.6). Let $\operatorname{mat}(\mathcal{A}) = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^{\mathsf{t}}$ be a compact SVD. Let $\boldsymbol{U}_i$ be the ith column of $\boldsymbol{U}$ reshaped (columnwise) as an

$m \times n$ matrix, and let $\boldsymbol{V}_i$ be a similar reshaping of the ith column of $\boldsymbol{V}$ into a $p \times q$ matrix. Then the *dyadic SVD* of $\mathcal{A}$ is

$$\mathcal{A} = \sum_{i=1}^{r} \sigma_i \boldsymbol{U}_i \otimes \boldsymbol{V}_i, \tag{3.7}$$

where σ_i is the ith diagonal entry of $\boldsymbol{\Sigma}$ and r is the rank of $\operatorname{mat}(\mathcal{A})$.

Note that this also appears in the literature as the Kronecker product SVD (KP-SVD) [Vanoo].

The inner product. The inner product on operators is

$$\langle \mathcal{A}, \mathcal{B} \rangle = \sum_{i=1}^{m} \sum_{j=1}^{n} \sum_{k=1}^{p} \sum_{l=1}^{q} a_{ijkl} b_{ijkl},$$

where the a_{ijkl} and b_{ijkl} are the entries of the operators. Note that this definition corresponds exactly to the usual inner product of the matrix representations (3.5) of $\mathcal{A}$ and $\mathcal{B}$.

Also useful is the inner product identity

$$\langle \boldsymbol{X} \otimes \boldsymbol{Y}, \boldsymbol{X}' \otimes \boldsymbol{Y}' \rangle = \langle \boldsymbol{X}, \boldsymbol{X}' \rangle \langle \boldsymbol{Y}, \boldsymbol{Y}' \rangle. \tag{3.8}$$

Operator rank. Finally, the rank of an operator is the smallest number of dyads that may be used in any of its decompositions. That is,

$$\operatorname{rank}(\mathcal{A}) = \min \left\{ r : \mathcal{A} = \sum_{i=1}^{r} \boldsymbol{X}_i \otimes \boldsymbol{Y}_i \right\}.$$

This corresponds to the matrix rank of $\operatorname{mat}(\mathcal{A})$.

3.3 The nuclear norm

We desire a complexity measure for operators satisfying the following:

- It should be a convex function to facilitate its minimization.
- The complexity of an operator should grow linearly with respect to its absolute scale.
- It should account for the scale and complexity of each dyad used to construct the operator.

- Each additional dyad used to construct the operator should contribute as much as possible to the complexity.
- The complexity must only penalize the best possible decomposition of the operator.

We now show how we create such a measurement and discuss some of its geometric properties.

3.3.1 Crossnorms

The above criteria suggest that we must have a sensible way to measure the complexity of individual dyads. We choose to do this through the use of *crossnorms*.

Definition 3.3.1 (Crossnorm)**.** Consider two normed spaces

$$X = (\mathbb{M}^{m\times n}, \|\cdot\|_X) \quad \text{and} \quad Y = (\mathbb{M}^{p\times q}, \|\cdot\|_Y).$$

The norm $|||\cdot|||$ on $\mathbb{O}^{m\times n\otimes p\times q}$ is an $X \otimes Y$ *crossnorm* if

$$|||\boldsymbol{X}\otimes\boldsymbol{Y}||| = \|\boldsymbol{X}\|_X\|\boldsymbol{Y}\|_Y,$$

for every dyad $\boldsymbol{X}\otimes\boldsymbol{Y}$ with $\boldsymbol{X}\in X$ and $\boldsymbol{Y}\in Y$.

Note that the crossnorm property applies to norms on the space of operators, but the definition only concerns their action on dyads. When applied to dyads, crossnorms equally account for the complexity of each factor. As norms, they are necessarily positive definite and absolutely homogenous. We can easily confirm these properties hold on dyads.

Additionally, crossnorms must satisfy the triangle inequality. Thus for any operator with the decomposition $\mathcal{A} = \sum_i \boldsymbol{X}_i\otimes\boldsymbol{Y}_i$ we must have that

$$|||\mathcal{A}||| = \left|\left|\left| \sum_i \boldsymbol{X}_i\otimes\boldsymbol{Y}_i \right|\right|\right| \leq \sum_i |||\boldsymbol{X}_i\otimes\boldsymbol{Y}_i||| = \sum_i \|\boldsymbol{X}_i\|_X\|\boldsymbol{Y}_i\|_Y.$$

The crossnorm must satisfy this inequality for every possible decomposition of $\mathcal{A}$. Therefore we conclude that

$$|||\mathcal{A}||| \leq \inf\left\{ \sum_i \|\boldsymbol{X}_i\|_X\|\boldsymbol{Y}_i\|_Y : \mathcal{A} = \sum_i \boldsymbol{X}_i\otimes\boldsymbol{Y}_i \right\}, \tag{3.9}$$

where the infimum ranges over all decompositions of $\mathcal{A}$.

3.3.2 Nuclear norms

Since we desire a crossnorm on operators that penalizes additional dyads as much as possible, it seems natural to choose the largest one. As all crossnorms must satisfy the inequality (3.9), we use it to define our complexity measure: the nuclear norm.

Definition 3.3.2 (Nuclear norm)**.** Given two normed spaces

$$X = (\mathbb{M}^{m\times n}, \|\cdot\|_X) \quad \text{and} \quad Y = (\mathbb{M}^{p\times q}, \|\cdot\|_Y),$$

we define $N_{X,Y}$, the *nuclear norm* on $X \otimes Y$, as

$$N_{X,Y}(\mathcal{A}) := \inf\left\{ \sum_i \|\boldsymbol{X}_i\|_X \|\boldsymbol{Y}_i\|_Y : \mathcal{A} = \sum_i \boldsymbol{X}_i \otimes \boldsymbol{Y}_i \right\}. \tag{3.10}$$

Alternatively, we have

$$N_{X,Y}(\mathcal{A}) := \inf\left\{ \sum_i |\lambda_i| : \mathcal{A} = \sum_i \lambda_i \boldsymbol{X}_i \otimes \boldsymbol{Y}_i,\ \|\boldsymbol{X}_i\|_X = 1,\ \|\boldsymbol{Y}_i\|_Y = 1 \right\}. \tag{3.11}$$

Both infima are taken over all dyadic decompositions of $\mathcal{A}$.

This is exactly the definition we provided in Section 1.3.4. As we expect, the nuclear norm is not only a norm but a crossnorm too.

Proposition 3.3.3 (Properties of the nuclear norm)**.** *Let $X = (\mathbb{M}^{m\times n}, \|\cdot\|_X)$ and $Y = (\mathbb{M}^{p\times q}, \|\cdot\|_Y)$ be normed vector spaces. The nuclear norm $N_{X,Y}$ satisfies the following.*

1. *The nuclear norm $N_{X,Y}$ is a norm on $\mathbb{O}^{m\times n\otimes p\times q}$.*
2. *$N_{X,Y}$ is a crossnorm.*
3. *$N_{X,Y}$ dominates all other crossnorms on $X \otimes Y$ uniformly.*

We prove these results in Section A.1 of the appendix.

We interpret the nuclear norm as measuring the cost of constructing an operator through its dyadic decompositions. Each dyad $\boldsymbol{X}_i \otimes \boldsymbol{Y}_i$ of the optimal decomposition in the infimum (3.10) contributes the associated "cost" $\|\boldsymbol{X}_i\|_X \|\boldsymbol{Y}_i\|_Y$.

Dyads with low complexity as measured by the X and Y norms, therefore, contribute less to the nuclear norm. Operators with low nuclear norm are precisely the operators that have decompositions using few, structured dyads.

Given the importance of the optimal dyadic decomposition in this interpretation, we wish to know what we can say about its existence and general form.

Proposition 3.3.4 (Optimal decompositions)**.** *Let* $X = (\mathbb{M}^{m\times n}, \|\cdot\|_X)$ *and* $Y = (\mathbb{M}^{p\times q}, \|\cdot\|_Y)$ *be normed vector spaces. The nuclear norm* $N_{X,Y}$ *satisfies the following.*

1. *The infima in Definition 3.3.2 of* $N_{X,Y}$ *are attained.*
2. *The number of dyads at such an optimal decomposition is no more than* $mnpq$*.*

We provide the proof of these results in Section A.2 of the appendix. Note, however, that for some nuclear norms $N_{X,Y}$ the bound on the number of dyads required for an optimal decomposition can be improved. In particular, refer to the calculation of the $\ell_2 \otimes \ell_2$ nuclear norm (Section 3.4) and nuclear norms involving ℓ_1 (Section 3.5).

While we will refer back often to Definition 3.3.2, we now turn our attention to a geometric construction of the nuclear norm. The reader may find this alternative viewpoint more intuitive.

3.3.3 The unit ball

In the previous section we derived the definition of the nuclear norm from a list of requisites and the notion of crossnorms. Here we derive an equivalent definition through the construction of the nuclear unit ball. We adopt the factor spaces $X = (\mathbb{M}^{m\times n}, \|\cdot\|_X)$ and $Y = (\mathbb{M}^{p\times q}, \|\cdot\|_Y)$ as above. It is natural for us to ask that dyads with unit norm factors also have unit norm. So let us define

$$D_{X\otimes Y} := \{\boldsymbol{X} \otimes \boldsymbol{Y} : \|\boldsymbol{X}\|_X = 1, \|\boldsymbol{Y}\|_Y = 1\}, \tag{3.12}$$

the set of all dyads in $X \otimes Y$ with unit norm factors. For these dyads to have unit nuclear norm, we require that they lie on the boundary of the nuclear norm's unit ball.

Additionally, we seek the smallest unit ball meeting this requirement; the smallest unit ball will serve to penalize dyads as much as possible while retaining our notion of unit-norm dyads. The unit ball of a norm must be absolutely convex, and the smallest such set containing $D_{X\otimes Y}$ is its absolutely convex hull.

Definition 3.3.5 (Nuclear unit ball)**.** Let $X = (\mathbb{M}^{m\times n}, \|\cdot\|_X)$ and $Y = (\mathbb{M}^{p\times q}, \|\cdot\|_Y)$ be normed vector spaces. Define the nuclear unit ball $S_{X\otimes Y}$ as

$$S_{X\otimes Y} := \operatorname{abs\,conv} D_{X\otimes Y} = \left\{ \sum_i \lambda_i \boldsymbol{X}_i \otimes \boldsymbol{Y}_i : \boldsymbol{X}_i \otimes \boldsymbol{Y}_i \in D_{X,Y}, \sum_i |\lambda_i| \leq 1 \right\}. \quad (3.13)$$

Since the set $D_{X\otimes Y}$ is symmetric itself, its absolutely convex hull coincides with its convex hull, and we can instead write

$$S_{X\otimes Y} = \operatorname{conv} D_{X\otimes Y} = \left\{ \sum_i \lambda_i \boldsymbol{X}_i \otimes \boldsymbol{Y}_i : \boldsymbol{X}_i \otimes \boldsymbol{Y}_i \in D_{X,Y}, \sum_i \lambda_i = 1, \lambda_i \geq 0 \right\}. \quad (3.14)$$

We claim that the nuclear unit ball $S_{X\otimes Y}$ corresponds exactly with the set $\{\mathcal{A} : N_{X,Y}(\mathcal{A}) \leq 1\}$.

Proposition 3.3.6. *For normed spaces X and Y as in Definition 3.3.5, the nuclear unit ball (3.13) coincides with the unit ball of the nuclear norm $N_{X,Y}$ in Definition 3.3.2. That is,*

$$\mathcal{A} \in S_{X\otimes Y} \iff N_{X,Y} \leq 1.$$

Proof. For every $\mathcal{A} \in S_{X\otimes Y}$, we have that

$$\mathcal{A} = \sum_i \lambda_i \boldsymbol{X}_i \otimes \boldsymbol{Y}_i,$$

where $\sum_i |\lambda_i| \leq 1$, and $\|\boldsymbol{X}_i\|_X = \|\boldsymbol{Y}_i\|_Y = 1$. By Definition 3.3.2 of $N_{X,Y}$, we immediately have that

$$N_{X,Y}(\mathcal{A}) \leq \sum_i |\lambda_i| = 1.$$

If, instead, we assume that $N_{X,Y}(\mathcal{A}) \leq 1$ then by Proposition 3.3.4 there exists an optimal decomposition

$$\mathcal{A} = \sum_i \lambda_i \boldsymbol{X}_i \otimes \boldsymbol{Y}_i,$$

with $\|\boldsymbol{X}_i\|_X = \|\boldsymbol{Y}_i\|_Y = 1$, and $\sum_i |\lambda_i| = N_{X,Y}(\mathcal{A}) \leq 1$. Therefore $\mathcal{A} \in S_{X\otimes Y}$. □

3.3.4 Dual norms

Using the unit ball of the nuclear norm on $X \otimes Y$, we can define the dual norm.

Definition 3.3.7 (Dual norm)**.** Let $X = (\mathbb{M}^{m\times n}, \|\cdot\|_X)$ and $Y = (\mathbb{M}^{p\times q}, \|\cdot\|_Y)$ be normed vector spaces. The dual norm $N^*_{X,Y}$ is given by

$$N^*_{X,Y}(\mathcal{B}) := \max\{\langle \mathcal{B}, \mathcal{A}\rangle : \mathcal{A} \in S_{X\otimes Y}\}, \tag{3.15}$$

for every $\mathcal{B} \in \mathbb{O}^{m\times n\otimes p\times q}$.

The dual norm proves useful subsequently when we calculate particular nuclear norms.

Proposition 3.3.8 (The dual of the nuclear norm)**.** *Let $X = (\mathbb{M}^{m\times n}, \|\cdot\|_X)$ and $Y = (\mathbb{M}^{p\times q}, \|\cdot\|_Y)$ be normed vector spaces. The dual norm $N^*_{X,Y}$ is given by*

$$N^*_{X,Y}(\mathcal{B}) = \max\{\langle \mathcal{B}, \boldsymbol{X}\otimes\boldsymbol{Y}\rangle : \|\boldsymbol{X}\|_X = 1, \|\boldsymbol{Y}\|_Y = 1\}. \tag{3.16}$$

By using the correspondence between operators and matrices, we can also express the dual norm as follows.

$$N^*_{X,Y}(\mathcal{B}) = \max\{\langle \operatorname{vec}\boldsymbol{Y}, \boldsymbol{B}^{\mathsf{t}}\operatorname{vec}\boldsymbol{X}\rangle : \|\boldsymbol{X}\|_X = 1, \|\boldsymbol{Y}\|_Y = 1\}, \tag{3.17}$$

where $\boldsymbol{B} = \operatorname{mat}(\mathcal{B})$, the matrix representation (3.6) of $\mathcal{B}$.

Proof. The dual norm (3.15) is the maximum of a linear functional over the compact, convex set $S_{X\otimes Y}$. This maximum must occur at an extreme point of the set. Recall, however, that $S_{X\otimes Y}$ is the convex hull of the $D_{X\otimes Y}$, the set of unit-norm dyads. Therefore the extreme points of $S_{X\otimes Y}$ must reside in $D_{X\otimes Y}$, and we can let the maximum range over $D_{X\otimes Y}$ instead. This results in the expression (3.16).

For any dyadic decomposition of $\mathcal{B}$, let $\boldsymbol{B} = \operatorname{mat}(\mathcal{B})$ be its matrix representation (3.6). Applying standard inner product rules gives (3.17). □

3.3.5 Connections to sparse approximation

A straightforward modification of (3.11) in Definition 3.3.2 of the nuclear norm on $X \otimes Y$ gives that

$$N_{X,Y}(\mathcal{A}) = \inf\left\{\|\boldsymbol{\lambda}\|_{\ell_1} : \mathcal{A} = \sum_i \lambda_i \boldsymbol{X}_i\otimes\boldsymbol{Y}_i, \boldsymbol{X}_i\otimes\boldsymbol{Y}_i \in D_{X\otimes Y}\right\}, \tag{3.18}$$

Norm	Structure
ℓ_1	1-sparse vector
ℓ_∞	Sign vector (± 1 entries)
S_1	Rank-1 matrix
S_∞	Orthogonal matrix

Table 3.1: **Examples of atomic norms.** This table lists familiar vector and matrix norms that arise as complexity measurements on atomic decompositions. The structure column indicates the atoms composing these decompositions.

where $D_{X\otimes Y}$ is once again the set of unit-norm dyads (3.12).

We can think of the nuclear norm as building $\mathcal{A}$ from linear combinations of unit-norm dyads and finding the decomposition with the smallest absolute sum of coefficients. In the language of sparse approximation, we think of $D_{X\otimes Y}$ as a *dictionary* containing the basic building blocks of operators. These entries in the dictionary, called *atoms*, are structured rank-1 operators.

The principle of finding an atomic decomposition of a signal through ℓ_1-minimization is known as *basis pursuit* and was introduced by Chen, Donoho, and Saunders [CDS98]. The effect of ℓ_1-minimization is to promote sparse decompositions. We therefore interpret the nuclear norm as finding a sparse representation of $\mathcal{A}$ from the dictionary $D_{X\otimes Y}$. That is, it seeks a low-rank decomposition for $\mathcal{A}$ comprising structured dyads.

The nuclear norm of an operator (3.18) is the optimal *value* of this optimization problem. Over all decompositions of the operator into linear combinations of dictionary elements, the nuclear norm is the lowest possible absolute sum of the coefficients. In this way, we can think of the nuclear norm as representing the "cost" of constructing an operator from a given set of dictionary elements. We consider operators that are themselves sums of few atoms as having lower cost, and therefore, lower nuclear norms. The idea of using the nuclear norm as a complexity measure exactly corresponds with the intuition that "simpler" operators cost less to construct. This idea of creating a norm-like complexity measure based on atomic decompositions from dictionaries appears in the work of DeVore and Temlyakov [DT96] on functional approximation.

Many common norms appearing in the sparse approximation literature have similar interpretations as complexity measures on atomic decompositions. Table 3.1 lists several of these norms and their associated structured atoms. By constructing nuclear norms from such norms, we can promote operators

having desirable factor structure. For instance, we expect the $\ell_\infty \otimes S_\infty$ nuclear norm to promote dyads where the left factor is a sign matrix and the right factor is an orthogonal matrix.

This idea of encoding prior assumptions of factor structure into the nuclear norm is central to our framework. For instance, we conduct numerical experiments in Chapter 5 to measure the performance of various nuclear norms as regularizers for denoising structured low-rank operators. We demonstrate that matching the nuclear norm to the factor structure of the true operator provides the best denoising performance. In particular, denoising with a well-chosen nuclear norm outperforms standard trace-norm minimization.

3.3.6 The nuclear norm as an atomic norm

Since $S_{X\otimes Y}$ is the convex hull of $D_{X\otimes Y}$, see (3.14), the nuclear norm is the gauge function of a convex hull:

$$N_{X,Y}(\mathcal{A}) = \inf\{t > 0 : \mathcal{A} \in t \cdot S_{X\otimes Y}\}.$$

The nuclear norm, therefore, falls into the *atomic norm framework* of Chandrasekaran et al. [CRPW12]. The convex hull of the dictionary $D_{X\otimes Y}$ provides the tightest convex superset of the dictionary, and defining *atomic norms* as gauges of those hulls creates complexity measures that maximally penalize deviation away from convex combinations of dictionary elements. That is, operators with low atomic norm arise as superpositions of few structured atoms.

3.3.7 The nuclear norm recovery problem

In Chapter 1 we motivated our discussion of nuclear norms using the problem of approximating operators from noisy and incomplete linear measurements. This culminated in the nuclear norm recovery problem (1.11) that we restate here.

Let $\mathcal{A}^\natural \in \mathbb{O}^{m\times n\otimes p\times q}$ be an operator and $\boldsymbol{\mu}\colon \mathbb{O}^{m\times n\otimes p\times q} \to \mathbb{R}^s$ be a linear measurement map. Assume that we observe $\boldsymbol{b} = \boldsymbol{\mu}(\mathcal{A}^\natural) + \boldsymbol{z}$, where $\boldsymbol{z} \in \mathbb{R}^s$ is additive noise. We propose to estimate $\mathcal{A}^\natural$ given $\boldsymbol{b}$ by solving the problem

$$\underset{\mathcal{A}}{\text{minimize}} \quad \frac{1}{2}\|\boldsymbol{b} - \boldsymbol{\mu}(\mathcal{A})\|_{\ell_2}^2 + \lambda N_{X,Y}(\mathcal{A}),$$

where $\lambda > 0$ is a parameter that balances the relative importance of maintaining fidelity to the measurements $\boldsymbol{b}$ and promoting solutions with low nuclear norm.

Our interpretations of the nuclear norm in Sections 3.3.5 and 3.3.6 suggest that using the nuclear norm on $X \otimes Y$ as a regularizer in (1.11) will promote solutions that have sparse representations in the dictionary $D_{X\otimes Y}$. Therefore, our choice to use any particular nuclear norm $N_{X,Y}$ in (1.11) should be based on the assumption that $\mathcal{A}^\natural$ admits a sparse atomic decomposition in the dictionary $D_{X\otimes Y}$.

3.3.8 Computation

While the nuclear norm appears to be a useful tool in recovering structured low-rank operators, we face computational difficulties. Simply put, we have very few instances where we can directly compute the nuclear norm of an operator (and we address these in the following two sections). In other cases, we can show that computing the nuclear norm is, in fact, intractable.

Even though this situation seems dire, we do have some reason for optimism. First, some difficult nuclear norms can be well approximated by semidefinite programs. We consider this scenario in Section 3.6. Fundamental results from functional analysis can also characterize the quality of these approximations.

Second, we can draw on techniques from nonconvex optimization to compute more general nuclear norms. Techniques such as alternating minimization and gradient descent, discussed in Section 2.2, have had success in various matrix factorization problems. We detail our preferred approach—alternating minimization—more thoroughly in Section 4.2.

While we utilize nonconvex optimization in our own software, we emphasize that the underlying nuclear norm minimization problems we wish to solve are truly convex. This provides us some solace even when we must employ heuristic approaches in their solution.

3.4 The trace norm

As we stated in Section 1.2.2.1, the trace norm is in fact the $\ell_2 \otimes \ell_2$ nuclear norm. Here we restate and prove this same result for operators. First, however, we review some implications of the dyadic SVD (3.7).

Consider the operator $\mathcal{A} \in \mathbb{O}^{m\times n\otimes p\times q}$ and its matrix representation $\boldsymbol{A} = \operatorname{mat}(\mathcal{A})$ from (3.6). We can compute the dyadic SVD as in (3.7),

$$\mathcal{A} = \sum_{i=1}^{r} \sigma_i \boldsymbol{U}_i \otimes \boldsymbol{V}_i.$$

We can also take the trace norm of the matrix $\|\boldsymbol{A}\|_{S_1}$ as we normally would

$$\|\boldsymbol{A}\|_{S_1} = \sum_{i=1}^{r} \sigma_i.$$

Note that the decomposition (and hence the singular values) correspond to the *matrix* $\boldsymbol{A}$; we make no statement here about the singular values of an operator. Instead we call the σ_i dyadic singular values.

Given the properties of the SVD, we see that the $\boldsymbol{U}_i$ and $\boldsymbol{V}_i$ also have unit Euclidean norm. For matrices this is often referred to as the Frobenius norm (written as $\|\cdot\|_{\mathrm{F}}$), but we simply refer to it as the ℓ_2 norm. We now prove that the $\ell_2 \otimes \ell_2$ nuclear norm is indeed the trace norm. This result applies to the tensor product of vectors as well as the tensor product of matrices shown here.

Proposition 3.4.1 (The trace norm)**.** *Let $\mathcal{A} \in \mathbb{O}^{m\times n\otimes p\times q}$ have the dyadic SVD*

$$\mathcal{A} = \sum_{i=1}^{r} \sigma_i \boldsymbol{U}_i \otimes \boldsymbol{V}_i.$$

Then the $\ell_2 \otimes \ell_2$ nuclear norm, N_{ℓ_2,ℓ_2}, is given by the sum of the dyadic singular values

$$N_{\ell_2,\ell_2}(\mathcal{A}) = \sum_{i=1}^{r} \sigma_i.$$

Furthermore, the dual norm is

$$N^*_{\ell_2,\ell_2}(\mathcal{B}) = \sigma_1(\mathcal{B}),$$

the largest dyadic singular value of $\mathcal{B}$.

Proof. For any operator $\mathcal{B}$ in $\mathbb{O}^{m\times n\otimes p\times q}$, let $\boldsymbol{B}$ be its matrix representation (3.6) in $\mathbb{M}^{mn\times pq}$. By the definition (3.17) of the dual norm, we have that

$$N^*_{\ell_2,\ell_2}(\mathcal{B}) = \max\{\langle \operatorname{vec} \boldsymbol{Y}, \boldsymbol{B}^{\mathsf{t}} \operatorname{vec} \boldsymbol{X}\rangle : \|\boldsymbol{X}\|_{\ell_2} = \|\boldsymbol{Y}\|_{\ell_2} = 1\},$$

where the maximum is taken over $\boldsymbol{X} \in \mathbb{M}^{m\times n}$ and $\boldsymbol{Y} \in \mathbb{M}^{p\times q}$.

The maximum itself is the largest singular value of $\boldsymbol{B}$, and so

$$N^*_{\ell_2,\ell_2}(\mathcal{B}) = \sigma_1(\boldsymbol{B}) = \sigma_1(\mathcal{B}),$$

with the abuse of notation that $\sigma_1(\mathcal{B})$ returns the largest dyadic singular value of the operator $\mathcal{B}$.

Since the nuclear norm N_{ℓ_2,ℓ_2} is both a norm and a crossnorm, we can apply the triangle inequality to see that

$$N_{\ell_2,\ell_2}(\mathcal{A}) \leq \sum_{i=1}^{r} \sigma_i \|\boldsymbol{U}_i\|_{\ell_2} \|\boldsymbol{V}_i\|_{\ell_2} = \sum_{i=1}^{r} \sigma_i.$$

However, we also have that

$$\begin{aligned} N_{\ell_2,\ell_2}(\mathcal{A}) &= \max\{\langle \mathcal{A}, \mathcal{B}\rangle : N^*_{\ell_2,\ell_2}(\mathcal{B}) \leq 1\} \\ &= \max\{\langle \boldsymbol{A}, \boldsymbol{B}\rangle : \sigma_1(\boldsymbol{B}) \leq 1\} \\ &\geq \langle \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^{\mathsf{t}}, \boldsymbol{U}\boldsymbol{V}^{\mathsf{t}}\rangle \\ &= \operatorname{tr}(\boldsymbol{V}\boldsymbol{\Sigma}^{\mathsf{t}}\boldsymbol{U}^{\mathsf{t}}\boldsymbol{Y}\boldsymbol{V}^{\mathsf{t}}) = \operatorname{tr}(\boldsymbol{\Sigma}) = \sum_{i=1}^{r} \sigma_i, \end{aligned}$$

where the ith column of $\boldsymbol{U}$ is $\operatorname{vec}(\boldsymbol{U}_i)$, the ith column of $\boldsymbol{V}$ is $\operatorname{vec}(\boldsymbol{V}_i)$, and the diagonal matrix $\boldsymbol{\Sigma}$ has the σ_i as its entries. The inequality holds by choosing $\boldsymbol{B} = \boldsymbol{U}\boldsymbol{V}^{\mathsf{t}}$ and applying properties of the trace. □

In particular, this result still holds for the $\ell_2 \otimes \ell_2$ nuclear norm in the vector case.

3.5 Nuclear norms involving ℓ_1

We also have a closed form expression for the nuclear norm when equipping either of the factor spaces with the ℓ_1 norm. If, for instance, we measure the right factors with the ℓ_1 norm, then the $X \otimes \ell_1$ nuclear norm is the sum of the X norms of the "left slices". Similarly, the $\ell_1 \otimes Y$ is simply the sum of the Y norms of the "right slices".

Proposition 3.5.1. *Let $\mathcal{A} \in \mathbb{O}^{m\times n\otimes p\times q}$ be an operator. In the space $X \otimes \ell_1$, we have the nuclear norm*

$$N_{X,\ell_1}(\mathcal{A}) = \sum_{k=1}^{p}\sum_{l=1}^{q} \|\boldsymbol{A}_{::kl}\|_X,$$

and the operator $\mathcal{A}$ has the optimal decomposition

$$\mathcal{A} = \sum_{k=1}^{p}\sum_{l=1}^{q} \boldsymbol{A}_{::kl} \otimes \boldsymbol{E}_{kl}.$$

Furthermore, the dual norm is

$$N^*_{X,\ell_1}(\mathcal{B}) = \max_{k,l} \|\boldsymbol{B}_{::kl}\|_{X^*}.$$

Proof. We start with computing the dual norm

$$\begin{aligned}
N^*(\mathcal{B}) &= \max\{\langle \mathcal{B}, \boldsymbol{X} \otimes \boldsymbol{Y}\rangle : \|\boldsymbol{X}\|_X = 1; \|\boldsymbol{Y}\|_{\ell_1} = 1\} \\
&= \max\{|\langle \mathcal{B}, \boldsymbol{X} \otimes \boldsymbol{E}_{kl}\rangle| : \|\boldsymbol{X}\|_X = 1; k = 1, \ldots, p; l = 1, \ldots, q\} \\
&= \max\{|\langle \boldsymbol{B}_{::kl}, \boldsymbol{X}\rangle| : \|\boldsymbol{X}\|_X = 1; k = 1, \ldots, p; l = 1, \ldots, q\} \\
&= \max_{k,l} \|\boldsymbol{B}_{::kl}\|_{X^*},
\end{aligned}$$

where the second equality follows since the maximum over the ℓ_1-norm ball occurs at $\pm\boldsymbol{E}_{kl}$ for some k and l, and the third is an application of the inner product identity (3.8).

Since the nuclear norm is a crossnorm, we must have that

$$N(\mathcal{A}) \le \sum_{k=1}^{p}\sum_{l=1}^{q} \|\boldsymbol{A}_{::kl}\|_X \|\boldsymbol{E}_{kl}\|_{\ell_1} = \sum_{k=1}^{p}\sum_{l=1}^{q} \|\boldsymbol{A}_{::kl}\|_X.$$

On the other hand,

$$\begin{aligned}
N(\mathcal{A}) &= \max\{\langle \mathcal{A}, \mathcal{B}\rangle : N^*(\mathcal{B}) \le 1\} \\
&\ge \max\left\{\left\langle \sum_{k=1}^{p}\sum_{l=1}^{q} \boldsymbol{A}_{::kl} \otimes \boldsymbol{E}_{kl}, \sum_{k'=1}^{p}\sum_{l'=1}^{q} \boldsymbol{B}_{::k'l'} \otimes \boldsymbol{E}_{k'l'} \right\rangle : \|\boldsymbol{B}_{::k'l'}\|_{X^*} \le 1\right\} \\
&= \max\left\{\sum_{k=1}^{p}\sum_{l=1}^{q} \langle \boldsymbol{A}_{::kl}, \boldsymbol{B}_{::kl}\rangle : \|\boldsymbol{B}_{::kl}\|_{X^*} \le 1 \text{ for each } k, l\right\} \\
&= \sum_{k=1}^{p}\sum_{l=1}^{q} \|\boldsymbol{A}_{::kl}\|_X,
\end{aligned}$$

where the inequality holds by choosing a particular $\mathcal{B}$, and the last equality holds because the optimal $\boldsymbol{B}_{::kl}$s will norm the $\boldsymbol{A}_{::kl}$s. □

3.6 Semidefinite relaxations

While the trace norm and nuclear norms involving ℓ_1 have easily computable forms (along with optimal decompositions), many other interesting norms do not. In this section we describe how some nuclear norms may be well-approximated by solving semidefinite programs.

3.6.1 An alternative nuclear norm formulation

To arrive at the semidefinite relaxations, we make use of an additional formulation for the nuclear norm.

Proposition 3.6.1 (Alternate definition of the nuclear norm)**.** *Let* $X = (\mathbb{M}^{m\times n}, \|\cdot\|_X)$ *and* $Y = (\mathbb{M}^{p\times q}, \|\cdot\|_Y)$ *be normed vector spaces. We can write the nuclear norm on* $X \otimes Y$ *as*

$$N_{X,Y}(\mathcal{A}) = \inf\left\{\frac{1}{2}\sum_i (\|\boldsymbol{X}_i\|_X^2 + \|\boldsymbol{Y}_i\|_Y^2) : \mathcal{A} = \sum_i \boldsymbol{X}_i \otimes \boldsymbol{Y}_i\right\}, \tag{3.19}$$

where the infimum is over all decompositions of $\mathcal{A}$*.*

Proof. Recall from Definition 3.3.2 that the nuclear norm on $X \otimes Y$ (3.10) is

$$N_{X,Y}(\mathcal{A}) = \inf\left\{\sum_i \|\boldsymbol{X}_i\|_X \|\boldsymbol{Y}_i\|_Y : \mathcal{A} = \sum_i \boldsymbol{X}_i \otimes \boldsymbol{Y}_i\right\}.$$

Using the bilinearity of dyad arithmetic, we can write

$$\boldsymbol{X}_i \otimes \boldsymbol{Y}_i = (t_i \boldsymbol{X}_i) \otimes (t_i^{-1} \boldsymbol{Y}_i),$$

for every $t_i > 0$. Through the arithmetic–geometric mean inequality, we have that

$$\|\boldsymbol{X}_i\|_X \|\boldsymbol{Y}_i\|_Y = \inf_{t_i>0} \frac{1}{2}\left(\|t_i \boldsymbol{X}_i\|_X^2 + \|t_i^{-1}\boldsymbol{Y}_i\|_Y^2\right).$$

Through the changes of variables $t_i \boldsymbol{X}_i \mapsto \boldsymbol{X}_i$ and $t_i^{-1}\boldsymbol{Y}_i \mapsto \boldsymbol{Y}_i$, we obtain the desired result. □

3.6.2 The semidefinite representation

The key to creating the relaxed nuclear norms is our ability to express operator decompositions through a semidefinite inequality.

Proposition 3.6.2 (Semidefinite representation of operator decomposition)**.** *Fix operators $\mathcal{A} \in \mathbb{O}^{m\times n\otimes p\times q}$, $\mathcal{W}_1 \in \mathbb{O}^{m\times n\otimes m\times n}$, and $\mathcal{W}_2 \in \mathbb{O}^{p\times q\otimes p\times q}$. Let $\boldsymbol{A} \in \mathbb{M}^{mn\times pq}$, $\boldsymbol{W}_1 \in \mathbb{M}^{mn\times mn}$, and $\boldsymbol{W}_2 \in \mathbb{M}^{pq\times pq}$ be their matrix representations* (3.6)*. The following conditions are equivalent.*

1. *There exist matrices $\boldsymbol{X} \in \mathbb{M}^{mn\times r}$ and $\boldsymbol{Y} \in \mathbb{M}^{pq\times r}$ for some $r \in \mathbb{N}$ such that*

$$\boldsymbol{A} = \boldsymbol{X}\boldsymbol{Y}^{\mathsf{t}}, \quad \boldsymbol{W}_1 = \boldsymbol{X}\boldsymbol{X}^{\mathsf{t}}, \quad \text{and} \quad \boldsymbol{W}_2 = \boldsymbol{Y}\boldsymbol{Y}^{\mathsf{t}}.$$

2. *The following semidefinite inequality holds.*

$$\begin{bmatrix} \boldsymbol{W}_1 & \boldsymbol{A} \\ \boldsymbol{A}^{\mathsf{t}} & \boldsymbol{W}_2 \end{bmatrix} \succeq \boldsymbol{0}. \tag{3.20}$$

In operator notation, condition (1) states that

$$\mathcal{A} = \sum_{i=1}^{r} \boldsymbol{X}_i \otimes \boldsymbol{Y}_i,$$

where the $\boldsymbol{X}_i \in \mathbb{M}^{m\times n}$ and $\boldsymbol{Y}_i \in \mathbb{M}^{p\times q}$ are the reshaped columns of $\boldsymbol{X}$ and $\boldsymbol{Y}$.

Additionally, the semidefinite inequality (3.20) *implies that condition (1) holds for some $r \leq mn + pq$.*

Proof. Assume condition (2) holds so that the block matrix has the positive square root

$$\boldsymbol{S} = \begin{bmatrix} \boldsymbol{W}_1 & \boldsymbol{A} \\ \boldsymbol{A}^{\mathsf{t}} & \boldsymbol{W}_2 \end{bmatrix}^{1/2} = \begin{bmatrix} \boldsymbol{X} \\ \boldsymbol{Y} \end{bmatrix}, \tag{3.21}$$

where the matrices $\boldsymbol{X} \in \mathbb{M}^{mn\times(mn+pq)}$ and $\boldsymbol{Y} \in \mathbb{M}^{pq\times(mn+pq)}$ result from grouping the rows of $\boldsymbol{S}$.

Then we have that

$$\begin{bmatrix} \boldsymbol{W}_1 & \boldsymbol{A} \\ \boldsymbol{A}^{\mathsf{t}} & \boldsymbol{W}_2 \end{bmatrix} = \begin{bmatrix} \boldsymbol{X} \\ \boldsymbol{Y} \end{bmatrix} \begin{bmatrix} \boldsymbol{X} \\ \boldsymbol{Y} \end{bmatrix}^{\mathsf{t}} = \begin{bmatrix} \boldsymbol{X}\boldsymbol{X}^{\mathsf{t}} & \boldsymbol{X}\boldsymbol{Y}^{\mathsf{t}} \\ \boldsymbol{Y}\boldsymbol{X}^{\mathsf{t}} & \boldsymbol{Y}\boldsymbol{Y}^{\mathsf{t}} \end{bmatrix},$$

and condition (1) follows by equating the blocks.

Conversely, if condition (1) holds, we can substitute $\boldsymbol{X}$ and $\boldsymbol{Y}$ directly into (3.20). By the above, the resulting block matrix is indeed positive semidefinite.

Finally, our procedure for generating $\boldsymbol{X}$ and $\boldsymbol{Y}$ from the positive square root shows that $r \leq mn + pq$ in condition (1). □

The import of this result is that finding a semidefinite block matrix with a prescribed $\boldsymbol{A}$ also determines an operator decomposition. We now explore how optimization problems involving the $\boldsymbol{W}_i$ in the block matrix can recover structured decompositions.

3.6.3 Example: The trace norm

Before looking at how this semidefinite representation may be applied more generally, we outline the method with the familiar example of the trace norm.

Proposition 3.6.3 (Semidefinite representation of the trace norm)**.** *For every operator $\mathcal{A} \in \mathbb{O}^{m\times n\otimes p\times q}$, we have that*

$$N_{\ell_2,\ell_2}(\mathcal{A}) = \inf\left\{\frac{1}{2}[\operatorname{tr}(\boldsymbol{W}_1) + \operatorname{tr}(\boldsymbol{W}_2)] : \begin{bmatrix} \boldsymbol{W}_1 & \boldsymbol{A} \\ \boldsymbol{A}^{\mathsf{t}} & \boldsymbol{W}_2 \end{bmatrix} \succeq \boldsymbol{0}\right\}, \tag{3.22}$$

where $\boldsymbol{A}$ is the matrix representation (3.6) *of $\mathcal{A}$, and the block matrix formulation is as in Proposition 3.6.2.*

Proof. Starting with the nuclear norm formulation (3.19), we replace the factorization constraint with the semidefinite inequality (3.21) from Proposition 3.6.2. And by Proposition 3.6.2, we have that $\boldsymbol{A} = \boldsymbol{X}\boldsymbol{Y}^{\mathsf{t}}$, $\boldsymbol{W}_1 = \boldsymbol{X}\boldsymbol{X}^{\mathsf{t}}$, and $\boldsymbol{W}_2 = \boldsymbol{Y}\boldsymbol{Y}^{\mathsf{t}}$ for some $\boldsymbol{X}$ and $\boldsymbol{Y}$.

We see that

$$\begin{aligned}\sum_j \|\boldsymbol{x}_{:j}\|_{\ell_2}^2 &= \sum_j \left(\sum_i |x_{ij}|^2\right) \\ &= \sum_i \left(\sum_j |x_{ij}|^2\right) = \sum_i \|\boldsymbol{x}_{i:}\|_{\ell_2}^2 = \sum_i (\boldsymbol{X}\boldsymbol{X}^{\mathsf{t}})_{ii} = \operatorname{tr}(\boldsymbol{W}_1),\end{aligned}$$

where $\boldsymbol{x}_{:i}$ is the ith column of the matrix $\boldsymbol{X}$.

Since we are regarding $\boldsymbol{A} = \boldsymbol{X}\boldsymbol{Y}^{\mathsf{t}}$ as the matrix representation (3.6) of the operator $\mathcal{A}$, the semidefinite block matrix results in the dyadic decomposition

$$\mathcal{A} = \sum_{i=1}^{r} \boldsymbol{X}_i \otimes \boldsymbol{Y}_i,$$

where $\boldsymbol{X}_i$ is the ith column of $\boldsymbol{X}$ reshaped into an $m \times n$ matrix. In particular, we have that $\|\boldsymbol{X}_i\|_{\ell_2} = \|\boldsymbol{x}_{:i}\|_{\ell_2}$. Therefore,

$$\sum_i \|\boldsymbol{X}_i\|_{\ell_2}^2 = \operatorname{tr}(\boldsymbol{W}_1).$$

A similar equality holds for the factors $\boldsymbol{Y}_i$ and $\operatorname{tr}(\boldsymbol{W}_2)$. We conclude that (3.22) is in fact the $\ell_2 \otimes \ell_2$ nuclear norm. □

The key idea here is that we may replace the factorization constraint in the formulation of the nuclear norm with a semidefinite constraint and express the factor norms in terms of the entries of $\boldsymbol{W}_1$ and $\boldsymbol{W}_2$. In particular, we move from considering the dyad factors $\boldsymbol{X}_i$ to the matrix $\boldsymbol{X}$ whose columns are the vectorized $\boldsymbol{X}_i$. Then we express the sum of the squared *column* norms of $\boldsymbol{X}$ as a function of the squared Euclidean *row* norms of $\boldsymbol{X}$. Here the function is simply the sum, and the statement holds with equality.

As we will see shortly, we can make a similar expression for other squared column norms, but the relationship holds with an inequality. Note that the equality between the squared norms of the factors $\boldsymbol{X}_i$ and the squared column norms of $\boldsymbol{X}$ exists when we norm the $\boldsymbol{X}_i$ with *vector* norms.

3.6.4 Superquadratic norms

To generalize the above we need to express the sum of the squared column norms of a matrix as a function of its squared Euclidean row norms. In this section we show how to achieve this for *superquadratic* norms.

Definition 3.6.4 (Superquadratic norm)**.** A vector norm $\|\cdot\|_X$ on $\mathbb{R}^m$ is *superquadratic* if there exists a *gauge* $g_X : \mathbb{R}^m_+ \to \mathbb{R}_+$ such that

$$\|\boldsymbol{x}\|_X^2 = g_X(|\boldsymbol{x}|^2),$$

for every $\boldsymbol{x} \in \mathbb{R}^m$. We use $|\cdot|^2$ to denote the componentwise squaring of a vector.

Many norms are, in fact, superquadratic.

Example 3.6.5 (The ℓ_2 norm is superquadratic)**.** Let the normed space $X = \ell_2^m$. Observe that for all $\boldsymbol{x} \in \mathbb{R}^m$,

$$\|\boldsymbol{x}\|_{\ell_2}^2 = \sum_{i=1}^m |x_i|^2,$$

and so the ℓ_2 norm is superquadratic with gauge

$$g_{\ell_2}(\boldsymbol{s}) = \sum_{i=1}^m |s_i| = \|\boldsymbol{s}\|_{\ell_1},$$

for all $\boldsymbol{s} \in \mathbb{R}^m_+$.

Example 3.6.6 (The ℓ_∞ norm is superquadratic)**.** Let the normed space $X = \ell_\infty^m$. Observe that for all $\boldsymbol{x} \in \mathbb{R}^m$,

$$\|\boldsymbol{x}\|_{\ell_\infty}^2 = \max_{i=1,\ldots,m} |x_i|^2,$$

and so the ℓ_∞ norm is superquadratic with gauge

$$g_{\ell_\infty}(\boldsymbol{s}) = \max_{i=1,\ldots,m} |s_i| = \|\boldsymbol{s}\|_{\ell_\infty},$$

for all $\boldsymbol{s} \in \mathbb{R}_+^m$.

Example 3.6.7 (The ℓ_p norm, $p \geq 2$, is superquadratic)**.** Let the normed space $X = \ell_p^m$ with $p \geq 2$. Observe that for all $\boldsymbol{x} \in \mathbb{R}^m$,

$$\|\boldsymbol{x}\|_{\ell_p}^2 = \left(\sum_{i=1}^m (|x_i|^2)^{p/2} \right)^{2/p},$$

and so the ℓ_p norm, with $p \geq 2$ is superquadratic with gauge

$$g_{\ell_p}(\boldsymbol{s}) = \left(\sum_{i=1}^m |s_i|^{p/2} \right)^{2/p} = \|\boldsymbol{s}\|_{\ell_{p/2}},$$

for all $\boldsymbol{s} \in \mathbb{R}_+^m$.

We will focus on the use of the ℓ_2 and ℓ_∞ norms, but we also see that the last result allows us to interpolate between them.

3.6.5 Relaxed nuclear norms

In this section we show how we can create semidefinite relaxations of nuclear norms when the factor spaces have superquadratic vector norms.

Definition 3.6.8 (The relaxed nuclear norm)**.** Let $\|\cdot\|_X$ on $\mathbb{R}^{mn}$ and $\|\cdot\|_Y$ on $\mathbb{R}^{pq}$ be superquadratic vector norms. For every $\mathcal{A} \in \mathbb{O}^{m\times n\otimes p\times q}$, we define the *relaxed nuclear norm* as

$$R_{X,Y}(\mathcal{A}) := \inf \left\{ \frac{1}{2} \left[g_X(\operatorname{diag}(\boldsymbol{W}_1)) + g_Y(\operatorname{diag}(\boldsymbol{W}_2)) \right] : \begin{bmatrix} \boldsymbol{W}_1 & \boldsymbol{A} \\ \boldsymbol{A}^{\mathsf{t}} & \boldsymbol{W}_2 \end{bmatrix} \geq \boldsymbol{0} \right\},$$

where $\boldsymbol{A}$ is the matrix representation (3.6) of $\mathcal{A}$, and the block matrix has dimensions as in (3.20).

The relaxed norm $R_{X,Y}$ is thus the solution to a convex program, and it is computationally tractable whenever the gauges g_X and g_Y are themselves tractable.

Furthermore, note that this definition agrees with the trace norm example (3.22) in Proposition 3.6.3.

We now show that $R_{X,Y}$ is indeed a relaxation of $N_{X,Y}$.

Proposition 3.6.9 (Relaxation)**.** *Let $\|\cdot\|_X$ on $\mathbb{R}^{mn}$ and $\|\cdot\|_Y$ on $\mathbb{R}^{pq}$ be superquadratic vector norms. The relaxed norm $R_{X,Y}$ satisfies*

$$R_{X,Y}(\mathcal{A}) \leq N_{X,Y}(\mathcal{A}),$$

for all $\mathcal{A} \in \mathbb{O}^{m\times n\otimes p\times q}$.

Proof. We start with the formulation of the nuclear norm given in Proposition 3.6.1 and replace the decomposition constraint with the semidefinite constraint from Proposition 3.6.2. To complete the proof we must verify that the objective

$$\frac{1}{2}\left[g_X(\operatorname{diag}(\boldsymbol{W}_1)) + g_Y(\operatorname{diag}(\boldsymbol{W}_2))\right] \leq \frac{1}{2}\sum_i (\|\boldsymbol{X}_i\|_X^2 + \|\boldsymbol{Y}_i\|_Y^2).$$

As we saw in the proof of Proposition 3.6.3, the semidefinite block matrix (3.21) corresponds to a dyadic decomposition of the operator $\mathcal{A}$. In particular, for $\boldsymbol{W}_1 = \boldsymbol{X}\boldsymbol{X}^{\mathsf{t}}$, we have that $\boldsymbol{X}_j$ in the dyadic decomposition is exactly the jth column of $\boldsymbol{X}$ reshaped to have dimension $m \times n$.

If we let $\boldsymbol{x}_{:j}$ be the jth column of $\boldsymbol{X}$, then

$$\begin{aligned}\sum_j \|\boldsymbol{X}_j\|_X^2 = \sum_j \|\boldsymbol{x}_{:j}\|_X^2 &= \sum_j g_X(|x_{:j}|)^2 \\ &\geq g_X\left(\sum_j |x_{:j}|^2\right) = g_X(\operatorname{diag}(\boldsymbol{X}\boldsymbol{X}^{\mathsf{t}})) = g_X(\operatorname{diag}(\boldsymbol{W}_1)),\end{aligned}$$

where we use the fact that gauges satisfy a triangle inequality. As we saw earlier, the squared Euclidean norms of the rows of $\boldsymbol{X}$ coincide with the diagonal entries of $\boldsymbol{W}_1 = \boldsymbol{X}\boldsymbol{X}^{\mathsf{t}}$. A similar relationship holds for the squared Y norms of the $\boldsymbol{Y}_i$ in the dyadic decomposition of $\mathcal{A}$. □

We can now apply Definition 3.6.8 to the superquadratic norm examples from the previous section.

Example 3.6.10 (Relaxation of the $\ell_\infty \otimes \ell_\infty$ nuclear norm)**.** Let the factor spaces be $X = \ell_\infty^{mn}$ and $Y = \ell_\infty^{pq}$. Therefore we have that

$$g_X(\operatorname{diag}(\boldsymbol{W}_1)) = \max_{i=1,\ldots,mn} [\boldsymbol{W}_1]_{ii} \quad \text{and} \quad g_Y(\operatorname{diag}(\boldsymbol{W}_2)) = \max_{j=1,\ldots,pq} [\boldsymbol{W}_2]_{jj}.$$

The relaxed nuclear norm is then

$$R_{X,Y}(\mathcal{A}) := \inf \left\{ \frac{1}{2} \left[\max_i [\boldsymbol{W}_1]_{ii} + \max_j [\boldsymbol{W}_2]_{jj} \right] : \begin{bmatrix} \boldsymbol{W}_1 & \boldsymbol{A} \\ \boldsymbol{A}^\mathsf{t} & \boldsymbol{W}_2 \end{bmatrix} \succeq \boldsymbol{0} \right\},$$

for each operator $\mathcal{A} \in \mathbb{O}^{m \times n \otimes p \times q}$.

This relaxed nuclear norm appears in the machine learning literature as the *max-norm*. [Lee+10; SS05; SRJ05]

Example 3.6.11 (Relaxation of the $\ell_2 \otimes \ell_\infty$ nuclear norm)**.** Let the factor spaces be $X = \ell_2^{mn}$ and $Y = \ell_\infty^{pq}$. Therefore we have that

$$g_X(\operatorname{diag}(\boldsymbol{W}_1)) = \operatorname{tr}(\boldsymbol{W}_1) \quad \text{and} \quad g_Y(\operatorname{diag}(\boldsymbol{W}_2)) = \max_{j=1,\ldots,pq} [\boldsymbol{W}_2]_{jj}.$$

The relaxed nuclear norm is then

$$R_{X,Y}(\mathcal{A}) := \inf \left\{ \frac{1}{2} \left[\operatorname{tr}(\boldsymbol{W}_1) + \max_j [\boldsymbol{W}_2]_{jj} \right] : \begin{bmatrix} \boldsymbol{W}_1 & \boldsymbol{A} \\ \boldsymbol{A}^\mathsf{t} & \boldsymbol{W}_2 \end{bmatrix} \succeq \boldsymbol{0} \right\},$$

for each operator $\mathcal{A} \in \mathbb{O}^{m \times n \otimes p \times q}$.

A similar relaxation applies to the $\ell_\infty \otimes \ell_2$ nuclear norm.

Example 3.6.12 (Relaxation of the $\ell_r \otimes \ell_s$ nuclear norm for $r, s \geq 2$)**.** Let the factor spaces be $X = \ell_r^{mn}$ and $Y = \ell_s^{pq}$ with $r, s \geq 2$. Therefore we have that

$$g_X(\operatorname{diag}(\boldsymbol{W}_1)) = \|\operatorname{diag}(\boldsymbol{W}_1)\|_{r/2} \quad \text{and} \quad g_Y(\operatorname{diag}(\boldsymbol{W}_2)) = \|\operatorname{diag}(\boldsymbol{W}_2)\|_{s/2}.$$

The relaxed nuclear norm is then

$$R_{X,Y}(\mathcal{A}) := \inf \left\{ \frac{1}{2} \left[\|\operatorname{diag}(\boldsymbol{W}_1)\|_{r/2} + \|\operatorname{diag}(\boldsymbol{W}_2)\|_{s/2} \right] : \begin{bmatrix} \boldsymbol{W}_1 & \boldsymbol{A} \\ \boldsymbol{A}^\mathsf{t} & \boldsymbol{W}_2 \end{bmatrix} \succeq \boldsymbol{0} \right\},$$

for each operator $\mathcal{A} \in \mathbb{O}^{m \times n \otimes p \times q}$.

3.6.6 The quality of the relaxation

In Proposition 3.6.9 we showed the relaxed nuclear norms for superquadratic factor spaces indeed minorize their corresponding true nuclear norms. Given

this, a natural question is how closely the relaxations coincide with the true norms. Remarkably, these relaxations approximate their nuclear norms quite well in some important cases of interest. The results of this section originally appeared in the famous 1953 paper of Grothendieck [Gro53] (see also Section 2.3.2).

The central result of that paper, known as Grothendieck's Theorem, proves that the $\ell_\infty \otimes \ell_\infty$ semidefinite relaxation deviates from the true nuclear norm by a dimension-independent multiplicative constant. Grothendieck-type results exist for other nuclear norm relaxations, and we present one for the $\ell_2 \otimes \ell_\infty$ case (also due to Grothendieck himself). See Pisier's book [Pis86] and his more recent survey [Pis12] for a history of Grothendieck-type results.

3.6.6.1 Grothendieck's Theorem

Grothendieck, in his 1953 paper [Gro53], presented a result he called "the fundamental theorem in the metric theory of tensor products." This result, now known as Grothendieck's Theorem, is an inequality between three fundamental norms of the tensor product space $\ell_\infty \otimes \ell_\infty$.[1] While our terminology differs, we have already seen two of these norms.

On $\ell_\infty \otimes \ell_\infty$, the nuclear norm $N_{\ell_\infty,\ell_\infty}$ is also known as the *projective tensor product norm*, while the relaxed norm $R_{\ell_\infty,\ell_\infty}$ is the *Hilbertian norm*. The Grothendieck Theorem allows us to compare these two norms, and we restate the result here in our notation.

Proposition 3.6.13 (The Grothendieck Theorem)**.** *Let the factor spaces be* $X = \ell_\infty^{mn}$ *and* $Y = \ell_\infty^{pq}$*. For every* $\mathcal{A} \in \mathbb{O}^{m\times n\otimes p\times q}$*,*

$$R_{\ell_\infty,\ell_\infty}(\mathcal{A}) \leq N_{\ell_\infty,\ell_\infty}(\mathcal{A}) \leq K_G^{\mathbb{R}} \cdot R_{\ell_\infty,\ell_\infty}(\mathcal{A}),$$

where the real Grothendieck constant $K_G^{\mathbb{R}}$ *satisfies* $1.66 \leq K_G^{\mathbb{R}} < \pi/(2\log(1+\sqrt{2})) \approx 1.8$.

The above bounds, due to Krivine [Kri77], improve upon Grothendieck's original result that $1.57 \approx \pi/2 \leq K_G^{\mathbb{R}} \leq \sinh(\pi/2) \approx 2.30$. While Krivine conjectured that his upper bound was optimal, recent work by Braverman et al. [BMMN11] shows that it is not. The exact value for Grothendieck's constant remains unknown.

[1] In fact, Grothendieck's result is not limited to finite-dimensional spaces, but we remain firmly in the finite-dimensional setting for this work.

To prove this theorem, Grothendieck uses a relaxation and rounding argument. The relaxation step involves transforming a computationally difficult optimization problem into one that is tractable. This corresponds exactly to our relaxation of the nuclear norm. The rounding step relates the solution of the relaxed problem back to the original, difficult one by randomly rounding the computed optimal point into a feasible point of the original problem. This is done in such a way as to ensure that values of the relaxed and original problems are comparable.

This relaxation and rounding strategy for approximating solutions to combinatorial optimization problems—such as computing the $\ell_\infty \otimes \ell_\infty$ nuclear norm—has since become mainstream. Lovász [Lov79] harnessed this technique to devise a polynomial-time approximation—known as the *Lovász theta function*—of certain difficult-to-compute graph quantities. His formulation explicitly utilized semidefinite programming in the relaxation step.

More recently, Goemans and Williamson [GW95] famously used a semidefinite rounding and relaxation approach to approximate the optimal solution of the MAX-CUT problem. This work led to increased interest in semidefinite programming for approximation algorithms. Additionally, Alon and Naor [AN06] showed that a rounding approach based on proofs of Grothendieck's Theorem can well-approximate the CUT-NORM problem, a generalization of MAX-CUT. The survey of Khot and Naor [KN12] provides further connections between Grothendieck-type theorems and problems in combinatorial optimization.

3.6.6.2 The "Little" Grothendieck Theorem

In subsequent chapters we consider the empirical performance of the $\ell_2 \otimes \ell_\infty$ and $\ell_\infty \otimes \ell_2$ nuclear norms. It happens that a Grothendieck-type theorem exists to characterize the quality of their semidefinite relaxations as well.

Proposition 3.6.14 (The "Little" Grothendieck Theorem)**.** *Let the factor spaces be* $X = \ell_\infty^{mn}$ *and* $Y = \ell_\infty^{pq}$*. For every* $\mathcal{A} \in \mathbb{O}^{m\times n\otimes p\times q}$*,*

$$R_{\ell_2,\ell_\infty}(\mathcal{A}) \leq N_{\ell_2,\ell_\infty}(\mathcal{A}) \leq k_G^{\mathbb{R}} \cdot R_{\ell_2,\ell_\infty}(\mathcal{A}),$$

where the little Grothendieck constant $k_G^{\mathbb{R}} = \pi/2$ *is optimal. A similar relationship holds for* $\ell_\infty \otimes \ell_2$*.*

Grothendieck also proved this result [Gro53, Thm. 4, p. 51] in his 1953 paper and established that the constant $k_G^{\mathbb{R}}$ is optimal [Gro53, Coro. 1, p. 51].

Chapter 4

The `operfact` Python package

In this chapter we present the Python software package `operfact` we developed to implement operator recovery problems using the nuclear norm framework described in Chapter 3. This software is open-source and available on GitHub[I] and through PyPI, the official Python package repository. Here we discuss the implementation choices and basic usage of the package; a reference guide to its classes and methods is included with the software.

4.1 Overview

Our exploration of operator factorization models requires solving many similar optimization problems. Instead of writing individual solvers for the models we wish to test, we develop a higher-level tool that allows us to programmatically modify a generic solver to fit any of our particular needs. In this way we can more quickly prototype problems of interest and run the numerical experiments necessary to validate these models.

This document describes the design and implementation of this framework in the form of a Python package named `operfact` (an unimaginative shortening of operator factorization). Subsequent chapters present concrete examples that illustrate the use of the package.

[I] https://github.com/jbruer/operfact

4.1.1 The optimization problem

In Chapter 1 we described the situation where we observe linear measurements $\boldsymbol{b} \in \mathbb{R}^s$ of an operator $\mathcal{A}^\natural \in \mathbb{O}^{m\times n\otimes p\times q}$ given as

$$\boldsymbol{b} = \boldsymbol{\mu}(\mathcal{A}^\natural) + \boldsymbol{z},$$

where $\boldsymbol{\mu}\colon \mathbb{O}^{m\times n\otimes p\times q} \to \mathbb{R}^s$ is a linear map and $\boldsymbol{z} \in \mathbb{R}^s$ is additive noise. We wish to know when we can either:

- Find a factorization $\mathcal{A}^\natural = \sum_i \boldsymbol{X}_i \otimes \boldsymbol{Y}_i$ where the factors $\boldsymbol{X}_i$ and $\boldsymbol{Y}_i$ have some particular structure.
- Approximate $\mathcal{A}^\natural$ given that it admits such a factorization.

We use nuclear norms (Chapter 3) as regularizers that encode our prior assumptions on the factor structure of the true signal $\mathcal{A}^\natural$ and solve the problem

$$\underset{\mathcal{A}}{\text{minimize}} \quad \frac{1}{2}\|\boldsymbol{b} - \boldsymbol{\mu}(\mathcal{A})\|_{\ell_2}^2 + \lambda N_{X,Y}(\mathcal{A}), \tag{4.1}$$

where we use the squared Euclidean norm to measure the fidelity of the candidate operator $\mathcal{A}$ to the measurements $\boldsymbol{b}$, and $N_{X,Y}$ is the nuclear norm on $X \otimes Y$. The positive constant λ adjusts the balance between our desires to recover highly structured operators and to remain faithful to the observations. In the remainder of this work, we measure fidelity using the Euclidean norm. In general, this loss measurement should be sensitive to the additive noise $\boldsymbol{z}$. We consider the function loss$(\cdot)$ to be more sensitive to the noise $\boldsymbol{z}$ if loss$(\sigma\boldsymbol{z})$ increases more rapidly as the scale σ increases. We assume isotropic Gaussian noise, and so the Euclidean norm serves us well.

Due to the convexity of the nuclear norm and the linearity of $\boldsymbol{\mu}$, the problem (4.1) is a convex optimization problem. Therefore finding a local minimum of the objective yields the global minimum. This fact allows for solvers that can provide strong convergence guarantees.

When the nuclear norm $N_{X,Y}$ admits a simple optimal decomposition, the optimization problem (4.1) allows us to approximate the operator $\mathcal{A}$ and factor it. We saw examples of this in Sections 3.4 and 3.5, with the trace norm and whenever X or Y is the ℓ_1 norm.

In some cases, where computing the nuclear norm is difficult, we can rely on a semidefinite relaxation strategy. We detailed this in Section 3.6. These relaxations can improve the computational situation without much sacrifice in the power of the regularizer. Furthermore, we defined this relaxation in terms of a factorization. Therefore solving these problems also allows us to retrieve a corresponding dyadic decomposition. We implement this approach in Section 4.7.4.

In general, we do not have either option. For these cases we utilize alternating minimization. Factorization then requires an additional step or consideration during the optimization process. In the next section we describe alternating minimization. We discuss our implementation in Section 4.7.3.

4.1.2 Roadmap

We discuss alternating minimization, our default method for solving the operator recovery problem (4.1), in Section 4.2. We discuss the higher-level choices of our implementation in Section 4.3 before we outline the individual components of our software package.

The package `operfact` itself comprises four main modules. The `operators` module discussed in Section 4.4 concerns the generation and manipulation of operators. The `measurements` module discussed in Section 4.5 provides a standard way to define linear measurement maps of operators and provides several examples of such maps. The `regularizers` module discussed in Section 4.6 implements the nuclear norm framework for use in numerical solvers. The `solvers` module discussed in Section 4.7 implements solvers for the nuclear norm recovery problem (4.1) via CVXPY. Additionally, the `utils` module provides helper methods to perform embarrassingly parallel numerical experiments; see the reference manual for more information about this module.

We introduce the contents of each module and provide code samples highlighting the usage of these features. The reference manual included with the package contains additional details.

4.2 Alternating minimization

In most cases we consider, the nuclear norm on $X \otimes Y$ has no simple closed-form representation. We instead resort to solving (4.1) using alternating minimiza-

tion.

4.2.1 Transformation to a nonconvex problem

To obtain the nonconvex problem, we first form an explicit factorization $\mathcal{A} = \sum_{i=1}^{r} \boldsymbol{X}_i \otimes \boldsymbol{Y}_i$ of the decision variable. Using the definition (3.10) of the nuclear norm, we create a new optimization problem over the factors $\boldsymbol{X}_i$ and $\boldsymbol{Y}_i$:

$$\underset{\boldsymbol{X}_i, \boldsymbol{Y}_i}{\text{minimize}} \quad \frac{1}{2}\left\| \boldsymbol{b} - \boldsymbol{\mu}\left(\sum_{i=1}^{r} \boldsymbol{X}_i \otimes \boldsymbol{Y}_i \right) \right\|_{\ell_2}^2 + \lambda \sum_{i=1}^{r} \|\boldsymbol{X}_i\|_X \|\boldsymbol{Y}_i\|_Y. \tag{4.2}$$

We notice two important differences between (4.2) and (4.1). First, this problem is no longer convex due to the bilinear relationship between the $\boldsymbol{X}_i$ and $\boldsymbol{Y}_i$ under $\boldsymbol{\mu}$. We must pay this penalty to use this factored form.

Second, the bound r on the sum indicates that we now search over all decompositions of the operator with r dyads as opposed to all finite decompositions. We are therefore enforcing a rank constraint on the solution. In our quest to recover low-rank operators, this may be useful. On the other hand, results from the literature suggest that this nonconvex problem becomes "easier" as the solver rank grows [BM04; BMP08; HV15]. Namely, higher solver rank can remove spurious local minima from the problem.

4.2.1.1 An alternative formulation of the problem

Recall that Proposition 3.6.1 provides an alternative formulation (3.19) of the nuclear norm. Using this, instead of the definition (3.10), results in the nonconvex optimization problem

$$\underset{\boldsymbol{X}_i, \boldsymbol{Y}_i}{\text{minimize}} \quad \frac{1}{2}\left\| \boldsymbol{b} - \boldsymbol{\mu}\left(\sum_{i=1}^{r} \boldsymbol{X}_i \otimes \boldsymbol{Y}_i \right) \right\|_{\ell_2}^2 + \lambda \sum_{i=1}^{r} \frac{1}{2}(\|\boldsymbol{X}_i\|_X^2 + \|\boldsymbol{Y}_i\|_Y^2). \tag{4.3}$$

We refer to (4.2) as the "product" formulation and to (4.3) as the "sum" formulation. While both originate from equivalent definitions of the nuclear norm, the numerical implementations of these problems will be different. In fact, the documentation of CVX [GB14] states that:

> One particular reformulation that we strongly encourage is to eliminate quadratic forms [...] whenever it is possible to construct equivalent models using `norm` instead. Our experience tells us

Algorithm 1 Alternating minimization

Require: measurement map $\boldsymbol{\mu}$, observed vector $\boldsymbol{b}$
Require: regularization constant λ, solver rank r
Require: initialization $\boldsymbol{Y}_i^0$, $i = 1, \ldots, r$
1: **for** $t = 0, 1, \ldots, t_{\max}$ **do**
2: $\quad \{\boldsymbol{X}_i^t\} \leftarrow \arg\min_{\{\boldsymbol{X}_i\}} \frac{1}{2}\|\boldsymbol{b} - \boldsymbol{\mu}(\sum_{i=1}^r \boldsymbol{X}_i \otimes \boldsymbol{Y}_i^{t-1})\|_{\ell_2}^2 + \lambda \sum_{i=1}^r \|\boldsymbol{X}_i\|_X \|\boldsymbol{Y}_i^{t-1}\|_Y$
3: $\quad \{\boldsymbol{Y}_i^t\} \leftarrow \arg\min_{\{\boldsymbol{Y}_i\}} \frac{1}{2}\|\boldsymbol{b} - \boldsymbol{\mu}(\sum_{i=1}^r \boldsymbol{X}_i^t \otimes \boldsymbol{Y}_i)\|_{\ell_2}^2 + \lambda \sum_{i=1}^r \|\boldsymbol{X}_i^t\|_X \|\boldsymbol{Y}_i\|_Y$
4: **end for**
5: **return** $\{\boldsymbol{X}_i^{t_{\max}}\}$, $\{\boldsymbol{Y}_i^{t_{\max}}\}$

that quadratic forms often pose a numerical challenge for the underlying solvers that CVX uses.

We acknowledge that this advice goes against conventional wisdom: quadratic forms are the prototypical smooth convex function, while norms are nonsmooth and therefore unwieldy. But with the conic solvers that CVX uses, this wisdom is exactly backwards. It is the norm that is best suited for conic formulation and solution.

We use CVXPY to model our optimization problems, but it uses conic formulations just like CVX. This note, therefore, suggests that the "product" formulation (4.2) will offer higher accuracy. Our numerical experimentation, however, shows that this is not always the case, and we explore this further in Section 5.4.4.2.

4.2.2 The algorithm

The *alternating minimization* algorithm for (4.2) is shown in Algorithm 1. We proceed by first fixing the $\boldsymbol{Y}_i$ and solving with respect to the $\boldsymbol{X}_i$. Then we fix the $\boldsymbol{X}_i$ and solve for the $\boldsymbol{Y}_i$. Note that when fixing one set of factors, the problem again becomes convex in the other set. In this way we turn the nonconvex problem into a set of linked convex subproblems. We repeat these steps until we either reach convergence or complete a predetermined maximal number of iterations.

4.2.3 Initialization

In Section 2.2.4 we reviewed initialization schemes for nonconvex optimization methods found in the literature. Here we discuss in more detail our default

initialization scheme for the alternating minimization solver shown in Algorithm 1.

Jain et al. [JNS12] solve the related matrix sensing problem using alternating minimization. That is,

$$\underset{\boldsymbol{X},\boldsymbol{Y}}{\text{minimize}} \quad \|\boldsymbol{b} - \boldsymbol{\mu}(\boldsymbol{X}\boldsymbol{Y}^{\mathsf{t}})\|_{\ell_2}^2, \tag{4.4}$$

where the inner dimension of the matrix factorization is r. In our dyadic notation, this is

$$\underset{\boldsymbol{x}_i,\boldsymbol{y}_i}{\text{minimize}} \quad \left\| \boldsymbol{b} - \boldsymbol{\mu}\left(\sum_{i=1}^{r} \boldsymbol{x}_i \otimes \boldsymbol{y}_i \right) \right\|_{\ell_2}^2.$$

The solver first minimizes over the left factors $\boldsymbol{x}_i$ while keeping the right factors $\boldsymbol{y}_i$ fixed. For the first iteration, they must therefore provide the solver with an initial estimate of the right factors.

Jain et al. analyze the alternating minimization algorithm for matrix sensing as a perturbed power method. They note that due to this connection it is critical that the subspace spanned by the right factors $\boldsymbol{y}_i$ not be orthogonal (or nearly orthogonal) to the true subspace. They prove that *spectral initialization* works well.

That is, they first compute the SVD of the matrix $\boldsymbol{\mu}^*(\boldsymbol{b})$, where $\boldsymbol{\mu}^*$ is the adjoint of the measurement map. Then they set the $\boldsymbol{y}_i$ to be the top-r right singular vectors.

We recognize, however, that our definition of dyads (Definition 3.2.1) is through a correspondence with matrices. Therefore, by dropping the nuclear norm term from (4.2), we can formulate a similar low-rank operator sensing problem,

$$\underset{\boldsymbol{X}_i,\boldsymbol{Y}_i}{\text{minimize}} \quad \left\| \boldsymbol{b} - \boldsymbol{\mu}\left(\sum_{i=1}^{r} \boldsymbol{X}_i \otimes \boldsymbol{Y}_i \right) \right\|_{\ell_2}^2.$$

Even though we now have matrix factors $\boldsymbol{X}_i$ and $\boldsymbol{Y}_i$, we can still regard this as a low-rank *matrix* sensing problem. This leads us to also consider the same spectral initialization.

That is, we first compute

$$\mathcal{T} = \boldsymbol{\mu}^*(\boldsymbol{b}),$$

where $\boldsymbol{\mu}^*$ denotes the adjoint of our measurement map. The initialization then results by taking the dyadic SVD (3.7) of $\mathcal{T}$,

$$\mathcal{T} = \sum_i \sigma_i \boldsymbol{U}_i \otimes \boldsymbol{V}_i,$$

and setting the initial right factors $\boldsymbol{Y}_i^0 := \boldsymbol{V}_i$ for $i = 1, \ldots, r$.

Our tests using this initialization on the nuclear norm recovery problem (4.2) show that it works well. In particular, the spectral initialization leads to faster—and more accurate—solutions of various nuclear norm problems as compared to random initialization.

4.2.4 Convergence

We have no guarantees that the alternating minimization algorithm applied to (4.2) will converge to a globally optimal solution. This is unfortunate, but it does not dissuade us from using this nonconvex method. In fact, recent work on matrix completion [JNS12; Har14], dictionary learning [AAJN16], phase retrieval [NJS15], and blind deconvolution [LLJB15] provide recovery guarantees for specific implementations of alternating minimization and begin to explain the empirical successes of these methods.

Note, however, that all of these results still depend on randomized models for their measurement maps or underlying signals. For instance, the analysis of matrix sensing in Jain et al. [JNS12] assumes that the measurement map $\boldsymbol{\mu}$ in (4.4) satisfies a *restricted isometry property (RIP)* [CT05; CRT06b]. Unfortunately, it is computationally hard to show that a particular map $\boldsymbol{\mu}$ actually satisfies this property [TP14; NW14], and so a common technique relies on using maps $\boldsymbol{\mu}$ from random ensembles that satisfy the RIP with high probability [BDDW08].

Nevertheless, alternating minimization allows for flexibility in our applications. We can quickly implement new measurement maps $\boldsymbol{\mu}$ and nuclear norms $N_{X,Y}$ in conjunction with standard front-ends to convex solvers (e.g., CVXPY [DB16]). Additionally, the storage savings in using relatively low-rank factorized forms of operators allows for larger problem sizes than the convex solvers. These reasons, combined with our empirical success in using alternating minimization, make this a suitable method for our work.

4.3 Design choices

The `operfact` package requires Python 3.5 and relies on the convex optimization modeling package CVXPY [DB16] to build representations of the operator recovery problem (4.1) that we can then pass to external solvers.[2] We use CVXPY since our goal is to prototype optimization problems in a flexible manner with minimal code. This library arose from the success of CVX [GB14; GB08] and its implementation of disciplined convex programming (DCP) [GBY06]. While we must pay some overhead associated with the modeling step—including potential inefficiencies in the standard form problem that is sent to the solver—we can more quickly get to the task of actually solving the problem. In other words, we pay a smaller cost each time we solve the problem versus paying a potentially much larger cost in writing software.

The use of `operfact` again requires a small computational overhead in setting up our factorization problems, and the optimization problems generated by CVXPY are somewhat less efficient than a purpose-built solver. In addition, we have had to spend time developing `operfact` itself. The result, however, is that we are able to test many different combinations of measurements and regularizers now with little additional effort. That is, we can focus on finding interesting operator models to pursue. Any particular application could be optimized for computational performance in the future.

4.3.1 Why CVXPY?

We recognize that our choice of CVXPY as our modeling software is not obvious. We could have used CVX under Matlab or Convex.jl [Ude+14] under Julia. All of these packages adhere to the principles of disciplined convex programming. Here we review the major factors in our decision.

4.3.1.1 Base language

In addition to the choice of modeling software, we must consider the base language of our package. We envisioned our software as an object-oriented interface for creating operator optimization problems. Python, Matlab, and Julia all allow for user-defined objects. In our opinion, Python and Julia provide a cleaner

[2] The package uses Python 3.5-specific features sparingly, but for simplicity we do not maintain compatibility with earlier versions of Python.

interface for doing so. All three languages have sufficient support for working with arrays and data either through built-in functions or well-established external libraries.

While the Julia language provides great promise for scientific computing, its core language is still undergoing rapid development and change. When starting this project, we simply wanted a programming language that was more stable in both its design and implementation. Our current familiarity with both Matlab and Python also led us to shy away from Julia. We do recognize, though, that the Julia syntax provides a familiar interface for current Matlab users. Furthermore, its use of method overloading (termed *multiple dispatch*) has a certain attractiveness for defining and operating upon mathematical objects. Note that the performance advantages of Julia were not a consideration for our project. The time spent in external convex solvers dominates the time spent within both the modeling framework and `operfact` itself.

4.3.1.2 Stateful optimization problems

Our initial numerical work on this project involved tests in Matlab using both a Burer–Monteiro-type approach [BM03] to rank-constrained minimization and alternating minimization with CVX. Alternating minimization requires repeated optimization over subsets of the variables in a problem. CVX, however, provides no ability to retain a model after the solver returns. That is, we must create a new model every time we solve a subproblem. This costs us time and also prevents us from using previous solutions to warm-start the solver.

Meanwhile, both CVXPY and Convex.jl allow us to create optimization problems as objects. These objects retain their state and allow us to reuse them across iterations. With CVXPY, we create objects for each subproblem that contain both the variables to optimize as well as parameters—the variables we hold fixed. We use the solution of one subproblem to update the parameters of the other. Convex.jl, on the other hand, actually allows us to create just one optimization problem and fix/free variables as we alternate between subproblems. While this approach is more elegant than our solution with CVXPY, the key advantage of both packages over CVX is stateful optimization objects.[3]

[3] A very recent paper by Shen et al. [She+16] describes an extension to the DCP ruleset to handle multi-convex problems. Given a partition of the decision variables, an optimization problem is multi-convex provided that it is convex when optimizing over each set of the partition (and holding the remaining variables fixed). An accompanying Python package, DMCP, extends

4.3.1.3 Connection to external solvers

CVPXY also integrates with the SCS solver [OCPB16]. This first-order solver for convex cone programs allows us to more efficiently handle larger operator recovery problems than the default interior-point solvers of both CVXPY and CVX. We note that the latest beta versions of CVX also include SCS support, and so this no longer serves as a strong differentiator.

4.3.1.4 Parallelization

In our work, we must also solve many independent operator recovery problems. While CVX can work with parallel computing facilities in Matlab, this is not well-supported. We admittedly do not require anything more than the ability to run multiple instances of our software; the parallelization could be accomplished through the use of shell scripts. The ability, however, to use the basic parallelization features of Python proves useful. Furthermore, running Matlab on multiple machines requires licenses for those machines. This licensing may not have posed a problem for our work, but it is a consideration in the use of Matlab.

4.3.1.5 Caveats

We must note that CVXPY does have some disadvantages versus CVX. First, it does not yet support convex variables. While complex numbers certainly appear in practical applications, we do not require this for our investigation of nuclear norms. Second, CVX creates more efficient representations of convex problems. Its ability to eliminate redundant constraints, for instance, surpasses that of CVXPY. The problems sent to the external solvers are often more compact, and this can lead to a speedup. Third, CVX provides a greater number of convex "atoms" that can be used in creating optimization problems. This reflects the more established nature of CVX in general. For our application, however, the advantages of CVXPY and Python outweigh their shortcomings.

4.4 Operators

The `operfact.operators` module defines two types of operator objects along with some utility functions to generate and manipulate them.

CVXPY to solve multi-convex problems using alternating minimization. A future version of `operfact` could make use of this facility.

4.4.1 The `ArrayOperator`

As discussed in Section 3.2.2 we can think of operators as 4-dimensional arrays with entries a_{ijkl} as in

$$\mathcal{A} = \sum_{ijkl} a_{ijkl}(\boldsymbol{E}_{ij} \otimes \boldsymbol{E}_{kl}),$$

where the a_{ijkl} are the entries of the operator and the $\boldsymbol{E}_{ij}$, $\boldsymbol{E}_{kl}$ are the standard basis matrices.

We implement this operator as a wrapper around a standard NumPy 4-dimensional array. This means that all built-in array operations work with these operators as well.

We create such an operator from an existing 4D array as follows:

```
import numpy as np
from operfact import operators

shape = (m, n, p, q)
oper = operators.ArrayOperator(np.random.normal(size=shape))  # a
    random Gaussian operator
oper.T  # standard NumPy array methods work
```

The `operfact` package assumes that all operators have four dimensions, but it is worth noting that NumPy arrays will not enforce this condition. It is therefore possible to create or manipulate `ArrayOperator` objects so that they have fewer dimensions. Care must be taken to ensure that any operations on `ArrayOperators` perform adequate error-checking.

4.4.2 The `DyadsOperator`

While the `ArrayOperator` proves useful for ingesting and outputting tabular (or tensorial) data, we also require an operator object that can represent operator factorizations. The `DyadsOperator` class allows for representing operators as sums of dyads, precisely how we defined operators in Section 3.2.2.

Given an operator

$$\mathcal{A} = \sum_{i=1}^{r} \boldsymbol{X}_i \otimes \boldsymbol{Y}_i,$$

we call the $\boldsymbol{X}_i$ the left factors and the $\boldsymbol{Y}_i$ the right factors.

We construct the same operator in Python as follows from lists of the factors:

```
import numpy as np
from operfact import operators

shape = (m, n, p, q)
nfactors = r
# Generate random left and right factors
Xs = [np.random.normal(size=shape[0:2]) for r in range(nfactors)]
Ys = [np.random.normal(size=shape[2:4]) for r in range(nfactors)]
# Combine to form the DyadsOperator
oper = operators.DyadsOperator(Xs, Ys)
# The DyadsOperator infers the shape and nfactors from the lists
assert oper.shape == shape
assert oper.nfactors == nfactors
```

As shown in the listing, a `DyadsOperator` computes its shape and number of factors (dyads) from the lists passed in its initialization. We can access the left and right factors using the `lfactors` and `rfactors` properties. These are simply lists of the factor matrices.

We can transform a `DyadsOperator` into other useful representations. The `asArrayOperator` method computes a 4-dimensional `ArrayOperator` object from the dyadic representation. Similarly, the `asmatrix` method computes the matrix representation (3.6) of the operator.

Note that the `DyadsOperator` in the example has a storage requirement of $r(mn + pq)$ numbers whereas the `ArrayOperator` and matrix representations require $mnpq$ numbers. There can be a substantial savings with the dyadic representation of low-rank operators.

Finally, we can apply the dyadic representation of the operator to a matrix as the linear operator (3.3). That is,

$$\mathcal{A}(\boldsymbol{M}) = \sum_i \boldsymbol{X}_i \boldsymbol{M} \boldsymbol{Y}_i^{\mathrm{t}},$$

for any appropriately-sized matrix $\boldsymbol{M}$. The methods `apply` and `cvxapply` implement this calculation for NumPy arrays and CVXPY matrices, respectively.

4.4.3 Utility functions

The module also contains a few utility methods. First, the `RandomDyadsOperator` method creates a new `DyadsOperator` object with randomly generated factors.

The default distribution for the factors is standard Gaussian, but it may also be specified by the user. (See the manual for more details.)

We can then create the operator in the previous listing as follows:

```
from operfact import operators

shape = (m, n, p, q)
nfactors = r
# Generate a DyadsOperator with random standard Gaussian factors
oper = operators.RandomDyadsOperator(shape, nfactors)
```

We often wish to compute the inner product between operators. The `innerprod` method takes two operators as input and returns their inner product. The inputs may be any combination of `ArrayOperator` and `DyadsOperator` objects. In the case where two `DyadsOperator` objects have factors that are CVXPY expressions, we can compute their inner product using the specialized `cvxinnerprod` method.

Finally, the `kpsvd` method computes the dyadic SVD (3.7) of both `ArrayOperator` and `DyadsOperator` objects. It handles the necessary conversion of the operator to is matrix representation (3.6) and calls the NumPy SVD method. The output follows NumPy conventions as shown in this listing.

```
from operfact import operators

shape = (m, n, p, q)
nfactors = r
# Generate a DyadsOperator with random standard Gaussian factors
oper = operators.RandomDyadsOperator(shape, nfactors)
U, S, Vt = operators.kpsvd(oper)  # take the dyadic SVD
mat = U @ S @ Vt  # reconstruct the matrix representation of the
    operator
```

4.5 Measurements

The `operfact.measurements` module defines objects that represent linear measurement maps. A base class `Measurement` outlines the properties and methods these objects may have; Table 4.1 lists these. We have also implemented several different types of measurement maps. The system, however, is extensible. We show this, for instance, in Section 7.3.2 where we implement a measurement object for self-calibration problems.

Property	Description
`shape`	The dimension of operators in the domain
`nmeas`	The size of the resulting measurement vector (co-domain)
Method	**Description**
`apply`	Apply the measurement map to an operator
`cvxapply`	Apply the measurement map to a CVXPY object
`matapply`	Apply the measurement map to an operator in matrix form
`asOperator`	Return the linear map as list of `ArrayOperator` objects
`initfrommeas`	Apply the adjoint (used in spectral initialization, Sec. 4.2.3)

Table 4.1: **Properties and methods of a `Measurement` object.** The `Measurement` base class provides the abstract specification for a linear measurement map. This table lists the properties and methods required for its implementation.

4.5.1 `InnerProductMeasurement`

Given the space of operators $\mathbb{O}^{m\times n\otimes p\times q}$ equipped with the inner product as in Section 3.2.2, we can write any linear functional on that space as an inner product. That is, every linear functional $\mu\colon \mathbb{O}^{m\times n\otimes p\times q} \to \mathbb{R}$ takes the form

$$\mu(\mathcal{A}) = \langle \mathcal{M}, \mathcal{A}\rangle,$$

for some $\mathcal{M} \in \mathbb{O}^{m\times n\otimes p\times q}$.

Such a functional returns a single linear measurement of an operator, and we provide an implementation of such measurement maps in the `InnerProductMeasurement` class. To instantiate this measurement object, we simply provide the fixed operator corresponding to $\mathcal{M}$ above.

```
from operfact import measurements, operators

shape = (m, n, p, q)
# Generate a random Gaussian ArrayOperator
oper = operators.ArrayOperator(np.random.normal(size=shape))
# Measurement map returning a single random Gaussian measurement
meas = measurements.InnerProductMeasurement(oper)
```

Note that the adjoint $\boldsymbol{\mu}^*\colon \mathbb{R} \to \mathbb{O}^{m\times n\otimes p\times q}$ is simply

$$\boldsymbol{\mu}^*(b) = b\mathcal{M}.$$

Every linear measurement map $\boldsymbol{\mu}\colon \mathbb{O}^{m\times n\otimes p\times q} \to \mathbb{R}^s$ can be written entrywise as

$$[\boldsymbol{\mu}(\mathcal{A})]_i = \langle \mathcal{M}_i, \mathcal{A}\rangle,$$

where the $\{\mathcal{M}_i\}_{i=1}^s$ are the fixed operators defining the linear map. It can be useful to represent measurement maps in this standard way, and the method `asOperator` serves to convert measurement objects into lists of the operators $\mathcal{M}_i$ that define their mapping.

4.5.2 `IdentityMeasurement`

Sometimes we have access to every entry of our operator of interest, but those observations are corrupted by noise. In this case, the linear measurement map returns the entries of the operator as a vector. Here $\boldsymbol{\mu}\colon \mathbb{O}^{m\times n\otimes p\times q} \to \mathbb{R}^{mnpq}$ is simply

$$\boldsymbol{\mu}(\mathcal{A}) = \mathrm{vec}(\mathcal{A}),$$

and the adjoint is simply the inverse reshaping operation.

We implement this operator as `IdentityMeasurement` and use it extensively in Chapter 5 where we solve denoising problems.

4.5.3 `DirectActionMeasurement`

We may also observe an operator through its action on a matrix. In this case we have a linear map $\boldsymbol{\mu}\colon \mathbb{O}^{m\times n\otimes p\times q} \to \mathbb{R}^{mp}$ given by

$$\boldsymbol{\mu}(\mathcal{A}) = \sum_i \mathrm{vec}(\boldsymbol{X}_i \boldsymbol{M} \boldsymbol{Y}_i^{\mathrm{t}}),$$

where $\boldsymbol{M} \in \mathbb{R}^{n\times q}$ is the fixed matrix defining the measurement map, and $\mathcal{A} = \sum_i \boldsymbol{X}_i \otimes \boldsymbol{Y}_i$ is any dyadic decomposition of $\mathcal{A}$. We implement this with the `DirectActionMeasurement` class.

For the sake of convenience, let us index entries of $\boldsymbol{\mu}(\mathcal{A})$ with two indices $i = 1, \ldots, m$ and $k = 1, \ldots, p$. We then have that

$$[\boldsymbol{\mu}(\mathcal{A})]_{ik} = \langle \mathcal{M}_{ik}, \mathcal{A} \rangle,$$

where

$$[\mathcal{M}_{i'k'}]_{ijkl} = \delta_{ii'}\delta_{kk'} m_{jl},$$

where δ is the Kronecker delta, and m_{jl} is the jl-entry of the fixed matrix $\boldsymbol{M}$.

If we then doubly-index the vector $\boldsymbol{b} \in \mathbb{R}^{mp}$ in exactly the same order as the entries of $\boldsymbol{\mu}(\mathcal{A})$, we compute the adjoint as

$$[\boldsymbol{\mu}^*(\boldsymbol{b})]_{ijkl} = b_{ik} m_{jl},$$

where b_{ik} is the ik-entry of $\boldsymbol{b}$.

4.5.4 `SubsampleMeasurement`

The measurement map that returns a subset of the entries of an operator is a linear map. Indeed, for an ordered set Ω of indices of an operator in $\mathbb{O}^{m\times n\otimes p\times q}$, the measurement map $\boldsymbol{\mu}\colon \mathbb{O}^{m\times n\otimes p\times q} \to \mathbb{R}^{|\Omega|}$ may be written entrywise as

$$[\boldsymbol{\mu}(\mathcal{A})]_i = \langle \mathcal{E}_{\Omega_i}, \mathcal{A}\rangle, \text{ for } i = 1, \ldots, |\Omega|,$$

where the $\mathcal{E}_{\Omega_i}$ are standard basis operators.

The adjoint $\boldsymbol{\mu}^*\colon \mathbb{R}^{|\Omega|} \to \mathbb{O}^{m\times n\otimes p\times q}$ is then given by

$$\boldsymbol{\mu}^*(\boldsymbol{b}) = \sum_{i=1}^{|\Omega|} b_{\Omega_i}\mathcal{E}_{\Omega_i}.$$

The `SubsampleMeasurement` implements this map.

4.5.5 `CombinedMeasurements`

Finally we may use the `CombinedMeasurements` object to form new linear measurement maps by combining other linear measurement maps. Let $\boldsymbol{\mu}_i\colon \mathbb{O}^{m\times n\otimes p\times q} \to \mathbb{R}^{s_i}$ for $i = 1, \ldots, I$. Then if $S = \sum_i s_i$, we can derive the measurement operator $\boldsymbol{\mu}\colon \mathbb{O}^{m\times n\otimes p\times q} \to \mathbb{R}^S$ as

$$\boldsymbol{\mu}(\mathcal{A}) = \begin{bmatrix} \boldsymbol{\mu}_1(\mathcal{A}) \\ \boldsymbol{\mu}_2(\mathcal{A}) \\ \vdots \\ \boldsymbol{\mu}_I(\mathcal{A}) \end{bmatrix}.$$

The adjoint is then

$$\boldsymbol{\mu}^*(\boldsymbol{b}) = \sum_{i=1}^{I} \boldsymbol{\mu}_i^*(\boldsymbol{b}_i),$$

where the $\boldsymbol{b}_i$ are the corresponding subvectors from the above construction of $\boldsymbol{\mu}(\mathcal{A})$.

4.6 Regularizers

The module `operfact.regularizers` implements the nuclear norm framework from Chapter 3 programmatically. We represent nuclear norms (and relaxed

nuclear norms) as objects that can then be passed to solvers. The objects themselves represent a single nuclear norm but may include various computational implementations to work with different solvers.

This package includes three solvers: a direct convex solver (`matsolve`), a nonconvex alternating minimization solver (`altminsolve`), and a semidefinite solver (`sdpsolve`). We discuss the details of these solvers in Section 4.7. A nuclear norm object may be compatible with any combination of these solvers depending on its computability. For instance, Proposition 3.4.1 showed that the $\ell_2 \otimes \ell_2$ nuclear norm is the trace norm, and it is representable directly in CVXPY. Therefore we may use it in the direct convex solver `matsolve`. Proposition 3.6.3, on the other hand, showed that the $\ell_2 \otimes \ell_2$ nuclear norm has a semidefinite representation, and so we may use it with the semidefinite solver `sdpsolve`. Finally, as with all nuclear norms we consider, we can apply the nonconvex alternating minimization approach described in Section 4.2.

All of the nuclear norm objects we discuss derive from a single `Regularizer` base class that serves to codify the relationship between regularizer objects and solvers. The `Regularizer` object defines three methods `norm_altmin`, `norm_mat`, and `norm_sdp` to provide implementations of norms in the specific ways expected by each of the alternating minimization, direct convex, and semidefinite solvers. If a particular regularizer may not be used with a particular solver, its corresponding method may be set to `None`. The `available_solvers` method returns a list of solvers that the regularizer supports (i.e., those `norm` methods not set to `None`). This system allows users to construct additional norms and extend existing norms to work with new solvers.

4.6.1 The helper functions

CVXPY has built-in functions allowing the use of certain norms in its disciplined convex programming framework. We provide a set of aliases for common norms taking the form `norm_x`. For example, setting `x` to be `l1` gives the ℓ_1 norm, while setting `x` to be `s1` gives the Schatten 1-norm (trace norm). We also include easily computable nuclear norms, e.g., `norm_l1l2` represents the $\ell_1 \otimes \ell_2$ nuclear norm. All of these helper functions work directly on CVXPY expressions, and we use them to compose nuclear norm objects. See the reference manual for the complete list.

4.6.2 The `NucNorm` class

The `NucNorm` class derives from the basic `Regularizer` class and defines the interface for working with nuclear norms. To create such an object, we must specify which norms we wish to use on the left and right factor spaces. For instance, to create an object representing the $\ell_2 \otimes \ell_2$ nuclear norm we write

```
from operfact import regularizers as regs

N_l2l2 = regs.NucNorm(regs.norm_l2, regs.norm_l2)
assert N_l2l2.norm_mat is regs.norm_s1
```

Notice the last assertion. Since the $\ell_2 \otimes \ell_2$ nuclear norm corresponds to the Schatten 1-norm (trace norm) on matrices, the `NucNorm` class automatically assigns the appropriate `norm_mat` method. For nuclear norms with no such simple alias, the `norm_mat` method is set to `None`. The flexibility of using objects for nuclear norms allows us to make such a determination *outside* of the solver. That is, the solver remains agnostic to the implementation; it simply calls the appropriate `norm` method of the regularizer object.

For the nuclear norms without simple aliases in CVXPY, we resort to alternating minimization. In Section 4.2.1 we saw two possible formulations for the non-convex nuclear norm recovery problem. Here we discuss the two subclasses of `NucNorm` that implement those formulations.

4.6.2.1 `NucNorm_Prod`

Recall the "product" formulation (4.2):

$$\underset{\boldsymbol{X}_i, \boldsymbol{Y}_i}{\text{minimize}} \quad \frac{1}{2}\left\| \boldsymbol{b} - \boldsymbol{\mu}\left(\sum_{i=1}^{r} \boldsymbol{X}_i \otimes \boldsymbol{Y}_i \right) \right\|_{\ell_2}^2 + \lambda \sum_{i=1}^{r} \|\boldsymbol{X}_i\|_X \|\boldsymbol{Y}_i\|_Y .$$

To implement this formulation, the regularizer object should compute the following for the solver:

$$\sum_i \|\boldsymbol{X}_i\|_X \|\boldsymbol{Y}_i\|_Y,$$

given the factors $\boldsymbol{X}_i$ and $\boldsymbol{Y}_i$ themselves. The `NucNorm_Prod` class implements this computation in its `norm_altmin` method.

4.6.2.2 `NucNorm_Sum`

Alternatively, we have the "sum" formulation (4.3):

$$\underset{\boldsymbol{X}_i,\boldsymbol{Y}_i}{\text{minimize}} \quad \frac{1}{2}\left\|\boldsymbol{b} - \boldsymbol{\mu}\left(\sum_{i=1}^{r} \boldsymbol{X}_i \otimes \boldsymbol{Y}_i\right)\right\|_{\ell_2}^2 + \lambda \sum_{i=1}^{r} \frac{1}{2}(\|\boldsymbol{X}_i\|_X^2 + \|\boldsymbol{Y}_i\|_Y^2).$$

To implement this formulation, the regularizer object should compute the following for the solver:

$$\sum_{i=1}^{r} \frac{1}{2}(\|\boldsymbol{X}_i\|_X^2 + \|\boldsymbol{Y}_i\|_Y^2),$$

given the factors $\boldsymbol{X}_i$ and $\boldsymbol{Y}_i$ themselves. The `NucNorm_Sum` class implements this computation in its `norm_altmin` method.

4.6.3 The `NucNorm_SDR` class

Finally we consider the semidefinite relaxations of nuclear norms we discussed in Section 3.6. Recall that for superquadratic norms X and Y, we have the following semidefinite relaxation of the nuclear norm (Definition 3.6.8):

$$R_{X,Y}(\mathcal{A}) := \inf\left\{\frac{1}{2}\left[g_X(\operatorname{diag}(\boldsymbol{W}_1)) + g_Y(\operatorname{diag}(\boldsymbol{W}_2))\right] : \begin{bmatrix}\boldsymbol{W}_1 & \boldsymbol{A} \\ \boldsymbol{A}^{\mathrm{t}} & \boldsymbol{W}_2\end{bmatrix} \succeq \boldsymbol{0}\right\},$$

where the semidefinite constraint follows from Proposition 3.6.2.

The semidefinite solver passes the diagonals of $\boldsymbol{W}_1$ and $\boldsymbol{W}_2$ to the `NucNorm_SDR` object and expects it to compute

$$\frac{1}{2}\left[g_X(\operatorname{diag}(\boldsymbol{W}_1)) + g_Y(\operatorname{diag}(\boldsymbol{W}_2))\right],$$

where g_X and g_Y are the appropriate gauge functions of the superquadratic norms X and Y.

The `NucNorm_SDR` class represents these regularizers, and we can create an instance as follows:

```
from operfact import regularizers as regs

R_l2linf = regs.NucNorm_SDR(regs.norm_l2, regs.norm_linf)
assert R_l2linf.lgauge is cvxpy.sum_entries
assert R_l2linf.rgauge is cvxpy.max_entries
```

Property	Description
`shape`	The dimension of operators in the domain
`measurementobj`	An object representing the linear measurement map
`measurementvec`	The observed measurements
`norm`	A `Regularizer` object
`penconst`	Regularization constant
`solver`	The CVXPY solver to use
`rank`	Number of dyads to use in the solver (`altminsolve` only)
`relconvergetol`	Stopping criterion (`altminsolve` only)
`maxiters`	Maximum iterations (`altminsolve` only)
`rfactorsinit`	Initialization (`altminsolve` only)

Table 4.2: **Properties of a `Problem` object.** The `Problem` object stores all of the information required by our solvers. This table lists its properties.

We see that the `NucNorm_SDR` constructor converts the provided norms into their respective gauges. It uses these in computing the objectives for `norm_sdp`.

The relaxed $\ell_\infty \otimes \ell_\infty$ nuclear norm also has the alias `MaxNorm` for convenience.

4.7 Solvers

The focus of this work is solving the operator recovery problem

$$\underset{\mathcal{A}}{\text{minimize}} \quad \frac{1}{2}\|\boldsymbol{b} - \boldsymbol{\mu}(\mathcal{A})\|_{\ell_2}^2 + \lambda N_{X,Y}(\mathcal{A}),$$

where $\boldsymbol{\mu}$ is a linear measurement map, $\boldsymbol{b}$ contains the observed measurements, $N_{X,Y}$ is the nuclear norm on $X \otimes Y$, and $\lambda > 0$ is a regularization constant.

We implement solvers in the `operfact.solvers` submodule. The solvers themselves are functions that take a `Problem` object containing all the information needed to instantiate and solve the problem.

4.7.1 The `Problem` and `SolverOutput` classes

We store the specification of an operator recovery problem using the `Problem` class. Table 4.2 lists the most important properties of this class. For more advanced options, refer to the reference manual.

Listing 1 shows the creation of a synthetic denoising problem. The resulting object `prob` is ready to be passed to our solver of choice. Some of the options apply solely to the alternating minimization solver, `altminsolve`, and we dis-

Listing 1 Denoising in `operfact`. Sample Python code to model a denoising problem with `operfact`.

```
import numpy as np
from operfact import operators, measurements, solvers
from operfact import regularizers as regs

shape = (m, n, p, q)
nfactors = r
sigma = 0.1

# Create the true operator and noisy measurements
oper = operators.RandomDyadsOperator(shape, nfactors)
measobj = measurements.IdentityMeasurement(shape)
measvec = measobj.apply(oper) + sigma*np.random.normal(size=
    measobj.nmeas)
# Create the problem instance
prob = solvers.Problem()
prob.shape = shape
prob.measurementobj = measobj
prob.measurementvec = measvec
prob.norm = regs.NucNorm_Prod(regs.norm_l2, regs.norm_l2)
prob.penconst = regs.penconst_denoise(shape, sigma, prob.norm)
prob.solver = cvxpy.SCS
prob.rank = r  # altminsolve only
prob.relconvergetol = 1e-3  # altminsolve only
prob.maxiters = 10  # altminsolve only
prob.rfactorsinit = None  # altminsolve only
```

cuss them in the next section. The solver then returns a `SolverOutput` object (or a list of such objects) with the properties outlined in Table 4.3.

4.7.2 Convex solver for matrix problems

When we have a simple closed-form representation of the nuclear norm $N_{X,Y}$, we can solve the convex operator recovery problem (4.1) directly with CVXPY. To solve the problem in Listing 1, we run

```
out = solvers.matsolve(prob)
```

The solver works directly with the matrix representation (3.6) of the operator and returns an `ArrayOperator` of the solution in `out.recovered`. Note that this solver performs no factorization itself, but recall that the examples of such closed-form nuclear norms in Sections 3.4 and 3.5 each come equipped with an optimal decomposition that is easy to compute. The solver can also return a `DyadsOperator` object using the dyadic SVD (3.7); see the reference manual for details.

Property	Description
`problem`	The original `Problem` object
`cvxpy_probs`	The CVXPY problem(s) created by the solver
`recovered`	The solution as an `ArrayOperator` or `DyadsOperator`
`objval`	The value of the objective after optimization
`setup_time`	The wall-time taken before calling CVXPY
`solve_time`	The wall-time taken by CVXPY
`total_time`	The total wall-time taken
`outer_iters`	The number of iterations taken (`altminsolve` only)
`relconvtol`	The effective convergence tolerance (`altminsolve` only)
`relchange`	The final relative objective change (`altminsolve` only)
`maxiters`	The effective iteration limit (`altminsolve` only)

Table 4.3: **Properties of a `ProblemOutput` object.** The `ProblemOutput` object stores all of the information returned by one of the solvers. This table lists its properties.

This solver proves useful as a truly convex solver for operator recovery problems. While we must resort to other techniques such as alternating minimization in most cases, the convex solver still applies to models of interest. It also can serve to benchmark the alternating solver, and we do this using denoising problems in Chapter 5. For large problems where we seek very low-rank solutions, the alternating solver can provide an advantage by limiting the problem size. Otherwise, the convex solver provides good performance and convergence guarantees without requiring the consideration of many hyperparameters.

4.7.3 Alternating minimization solver

To solve the problem in Listing 1 using alternating minimization (Algorithm 1, Section 4.2), we run

```
out = solvers.altminsolve(prob)
```

The operator in `out.recovered` is a `DyadsOperator` comprising the left factors $\boldsymbol{X}_i$ and right factors $\boldsymbol{Y}_i$ used as the decision variables in (4.2).

4.7.3.1 Initialization

The property `rfactorsinit` of the `Problem` object takes a list of right factors $\boldsymbol{Y}_i$ to use as initialization for the alternating minimization solver. If `None` is passed instead, the solver defaults to the spectral initialization described in

Section 4.2.3. The `initfrommeas` method of the measurement object allows for the required computation of its adjoint.

4.7.3.2 Stopping criteria

We measure our progress to convergence using the relative objective change

$$\frac{|f(\mathcal{A}^{i-1}) - f(\mathcal{A}^i)|}{f(\mathcal{A}^{i-1})},$$

where $\mathcal{A}^i$ is the decision variable after the ith iteration and f is the objective of the optimization problem. We stop when this relative change falls below a predetermined threshold. This is the property `relconvergetol` of the `Problem` object.

We also stop if we have failed to reach convergence after a fixed number of iterations. This is the `maxiters` property of the `Problem` object.

4.7.3.3 Solver rank

Finally, we must specify the number of dyads that the alternating minimization solver uses to construct the operator decompositions. This is the `rank` property of the `Problem` object.

4.7.4 Semidefinite representation solver

In Section 4.6.3 we discussed the implementation of semidefinite relaxations for some nuclear norms. When the X and Y norms are superquadratic, we have the semidefinite program

$$\begin{aligned} \underset{\boldsymbol{S}}{\text{minimize}} \quad & \frac{1}{2}\|\boldsymbol{b} - \boldsymbol{\mu}(\mathcal{A})\|_{\ell_2}^2 + \frac{\lambda}{2}\left[g_X(\operatorname{diag}(\boldsymbol{W}_1)) + g_Y(\operatorname{diag}(\boldsymbol{W}_2))\right] \\ \text{subject to} \quad & \boldsymbol{S} = \begin{bmatrix} \boldsymbol{W}_1 & \boldsymbol{A} \\ \boldsymbol{A}^{\mathrm{t}} & \boldsymbol{W}_2 \end{bmatrix} \\ & \boldsymbol{S} \succeq \boldsymbol{0}, \end{aligned}$$

where $\boldsymbol{A} = \operatorname{mat}(\mathcal{A})$, the matrix representation (3.6) of the decision variable $\mathcal{A}$.

The semidefinite solver `sdpsolve` finds the solution to this problem in conjunction with `NucNorm_SDR` regularizers. We can apply this solver to Listing 1 as follows.

```
prob.norm = regs.NucNorm_SDR(regs.norm_l2, regs.norm_l2)
out = solvers.sdpsolve(prob)
```

Just as with `matsolve`, the output is an `ArrayOperator`, but we can also have the solver return a `DyadsOperator` via either a dyadic SVD or the block matrices $\boldsymbol{W}_i$. See the reference manual for more details.

Chapter 5

Denoising with nuclear norms

In this chapter, we use our Python package `operfact` to complete a systematic study in denoising structured low-rank operators with nuclear norms.

5.1 Overview

We now turn our attention to denoising structured low-rank operators. Let $\mathcal{A}^\natural \in \mathbb{O}^{m\times n\otimes p\times q}$ be an operator, and assume that we observe

$$\mathcal{B} = \mathcal{A}^\natural + \sigma\mathcal{Z},$$

where $\mathcal{Z} \in \mathbb{O}^{m\times n\otimes p\times q}$ is random noise with the parameter $\sigma > 0$ controlling its scale. We wish to approximate $\mathcal{A}^\natural$ from this corrupted copy $\mathcal{B}$. To do so we solve the optimization problem

$$\underset{\mathcal{A}}{\text{minimize}} \quad \frac{1}{2}\|\mathcal{B} - \mathcal{A}\|_{\ell_2}^2 + \lambda N_{X,Y}(\mathcal{A}), \tag{5.1}$$

where the nuclear norm $N_{X,Y}$ is a regularizer, and $\lambda \geq 0$ is a penalty constant controlling the extent to which we remain faithful to our observations while promoting structured solutions.[1]

As we have discussed, $N_{X,Y}(\mathcal{A})$ simultaneously encodes the complexity of the individual factors of $\mathcal{A}$—with respect to the normed spaces X and Y—as well as the total complexity incurred by forming $\mathcal{A}$ as a superposition of those factors. That is, $N_{X,Y}$ serves to promote solutions composed using a small number of dyads whose factors each have low complexity in the respective norms of X and Y.

[1] For simplicity, we forego the explicit use of linear measurement maps in this formulation. We address this when discussing the implementation of this problem in `operfact`.

Operator shape: $4 \times 4 \otimes 4 \times 4$	**Rank:** 1	**SNR:** 15dB	**Solver:** `altmin` (`NucNorm_Sum`)	**Solver rank:** 16	**Tolerance:** $\epsilon = 5 \times 10^{-4}$																					
Factors	ℓ_1, ℓ_1	ℓ_1, ℓ_2	ℓ_1, ℓ_∞	ℓ_1, S_1	ℓ_1, S_∞	ℓ_2, ℓ_1	ℓ_2, ℓ_2	ℓ_2, ℓ_∞	ℓ_2, S_1	ℓ_2, S_∞	ℓ_∞, ℓ_1	ℓ_∞, ℓ_2	ℓ_∞, ℓ_∞	ℓ_∞, S_1	ℓ_∞, S_∞	S_1, ℓ_1	S_1, ℓ_2	S_1, ℓ_∞	S_1, S_1	S_1, S_∞	S_∞, ℓ_1	S_∞, ℓ_2	S_∞, ℓ_∞	S_∞, S_1	S_∞, S_∞
gaus, gaus	1.0	1.4	0.5	1.3	1.0	1.0	**6.1**	2.1	3.3	3.6	0.1	1.9	0.1	0.6	0.7	0.8	3.7	0.8	2.0	1.8	0.7	3.6	0.9	1.7	1.9
gaus, lr	1.2	1.1	-0.2	3.2	0.4	1.6	5.8	1.0	**7.4**	3.0	0.6	1.8	-0.2	2.7	0.3	1.3	3.6	0.1	5.0	1.0	1.1	3.3	0.1	4.4	1.1
gaus, orth	0.8	1.5	0.9	1.1	2.7	1.1	6.4	2.9	2.6	**7.8**	0.2	2.3	0.8	0.6	3.1	0.8	4.1	1.4	1.7	4.8	0.6	3.9	1.4	1.3	4.8
gaus, sign	0.4	1.4	3.8	1.3	1.1	0.6	6.2	**8.4**	3.7	4.5	-0.1	1.9	2.2	0.6	0.9	0.5	4.0	5.2	2.2	2.1	0.4	3.8	4.7	1.7	2.2
lr, lr	1.5	1.9	0.5	4.0	0.8	1.7	6.3	1.8	7.8	3.7	0.3	1.8	-0.2	2.7	0.2	3.4	7.9	2.8	**9.5**	4.8	0.6	3.9	0.4	4.8	1.0
lr, orth	1.3	1.8	1.1	1.4	3.2	1.2	6.3	2.9	2.9	7.4	0.2	2.1	0.5	0.5	2.9	2.8	7.7	3.9	4.2	**9.3**	0.5	4.0	1.2	1.2	4.7
orth, orth	1.0	1.6	0.9	1.1	3.0	1.3	6.7	3.0	2.9	8.1	0.7	2.9	1.2	1.1	4.0	0.7	3.6	1.3	1.6	5.0	2.4	7.0	3.9	4.1	**9.5**
sign, lr	0.7	0.8	-0.2	3.1	0.3	1.4	6.1	1.7	7.8	3.3	3.2	6.9	1.6	**8.7**	3.9	1.1	4.0	0.5	5.5	1.3	1.1	3.9	0.7	5.1	1.3
sign, orth	0.4	1.1	0.7	0.6	2.8	1.2	6.2	3.2	2.6	7.6	2.9	7.0	3.6	4.0	**8.5**	0.9	4.4	1.6	1.7	5.3	0.8	3.9	1.7	1.3	4.9
sign, sign	0.1	1.2	3.5	1.1	0.7	0.9	6.6	8.8	4.1	3.6	2.9	7.5	**11.7**	5.1	3.4	0.7	4.7	5.9	2.7	2.0	0.6	3.9	4.6	2.0	1.8
sparse, gaus	5.8	**8.5**	6.0	7.0	8.3	1.1	6.8	2.3	3.7	4.4	0.2	1.6	0.2	0.2	0.3	2.8	8.3	3.3	4.7	5.9	0.5	4.5	0.8	1.5	1.7
sparse, lr	5.9	7.7	5.6	**10.5**	7.0	1.3	6.1	1.9	7.9	3.4	-0.0	1.2	-0.0	2.0	-0.0	3.0	7.5	2.9	9.2	4.5	0.4	3.7	0.5	4.6	1.0
sparse, orth	5.3	7.6	6.5	6.0	**10.6**	0.8	6.0	2.9	2.6	7.3	-0.0	1.2	-0.0	-0.0	1.7	2.6	7.3	3.8	3.8	9.0	0.2	3.6	1.2	0.8	4.6
sparse, sign	5.4	7.7	**12.6**	6.5	7.4	0.6	6.0	8.1	3.3	3.9	0.0	1.3	1.9	0.0	0.1	2.5	7.3	10.0	4.3	5.1	0.1	3.7	4.7	1.2	1.5
sparse, sparse	**15.3**	8.2	4.0	10.8	7.4	8.6	6.6	0.7	8.1	4.1	4.2	1.3	-0.0	2.1	-0.0	10.7	8.1	1.9	9.6	4.8	7.8	3.9	-0.0	4.8	1.2

Table 5.1: **A preview of the results.** This table shows the gain in dB (5.9) for denoising $4 \times 4 \otimes 4 \times 4$ rank-1 operators with various combinations of factor structure and nuclear norm. Bold numbers indicate the highest value(s) in each row (i.e., the nuclear norm that empirically denoises the factor structure best). We notice that nuclear norms tuned to the factor structure of the operator perform best. The full discussion of these results is in Section 5.5. (Key: gaus = Gaussian, lr = low-rank, orth = orthogonal.)

5.1.1 A preview of the results

We hypothesize that:

> Matching nuclear norms to the factor structures of the true signal $\mathcal{A}^\natural$ provides superior denoising results.

And, indeed, Table 5.1 demonstrates this principle across a range of factor structures and nuclear norms! The rows of the table correspond to combinations of factor structures, and the columns correspond to nuclear norms. The values of the table reflect the denoising performance.[2] Higher values correspond to better performance, and the bold value in each row highlights the nuclear norm that results in the best performance.

For instance, if $\mathcal{A}^\natural = \boldsymbol{X}^\natural \otimes \boldsymbol{Y}^\natural$, where $\boldsymbol{X}^\natural$ and $\boldsymbol{Y}^\natural$ are both sparse, then—as expected—the $\ell_1 \otimes \ell_1$ nuclear norm is the best regularizer for denoising. In the case where $\boldsymbol{X}^\natural$ and $\boldsymbol{Y}^\natural$ are both orthogonal matrices, the $S_\infty \otimes S_\infty$ nuclear norm works best. This matches the intuition from Table 3.1 of atomic norms.

The bulk of this chapter describes a systematic study to validate our Python package `operfact` and its alternating minimization solver. This culminates in a discussion of results like the above in Section 5.5.

5.1.2 Roadmap

First we review relevant theoretical results on denoising in Section 5.2. In Section 5.3 we restate the problem and show its implementation under the `operfact` Python package. We discuss our systematic study of denoising in Section 5.4. Finally, we present our main results in Section 5.5.

Appendix B provides details on the experimental protocol along with additional figures and tables.

5.2 Theoretical considerations

In this section we review some results from the literature that are relevant to denoising with nuclear norms.

[2] They show the gain in decibels (5.9), and we explain these terms shortly.

5.2.1 Atomic norm denoising

Bhaskar et al. [BTR13] consider the use of atomic norms [CRPW12] as regularizers in denoising problems. Since nuclear norms are themselves atomic norms, their general results concerning atomic norm denoising apply to nuclear norm denoising as well.[3] Their work provides upper bounds on the denoising performance in terms of the nuclear norm of the true signal, and we restate this result in our notation.

Fact 5.2.1 ([BTR13, Thm. 1])**.** *For $\lambda \geq \mathbb{E}\, N^*_{X,Y}(\mathcal{Z})$ and $\widehat{\mathcal{A}}$ the solution to the nuclear norm denoising problem (5.1),*

$$\mathbb{E}\|\mathcal{A}^\natural - \widehat{\mathcal{A}}\|_{\ell_2}^2 \leq \sigma\lambda N_{X,Y}(\mathcal{A}^\natural).$$

$N^*_{X,Y}$ is the dual norm of $N_{X,Y}$ as given in Proposition 3.3.8.

This result links the expected absolute squared error of our solution to the penalty constant and the nuclear norm of our true operator.

Note that the condition $\lambda \geq \mathbb{E}\, N^*_{X,Y}(\mathcal{Z})$ is equivalent to $\sigma\lambda \geq \mathbb{E}\, N^*_{X,Y}(\sigma\mathcal{Z})$. Because of this, we find it useful to separate the noise level σ explicitly from the noise process $\mathcal{Z}$. This makes our choice of λ independent of the noise level.

5.2.2 The geometric view

Meanwhile, Chandrasekaran and Jordan [CJ13] consider constrained denoising problems and provide improved bounds that depend on the geometry of the constraint set. In our setting, this means solving the problem

$$\underset{\mathcal{A}}{\text{minimize}} \quad \frac{1}{2}\|\mathcal{B} - \mathcal{A}\|_{\ell_2} \quad \text{subject to} \quad N_{X,Y}(\mathcal{A}) \leq N_{X,Y}(\mathcal{A}^\natural), \tag{5.2}$$

where the notation is just as in (5.1).

The solution to (5.2) is the projection of $\mathcal{B}$ onto the sublevel set of the $X \otimes Y$ nuclear norm at $\mathcal{A}^\natural$. As illustrated in Figure 5.1, the error of this projection depends on the noise level (i.e., the expected distance between $\mathcal{B}$ and $\mathcal{A}^\natural$) as well as the "size" of the sublevel set of $N_{X,Y}$ at $\mathcal{A}^\natural$.

To be more precise, we define the *descent cone*.

[3] A nuclear norm is the gauge function of the convex hull of unit-norm dyads. See Section 3.3.6.

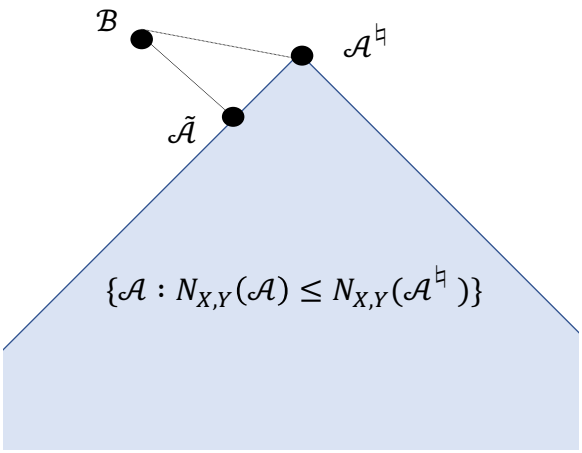


Figure 5.1: **Constrained denoising.** This figure illustrates the constrained denoising problem (5.2). The shaded blue set represents the sublevel set of the nuclear norm $N_{X,Y}$ at the true signal $\mathcal{A}^\natural$. The point $\mathcal{B}$ is the noisy observation, and its orthogonal projection $\widetilde{\mathcal{A}}$ onto the sublevel set is the solution. The error is the distance between $\widetilde{\mathcal{A}}$ and $\mathcal{A}^\natural$. Notice that the shape of the sublevel set controls the error.

Definition 5.2.2 (Descent cone)**.** Let $f : \mathbb{R}^d \to \mathbb{R} \cup \{+\infty\}$ be a proper convex function. The *descent cone* of f at the point $\boldsymbol{v}$ is the convex cone

$$\mathcal{D}(f; \boldsymbol{v}) := \bigcup_{\tau \geq 0} \{\boldsymbol{w} \in \mathbb{R}^d : f(\boldsymbol{v} + \lambda \boldsymbol{w}) \leq f(\boldsymbol{v})\}.$$

This is the convex cone in $\mathbb{R}^d$ that contains the directions into the sublevel set of f at $\boldsymbol{v}$.

In the present example, we care about the "size" of $\mathcal{D}(N_{X,Y}; \mathcal{A}^\natural)$. Chandrasekaran and Jordan [CJ13] measure this quantity in terms of the *Gaussian squared-complexity* of the cone. We follow the work of Amelunxen et al. [ALMT14] and refer to it as the *statistical dimension*.[4]

Definition 5.2.3 ([ALMT14, Def. 2.1] Statistical dimension)**.** The *statistical dimension* of the convex cone $\mathcal{C} \subseteq \mathbb{R}^d$ is

$$\delta(\mathcal{C}) := \mathbb{E}\left[\|\boldsymbol{\Pi}_{\mathcal{C}}(\boldsymbol{g})\|_{\ell_2}^2\right] \quad \text{where} \quad \boldsymbol{g} \sim \textsc{normal}(\mathbf{0}, \mathbf{I}_d),$$

[4] Note that the definition of *Gaussian squared-complexity* [CJ13, Def. 3] is ambiguous with respect to a squaring inside the expectation, but the resulting Gaussian squared-complexity of a convex cone [CJ13, Coro. 6] coincides with the statistical dimension formulation provided in Fact 5.2.5.

and $\mathbf{\Pi}_{\mathscr{C}}$ is the projection operator onto $\mathscr{C}$.

In short, the statistical dimension allows us to measure sizes of convex cones in a way that is compatible with the notion of dimension for linear subspaces.

We can now restate the denoising bound from Chandrasekaran and Jordan [CJ13] in our notation.

Fact 5.2.4 ([CJ13, Prop. 4])**.** *Assume that the error operator $\mathcal{Z}$ has independent, standard Gaussian entries. Then the solution $\widehat{\mathcal{A}}$ of the constrained denoising problem* (5.2) *obeys the bound*

$$\mathbb{E}\|\mathcal{A}^\natural - \widehat{\mathcal{A}}\|_{\ell_2}^2 \leq \sigma^2 \cdot \delta\big(\mathscr{D}(N_{X,Y}; \mathcal{A}^\natural)\big).$$

This bound still depends on the complexity of the true signal $\mathcal{A}^\natural$, but it depends on the shape of the regularizer and not its value—as opposed to Fact 5.2.1. This greatly improves the error bounds. To wit, consider denoising with the ℓ_1 to promote sparsity. The size of the descent cone of the ℓ_1 norm at a vector depends solely on its sparsity (see [CRPW12; ALMT14]) while the norm itself increases in magnitude as the scale of the vector increases.

Note that Fact 5.2.4 depends on the noise process $\mathcal{Z}$ being Gaussian given the use of the statistical dimension to measure the size of the descent cone. Even so, the key point here is the connection between the denoising performance and the geometry of the regularizer around the target signal. A regularizer that is more "pointed" around signals of interest performs better.

5.2.3 Worst-case performance

Finally, Oymak and Hassibi [OH15] present a result similar to Fact 5.2.4 for the regularized problem. Before examining their result, we introduce some additional formulations for the descent cone and statistical dimension.

Fact 5.2.5 (Alternative formulations [McC13, Prop. 3.7])**.** *Let $\mathscr{C} \subseteq \mathbb{R}^d$ be a convex cone with polar $\mathscr{C}^\circ$, then*

$$\delta(\mathscr{C}) = \mathbb{E}\left[\operatorname{dist}(\boldsymbol{g}, \mathscr{C}^\circ)^2\right] \quad \text{where} \quad \boldsymbol{g} \sim \operatorname{normal}(\mathbf{0}, \mathbf{I}_d).$$

Furthermore, we can write the polar of the descent cone of a proper, convex function $f : \mathbb{R}^d \to \mathbb{R} \cup \{+\infty\}$ in terms of its subdifferential $\partial f(\boldsymbol{v})$,

$$\mathscr{D}(f; \boldsymbol{v})^\circ = \bigcup_{\lambda \geq 0} \lambda \partial f(\boldsymbol{v}).$$

That is, the polar of the descent cone is the cone of scaled subdifferentials.

We can now state their result in our notation.

Fact 5.2.6 ([OH15, Thm. 1.1])**.** *For $\lambda \geq 0$ and $\widehat{\mathcal{A}}$ the solution to the nuclear norm denoising problem (5.1), the worst-case normalized mean squared error is*

$$\max_{\sigma>0} \frac{\mathbb{E}\|\mathcal{A}^\natural - \widehat{\mathcal{A}}\|_{\ell_2}^2}{\sigma^2} = \mathbb{E}\left[\operatorname{dist}(\mathcal{G}, \lambda\partial N_{X,Y}(\mathcal{A}^\natural))^2\right],$$

where $\mathcal{G}$ has the same dimensions as $\mathcal{A}^\natural$ with independent standard Gaussian entries, and $\lambda\partial N_{X,Y}(\mathcal{A}^\natural)$ is the subdifferential of $N_{X,Y}$ at $\mathcal{A}^\natural$ scaled by λ.

Furthermore, the maximum is achieved as $\sigma \to 0$.

Using Fact 5.2.5, we can relate this to 5.2.4. In particular, $\lambda\partial f(\boldsymbol{v}) \subseteq \mathcal{D}(f;\boldsymbol{v})^\circ$ for all $\lambda \geq 0$, and so

$$\delta\big(\mathcal{D}(f;\boldsymbol{v})\big) \leq \mathbb{E}\left[\operatorname{dist}(\boldsymbol{g}, \lambda\partial f(\boldsymbol{v}))^2\right].$$

The intuition is that a larger subdifferential of $N_{X,Y}$ at $\mathcal{A}^\natural$ results in smaller denoising error by reducing the expected distance between it and a Gaussian vector. Given the polar relationship between the subdifferential and the descent cone, this matches entirely with our intuition that smaller descent cones correspond with smaller error.

Additionally, this worst-case error occurs as the noise level tends to zero. This may be surprising, but it has a geometric interpretation. Refer back to the geometric view of denoising in Figure 5.1. For observations with low noise (i.e., near the true signal) the projection very closely conforms to the shape of the descent cone around the target signal $\mathcal{A}^\natural$. Therefore measuring the width of the descent cone also reveals how far away the projection is likely to fall from the true signal.

As the noise level increases, however, the local geometry is less critical and we may project to a point with (relatively) less noise by happy accident. The strongest case in support of using particular nuclear norms will therefore be found in the low-noise regime, and we address this point further in Section 5.4.2.

Note that while the geometry of descent cones exactly determines the denoising performance in our synthetic problems, calculating the statistical dimensions of these cones presents its own challenges. But upper bounds for interesting cases do exist [CRPW12; ALMT14]. These intuitions, however, lead us to believe

that the nuclear norms we construct for particular signal structures *do* have favorable geometry. We seek to confirm this through our numerical work.

5.2.4 A connection with linear inverse problems

Before moving on, we should note that the performance of nuclear norms in denoising problems has ramifications for their ability to recover operators in linear inverse problems.

Let $\boldsymbol{\mu}\colon \mathbb{O}^{m\times n\otimes p\times q} \to \mathbb{R}^s$ be a random Gaussian measurement map with $s < mnpq$.[5] Assume that we have observations $\boldsymbol{b} = \boldsymbol{\mu}(\mathcal{A}^\natural)$, and let f be a proper convex function on $\mathbb{O}^{m\times n\otimes p\times q}$.

We formulate a regularized linear inverse problem to recover $\mathcal{A}^\natural$:

$$\underset{\mathcal{A}}{\text{minimize}} \quad f(\mathcal{A}) \quad \text{subject to} \quad \boldsymbol{\mu}(\mathcal{A}) = \boldsymbol{b},$$

Amelunxen et al. [ALMT14, Thm. II] prove that, roughly speaking, this convex problem recovers $\mathcal{A}^\natural$ when $s > \delta\big(\mathcal{D}(f;\mathcal{A}^\natural)\big)$ and fails for s smaller.

Recall that the statistical dimension $\delta\big(\mathcal{D}(f;\mathcal{A}^\natural)\big)$ determines the worse-case error in the constrained denoising problem (Fact 5.2.4). Oymak and Hassibi [OH15] show that this quantity also roughly corresponds to the worst-case error of the regularized denoising problem (with optimal λ) in Fact 5.2.6. Together, this set of results confirms the connection between these problems observed in [DJM13].

A study of the denoising performance of nuclear norms, therefore, reveals something about their geometry around structured signals, and this has ramifications beyond the denoising problems themselves.

5.3 Nuclear norm denoising with `operfact`

In this section we restate the denoising problem and provide sample code to illustrate how we translate the mathematical formulation into the `operfact` Python package. We also discuss the choices of certain solver parameters.

[5] By this we mean that each entry of $\boldsymbol{\mu}(\mathcal{A})$ is the inner product between $\mathcal{A}$ and a known operator with independent standard Gaussian entries.

5.3.1 The nuclear norm denoising problem

Let $\mathcal{A}^\natural \in \mathbb{O}^{m\times n\otimes p\times q}$ be the true operator, and let $\boldsymbol{\mu}$ be the linear measurement map such that $\boldsymbol{\mu}(\mathcal{A}^\natural) = \operatorname{vec}(\mathcal{A}^\natural)$. Assume that we observe

$$\boldsymbol{b} = \boldsymbol{\mu}(\mathcal{A}^\natural) + \sigma \boldsymbol{z},$$

where $\boldsymbol{z} \in \mathbb{R}^{mnpq}$ is additive noise with independent, standard Gaussian entries. This is exactly the situation we described in the beginning of this chapter except that we have now vectorized the observations.

To recover $\mathcal{A}^\natural$ from $\boldsymbol{b}$, we solve

$$\underset{\mathcal{A}}{\text{minimize}} \quad \frac{1}{2}\|\boldsymbol{b} - \operatorname{vec}(\mathcal{A})\|_{\ell_2}^2 + \sigma\lambda N_{X,Y}(\mathcal{A}), \tag{5.3}$$

where $N_{X,Y}$, the nuclear norm on $X \otimes Y$, serves as the regularizer, and $\lambda \geq 0$ is the *penalty constant*. Again, this corresponds exactly to the problem (5.1). We can solve (5.3) using `operfact`, and the measurement map $\boldsymbol{\mu}$ is implemented by the `IdentityMeasurement` object (Section 4.5.2).

In Python code, this becomes:

```
from operfact import measurements, regularizers, solvers

prob = solvers.Problem()
prob.shape = (m, n, p, q)
prob.measurementobj = measurements.IdentityMeasurement(shape)
prob.measurementvec = b  # the observations
prob.norm = regularizers.NucNorm_Prod(X, Y)  # the X,Y nuclear
    norm
prob.penconst = LAMBDA
```

Before solving this problem, however, we must carefully consider the choice of the penalty constant λ.

5.3.2 The penalty constant

Even if we use the "best" regularizer to denoise our signal, a poor choice of λ will lead to poor denoising. Fact 5.2.1 provides some guidance in our selection. In particular we note that we can minimize the error bound in that result by choosing

$$\lambda = \mathbb{E}\, N_{X,Y}^*(\mathcal{Z}). \tag{5.4}$$

Operator generation	**Noise generation**	**Solver options**
Operator shape	Noise level (σ)	Regularizer ($N_{X,Y}$)
Operator rank		Penalty constant (λ)
Factor structures		`operfact` solver
		`altminsolve` only:
		Number of dyads in solution
		Relative convergence threshold
		Max. outer iterations

Table 5.2: **Denoising problem parameters.** These are the parameters we must specify for each of the three stages in the design and solution of our synthetic denoising problem.

That is, the choice of the penalty constant depends on the expected dual norm of the noise.

In the best cases we can compute this expectation over Gaussian $\mathcal{Z}$ exactly. In other cases we can compute the dual norm $N^*_{X,Y}(\mathcal{Z})$ easily and approximate the expectation. In the worst cases computing this dual norm itself requires solving an intractable optimization problem.

To facilitate the selection of λ, we provide the helper function `penconst_denoise` in `operfact.regularizers` to compute our best guess for λ given the dimensions of $\mathcal{Z}$ and the chosen nuclear norm $N_{X,Y}$. Details on this function and its logic are included in Section B.2 of the appendix.

Note that this result does not depend on a particular noise process $\mathcal{Z}$. While we are assuming isotropic Gaussian noise, this is not a requirement. We also recognize that in practice the exact noise process $\mathcal{Z}$ and its power may not be known, but any reasonable approximation of $\mathbb{E}\, N^*_{X,Y}(\mathcal{Z})$ should be helpful. In our synthetic experiments, we will verify that this approach indeed makes good choices for λ.

5.4 A systematic study

This chapter examines the generation and solution of synthetic denoising problems using nuclear norms. We have a number of parameters, however, that we must choose in this process. Table 5.2 lists them in three categories: operator generation, noise generation, and solver options.

We test a large combination of these parameters on small operators. This ex-

periment, detailed in Section B.1 of the appendix, allows us to answer some important questions surrounding our methodology. Namely, we ask:

- Are our choices for the penalty constant λ good?
- How does the noise level affect solver performance?
- Does the alternating minimization solver converge?
- Is the alternating minimization solver reliable?

Before evaluating the effect of matching nuclear norms with factor structure, we address these issues.

5.4.1 The penalty constant

Let λ_0 be the penalty constant computed in (5.4) as

$$\lambda_0 := \mathbb{E}\, N^*_{X,Y}(\mathcal{Z}),$$

where $\mathcal{Z}$ is Gaussian noise that corrupts the observations of our true signal. The `penconst_denoise` function described in Section B.2 of the appendix calculates (or estimates) λ_0.

In the initial experiment we consider penalty constants

$$\lambda_j = 2^{-j}\lambda_0, \tag{5.5}$$

where we call j the *offset*. We test over a small range of offsets around $j = 0$ and measure the absolute squared error of denoising

$$\|\mathcal{A}^\natural - \widehat{\mathcal{A}}\|^2_{\ell_2}, \tag{5.6}$$

where $\widehat{\mathcal{A}}$ is the result obtained from the solver.

Note that the offset $j = -\infty$ corresponds to setting the penalty constant $\lambda = 0$. We test this condition—corresponding to no regularization—as a sort of control.

Our initial experiment, described fully in Section B.1 of the appendix, denoises randomly-generated structured operators in $\mathbb{O}^{4\times4\otimes4\times4}$ with various nuclear

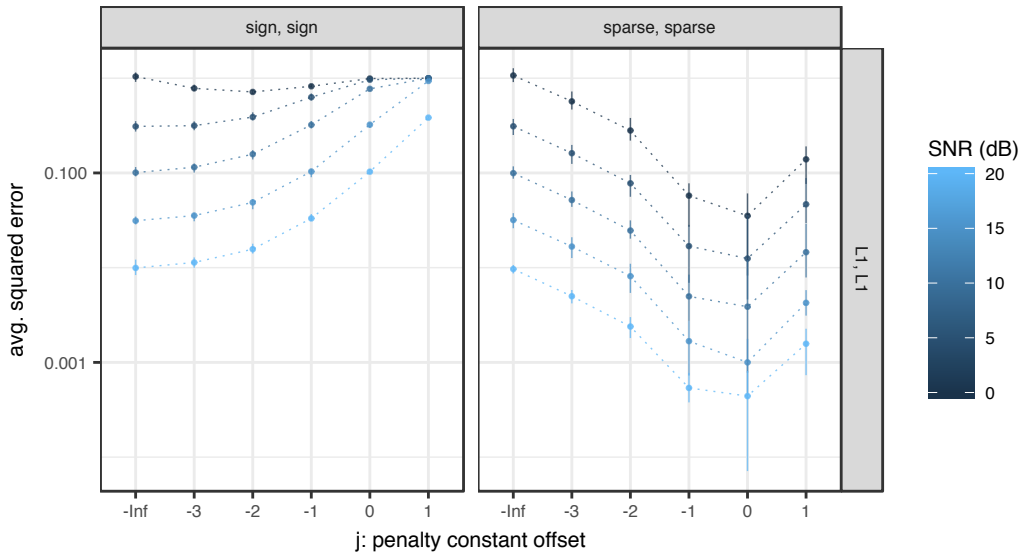


Figure 5.2: **Average error vs. penalty constant, ℓ_1 norm.** Each panel plots the average squared error (5.6) of the denoising procedure over 10 trials versus the penalty constant offset j (5.5) using the ℓ_1 norm as a regularizer at various SNR (5.7). Lighter hues correspond to higher SNR (less noise). Error bars show the minimum and maximum error over the trials. We facet the figure by factor structure. All tests were performed with the convex solver `matsolve`.

norms as regularizers. We denote the structure of the operator using the notation $(\cdot, \cdot)$ to indicate separately the structures of the left and right factors. For instance, a rank-r (sparse, sign) operator takes the form

$$\mathcal{A} = \sum_{i=1}^{r} \boldsymbol{X}_i \otimes \boldsymbol{Y}_i,$$

where the $\boldsymbol{X}_i$ are random 1-sparse matrices, and the $\boldsymbol{Y}_i$ are random sign matrices (i.e., their entries take the values ± 1 with equal probability). Section B.1.2 of the appendix describes the random operator generation in more detail along with all of the factor structures we consider.

In Figure 5.2 we show the results when using the $\ell_1 \otimes \ell_1$ nuclear norm (equivalent to the vector ℓ_1 norm) to denoise rank-1 (sparse, sparse) and (sign, sign) operators. We show the average squared error (5.6) measured over 10 trials at each offset with various noise levels. This test uses the convex solver `matsolve` (Section 4.7.2).

In the right panel, where we denoise a sparse operator with the ℓ_1 norm, we see that the lowest error occurs at offset $j = 0$. And even though the error rises with

the noise level[6], we see that the theory provides good guidance in computing λ_0.

In the left panel, however, we see that regularizing a (sign, sign) operator with the ℓ_1 norm proves fruitless. While this result is expected, it may lead us to question whether our choice for λ_0 is actually appropriate here. The theory tells us that the calculation of the penalty constant depends on the noise process and *not* on the signal itself. Therefore, the success with sparse operators is enough to satisfy us with the calculation of λ_0 for the ℓ_1 norm.

We have examined such results for all combinations of nuclear norm and factor structure that we consider in this chapter. They demonstrate that our estimate for λ_0 is adequate in allowing us to find an optimal penalty constant across all our regularizers. For completeness, we include the full set of results in the appendix for the convex solver `matsolve` (Figure B.1), the SDP solver `sdpsolve` (Figure B.2), and the alternating minimization solver `altminsolve` (Figure B.7).

5.4.2 The noise level

Recall from Fact 5.2.6 that the worst-case denoising performance occurs as the noise level tends to zero. That is, the relative value of using any one nuclear norm over any other is best judged in regimes with very little noise. We must also recognize, however, that our numerical solvers place practical limits on the noise floor. Solving problems accurately in low-noise regimes requires greater solver precision. We examine this phenomenon here.

Setting the noise level requires choosing the scale $\sigma \geq 0$ in our measurements

$$\boldsymbol{b} = \operatorname{vec}(\mathscr{A}^\natural) + \sigma \boldsymbol{z},$$

where $\boldsymbol{z} \in \mathbb{R}^{mnpq}$ has independent standard Gaussian entries.

Instead of working with the scale σ, we measure the noise level using the signal-to-noise ratio (SNR) in decibels (dB):

$$\text{SNR} := 10 \log_{10} \left(\frac{\|\mathscr{A}^\natural\|_{\ell_2}^2}{\|\sigma \mathscr{Z}\|_{\ell_2}^2} \right) \approx 10 \log_{10} \left(\frac{\|\mathscr{A}^\natural\|_{\ell_2}^2}{\sigma^2 \cdot mnpq} \right). \tag{5.7}$$

[6] We define the signal-to-noise ratio (SNR) in equation (5.7) of the next section. For now, note that lower SNR corresponds with a higher noise level.

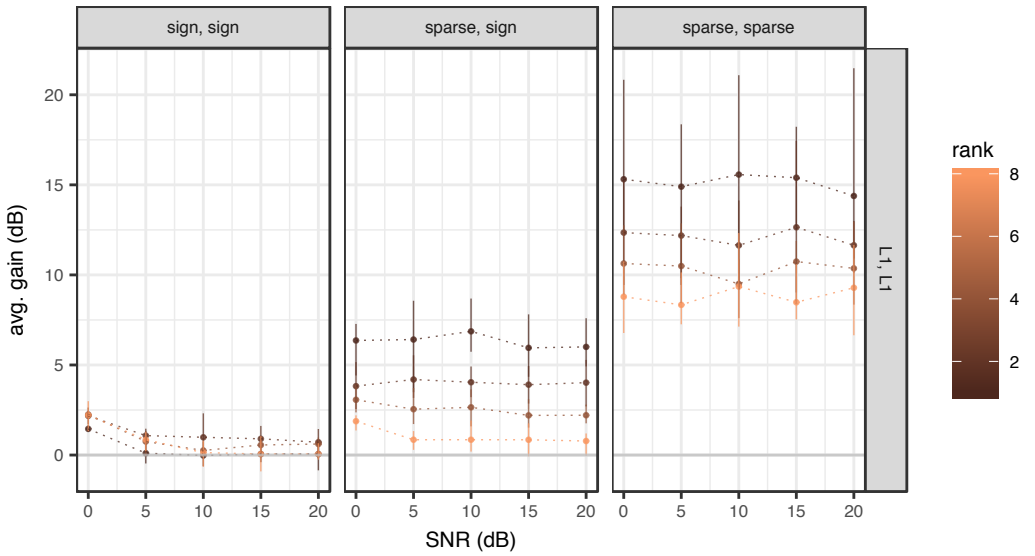


Figure 5.3: **Average gain vs. SNR, ℓ_1 norm.** Each panel plots the average gain (5.9) of the denoising procedure over 10 trials versus the SNR (5.7) using the ℓ_1 norm as a regularizer at various operator ranks. Lighter hues correspond to higher-rank operators. Error bars show the minimum and maximum gain over the trials. We facet the figure by factor structure. All tests were performed with the convex solver `matsolve`.

Maintaining a fixed SNR allows for easier comparisons between signals of different power levels.[7]

After we obtain the estimate $\widehat{\mathcal{A}}$ by solving the denoising problem, we can measure its error using the recovery signal-to-distortion ratio (RSDR)

$$\text{RSDR} := 10 \log_{10} \left(\frac{\|\mathcal{A}^\natural\|_{\ell_2}^2}{\|\widehat{\mathcal{A}} - \mathcal{A}^\natural\|_{\ell_2}^2} \right). \tag{5.8}$$

The improvement in signal quality achieved through the denoising procedure is called the *gain*, and we compute it as

$$\text{gain} := \text{RSDR} - \text{SNR} = 10 \log_{10} \left(\frac{\|\sigma \mathcal{Z}\|_{\ell_2}^2}{\|\widehat{\mathcal{A}} - \mathcal{A}^\natural\|_{\ell_2}^2} \right) \approx 10 \log_{10} \left(\frac{\sigma^2 \cdot mnpq}{\|\widehat{\mathcal{A}} - \mathcal{A}^\natural\|_{\ell_2}^2} \right). \tag{5.9}$$

We think of gain as the ratio between the noise level and the recovery error on a logarithmic scale. Positive gain indicates a benefit from denoising.

[7] But note that we normalize the Euclidean norms of the operators in this experiment anyway.

Figure 5.3 shows average gain versus SNR when using the convex solver `matsolve` with the ℓ_1 norm to denoise (sparse, sparse), (sparse, sign), and (sign, sign) operators. We display this for various operator ranks. Note that the average gain is computed over all trials using the empirically best penalty constant λ_j, where j is the offset in (5.5).

In the right panel, we see that the gain in denoising (sparse, sparse) operators with the ℓ_1 norm remains relatively constant across different levels of the SNR, with lower-rank operators showing higher gain. A similar situation occurs in the middle panel with (sparse, sign) operators even though the average gain is less than the (sparse, sparse) case. The ℓ_1 norm, however, shows very little gain when denoising (sign, sign) operators in the left panel. But note that at 0dB, we see some gain.

Figures B.3 and B.4 in the appendix show the full complement of results for `matsolve` and `sdpsolve`, respectively. Those figures are consistent with the above conclusions.

Namely, effective nuclear norms appear to remain more constant in their gain across SNRs. And at lower SNRs, poor denoisers perform *better* relative to the high SNR regime. This is as we have discussed earlier: denoising performance is relatively better when there is more noise.

Given that our goal is to assess the relative merits of different nuclear norms, we wish to examine performance at the highest SNR available. This regime provides the best assessment of the worst-case performance of the regularizer. We must be mindful, however, that higher SNR requires higher solver precision. We will return to this point in our discussion of the alternating minimization solver.

5.4.3 Convergence of the alternating minimization solver

Now we examine the convergence behavior of the alternating minimization solver. First, we consider the stopping criteria. Recall that we stop the alternating minimization when the relative objective change between outer iterations falls below a given threshold or when we complete a fixed number of outer iterations. That is, we halt when

$$\frac{|f_i - f_{i-1}|}{f_i} \leq \epsilon, \tag{5.10}$$

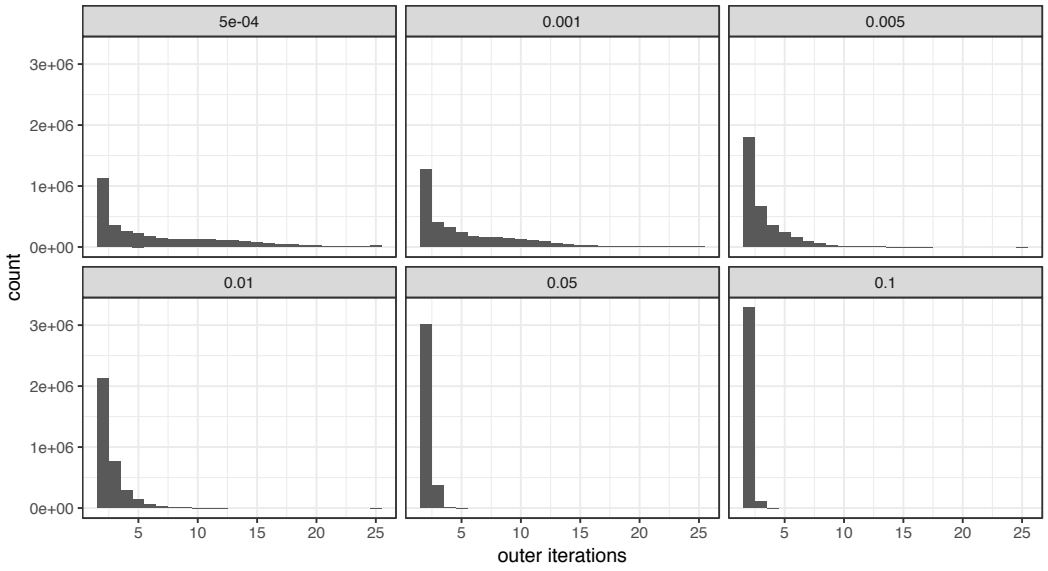


Figure 5.4: **Outer iterations by convergence tolerance.** Each facet shows a histogram of the number of individual denoising problems that reached convergence to a specified tolerance after each number of outer iterations. The title of the facet indicates the value ϵ of the relative convergence tolerance (5.10). These counts are over all 10 trials of all parameter combinations tested for the alternating minimization solver. See Table B.1 in the appendix for a listing of those parameters.

where f_i is the objective at the ith iteration, and ϵ is the *convergence tolerance* (see also Section 4.7.3.2).

In this experiment, we set the maximum number of outer iterations at 25 and test the effect of changing the convergence tolerance. Figure 5.4 shows histograms of the number of outer iterations required for each problem to converge to the six convergence tolerances we tested.

Unsurprisingly, a smaller convergence tolerance results in an increased number of iterations. We note, however, that while some of the problems at the strictest tolerance require at least the maximum number of iterations, these problems appear to be an extremely small minority. We make no claim that those problems are insignificant, and we address the choice of the tolerance later. For now, we note that the solver generally converges to our specified tolerance.

5.4.4 Reliability of the alternating minimization solver

In this section we compare the results from the nonconvex alternating minimization solver `altminsolve` (Section 4.7.3) to those of the convex `matsolve`

(Section 4.7.2) and `sdpsolve` (Section 4.7.4) solvers. Our notion of "reliability" here is the accuracy of `altminsolve` versus the convex solvers. If we can ensure agreement between these results for the cases where the convex solvers apply, we will have some confidence in using `altminsolve` in cases where it stands as our only option. Since the alternating minimization solver requires additional hyperparameters, we will also use these comparisons to judge their selection.

5.4.4.1 The penalty constant

The alternating minimization solver can handle nuclear norms that the direct convex solvers cannot. Just as before, we must ensure that the estimated penalty constant λ_0 is appropriate for these norms as well. Figure B.7 showing the average squared error in denoising versus the penalty constant offset (5.5) for all norms handled by `altminsolve` is included in the appendix. We analyze these figures just as in Section 5.4.1, and we conclude that the considered range of penalty constants is adequate.

5.4.4.2 Nuclear norm formulation

Recall from Section 4.2.1 that we have two implementations of the nuclear norm: the "product" formulation (4.2) and the "sum" formulation (4.3).

Figure 5.5 compares `matsolve` and both nuclear norm implementations of `altminsolve`. As before we plot gain (5.9) versus SNR (5.7) when denoising rank-1 (sparse, sparse), (sparse, sign), and (sign, sign) operators with the ℓ_1 norm as a regularizer, but here we plot each solver as its own series. We set `altminsolve` to use 16 dyads, and choose the convergence tolerance $\epsilon = 5 \times 10^{-4}$.

In the rightmost facet, we see that the sum formulation of the alternating minimization solver tracks the results from the convex solver very closely. The product formulation, however, shows higher gains across all noise levels. A similar situation holds for (sparse, sign) operators in the middle facet except that the sum formulation begins to perform worse when compared to the convex formulation at higher SNR. For (sign, sign) operators, we again see that the solvers do relatively better at lower SNR (higher noise).

We note that the relative change in the objective at the final iteration is comparable between both the sum and product formulation—and, in fact, the product formulation converges more quickly. In these instances, the product

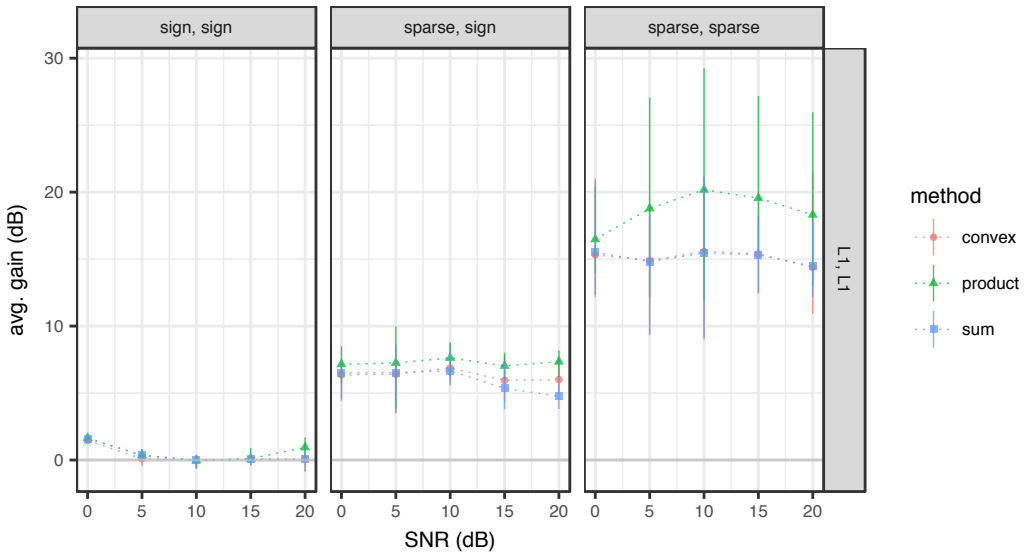


Figure 5.5: **Average gain vs. SNR, convex and nonconvex solvers.** Each panel plots the average gain (5.9) of the denoising procedure over 10 trials versus the SNR (5.7) using the ℓ_1 norm as a regularizer with the color/shape indicating the solution method. We test the convex solver `matsolve` and both the sum (4.3) and product (4.2) formulations of the alternating minimization solver `altminsolve`. Error bars show the minimum and maximum gain over the trials. We facet the figure by factor structure.

formulation requires 2 outer iterations versus the (usually) 3 from the sum formulation.

The discussion in Section 4.2.1.1 suggests that the product formulation should be more accurate, but we do not see this. With the product formulation, we do see faster convergence in wall-time, the number of outer iterations, and the time per outer iteration. Here it requires roughly 2/3rds of the time to complete each outer iteration, and it requires 2/3rds of the outer iterations to reach the same level of convergence as the sum formulation. This does not, however, explain the discrepancy in the gains.

We do not claim here that the product formulation performs more reliably even though it has higher gains. Observe that both nonconvex formulations largely agree with the convex solver in the left panel, where we denoise (sign, sign) operators.[8] At higher SNR, however, the product formulation finds some gain

[8]Note that in several cases here, the empirically best penalty constant offset is $j = -\infty$. That is, the best result occurs when we do not regularize at all. The formulations are clearly identical under this condition.

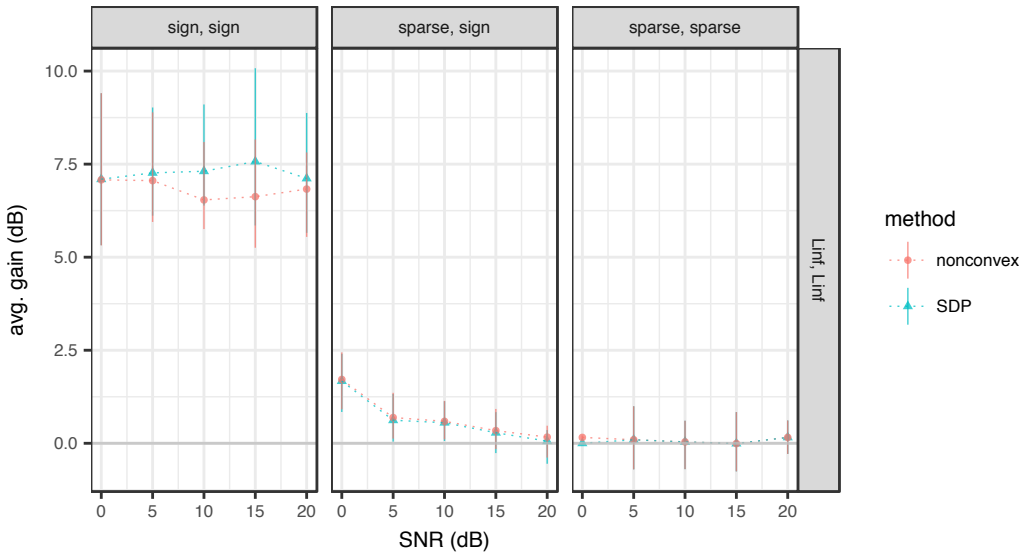


Figure 5.6: **Average gain vs. SNR, SDP and nonconvex solvers.** Each panel plots the average gain (5.9) of the denoising procedure over 10 trials versus the SNR (5.7) using the *relaxed* $\ell_\infty \otimes \ell_\infty$ nuclear norm (also called the max-norm) as a regularizer with the color/shape indicating the solution method. We test the SDP solver `sdpsolve` and the alternating minimization solver `altminsolve`. Error bars show the minimum and maximum gain over the trials. We facet the figure by factor structure.

where we believe none should exist. We believe this is due to relatively lower solver precision.

For the remainder of this chapter, we will utilize the sum formulation of alternating minimization as it agrees more closely with the convex formulation. We do, however, consider the product formulation again in later experiments as it is faster and does result in good denoising performance. In fact, some of our tests have shown that changing the quadratic loss term to just the Euclidean norm can greatly decrease solution times. Here, however, that is not the case.

Finally, we can do a similar experiment to test how the alternating minimization formulation of the semidefinite relaxations performs compared to the convex one. Figure 5.6 shows this for the semidefinite relaxation of the $\ell_\infty \otimes \ell_\infty$ nuclear norm (max-norm) on rank-1 (sparse, sparse), (sparse, sign), and (sign, sign) operators. We again see that the solvers roughly agree even though the nonconvex solver appears to perform slightly worse on the interesting (sign, sign) case.

Taken together, these experiments suggest that an SNR of 10dB or 15dB is reasonable for comparing the relative merits of different nuclear norms.

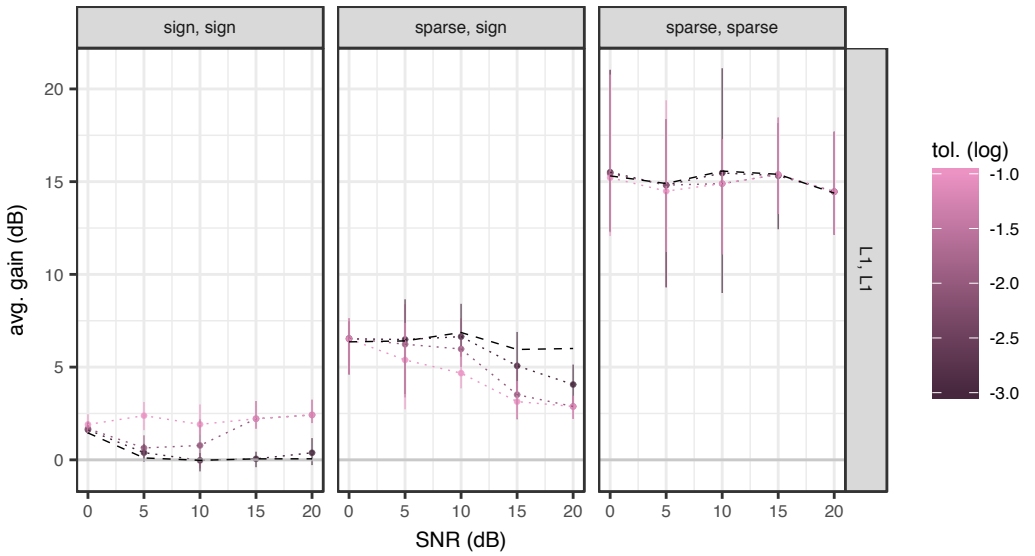


Figure 5.7: **Average gain vs. SNR, convergence tolerance.** Each panel plots the average gain (5.9) of the denoising procedure over 10 trials versus the SNR (5.7) using the ℓ_1 norm as a regularizer with the color indicating the relative convergence tolerance (5.10) of the alternating minimization solver on a logarithmic scale. Lighter hues correspond to looser tolerances. The black, dashed lines indicate the results from the convex solver for comparison. Error bars show the minimum and maximum gain over the trials. We facet the figure by factor structure.

We include the additional comparisons between `altminsolve` and `matsolve` in Figure B.5 in the appendix. Figure B.6 has the same for semidefinite relaxations with `sdpsolve`.

5.4.4.3 Convergence tolerance

In Section 5.4.3 we ensured that the alternating minimization solver is converging within our chosen tolerances before hitting the outer iteration limit. We also examined the agreement between the alternating solver with the direct convex solvers under the strictest convergence tolerance. Of course the strictest tolerance requires more computational time, and so we would like to relax it, if possible.

Figure 5.7 shows the same gain (5.9) vs. SNR (5.7) plots (using the ℓ_1 norm as a regularizer), but this time we focus on the effect of changing the relative convergence tolerance ϵ in (5.10). In the case of the (sign, sign) operators, we see that lowering the tolerance allows for the solver to find gains where we know

none should exist. That is, we see the spurious results of low solver precision.

When we consider the (sparse, sparse) operators, we see smaller differences between the tolerances. Furthermore, lowering solver precision appears to hurt gain slightly. The thing to remember here is that, in this setting, convergence happens quickly—usually within 3 outer iterations. Even if we lower the tolerance, we see that the solver converges to a higher level of precision anyway. Essentially we are seeing the effects of faster convergence rather than an ability to cope with less accuracy.

The combination of these results and those of the previous sections suggests that using $\epsilon = 10^{-3}$ at an SNR of 10dB or 15dB allows for good agreement with the convex solvers while avoiding false gains at lower SNRs. Figure B.8 in the appendix shows the complete set of gain versus SNR plots for each of our six choices of the convergence tolerance and all combinations of regularizer and factor structure.

5.4.4.4 Number of dyads

With the alternating minimization solver, we have the option of limiting the number of dyads used in the solution. This allows us to decrease the size of the optimization problems we must solve, and if we seek low-rank solutions, we can by definition represent them with fewer dyads. Doing this, however, materially alters the optimization problem we solve.

Consider the denoising problem with a rank constraint (and no nuclear norm)

$$\underset{\mathcal{A}_r}{\text{minimize}} \quad \|\mathcal{B} - \mathcal{A}_r\|_{\ell_2}^2 \quad \text{subject to} \quad \mathcal{A}_r = \sum_{i=1}^{r} \boldsymbol{X}_i \otimes \boldsymbol{Y}_i,$$

where we have indexed the decision variable $\mathcal{A}_r$ by the number of dyads allowed in the decomposition.

The solution is the best rank-r approximation to $\mathcal{B}$ (and we can obtain it using the truncated dyadic SVD (3.7)). If the true operator $\mathcal{A}^\natural$ is itself low-rank, we expect that this method may actually be effective in removing some of the noise. That is, limiting the number of dyads itself provides some denoising. This property of the solver can be useful, but in judging the utility of a given nuclear norm, it may be misleading.

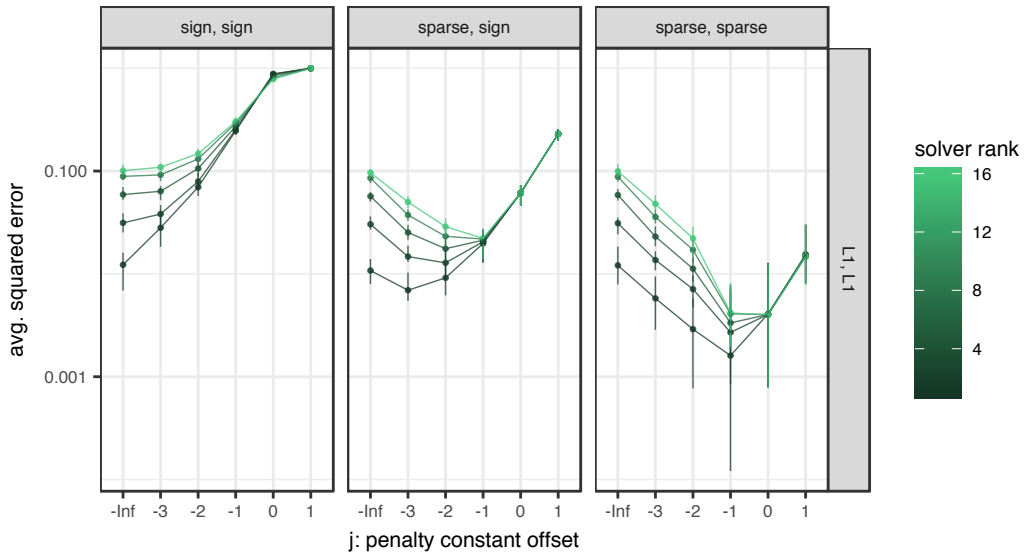


Figure 5.8: **Average error vs. penalty constant, solver rank.** Each panel plots the average squared error (5.6) of the denoising procedure over 10 trials versus the penalty constant offset j (5.5) using the ℓ_1 norm as a regularizer at various solver ranks (r). Lighter hues correspond to higher solver ranks (i.e., more dyads used in the solution). Error bars show the minimum and maximum error over the trials. We facet the figure by factor structure. All tests were performed at an SNR (5.7) of 10dB.

Consider the nuclear norm denoising problem with this rank constraint:

$$\underset{\mathcal{A}_r}{\text{minimize}} \quad \|\mathcal{B} - \mathcal{A}_r\|_{\ell_2}^2 + \sigma\lambda N_{X,Y}(\mathcal{A}_r) \quad \text{subject to} \quad \mathcal{A}_r = \sum_{i=1}^{r} \boldsymbol{X}_i \otimes \boldsymbol{Y}_i.$$

We call r the *solver rank*, and we expect that we can now remove noise while using regularizers that had no such benefit in the unconstrained case.

In short, we recognize that we must be more discerning in declaring a regularizer useful. We regard any nuclear norm that does not perform better than the truncated dyadic SVD as having no benefit. In practice, we can use the rank-constrained $\ell_2 \otimes \ell_2$ nuclear norm (trace norm) as a sensible baseline for low-rank operator denoising.

More importantly, we must reconsider the choice of penalty constant λ. Our intuition for picking λ relied on results for the denoising problem with no rank constraints. We should examine how well these choices perform in the rank-constrained case.

Figure 5.8 shows the average squared error (5.6) versus penalty constant offset

j (5.5) for denoising rank-1 (sparse, sparse), (sparse, sign), and (sign, sign) operators using the ℓ_1 norm as a regularizer. We fix the SNR (5.7) at 10dB and the convergence tolerance (5.10) at $\epsilon = 5 \times 10^{-4}$. We plot one series for each of the solver ranks tested.

First, notice that when the penalty constant offset is $-\infty$ (i.e., $\lambda = 0$), we do indeed see that lowering the number of dyads in the solver reduces the average squared error. This is simply the effect of using the truncated SVD to denoise the signals.

In the right panel, where we denoise (sparse, sparse) operators, we see that the minimum squared error occurs at the same penalty constant offset j for all of the solver ranks tested. And while reducing the solver rank does provide smaller squared error when the penalty constant is smaller than the optimal value, there is no difference when it is larger.

In the middle panel, where we denoise (sparse, sign) operators, the optimal value occurs at smaller offsets when we use lower solver rank. We posit that this is because the regularizer has less effect in light of the rank constraint. Again, at and above the optimal penalty constant from the full solver rank case, the results from all solver ranks coincide.

Lastly we consider the (sign, sign) operator. Our previous examinations have shown that the ℓ_1 norm provides no real denoising benefit to these signals. Indeed, the lowest error occurs when we simply take the rank-1 truncated SVD of the noisy observations (recall the true signal here is rank-1 as well). At larger penalty constants, the results between solver ranks again coincide *and* show worse error than no regularization at all.

Figure B.9 in the appendix shows these plots for all combinations of nuclear norm and factor structure. We see no evidence here that requires us to adjust penalty constants when the solver is rank-constrained. This allows us to take advantage of the computational speedup afforded by using fewer dyads in the solver. When comparing the relative merits of nuclear norms, however, we must ensure that our experiment controls for the gains inherent in using a rank constraint.

Note that we are generally happy to reduce error through both the choice of regularizer and the solver rank. But remember that this rank-constrained problem is inherently non-convex. Even in problems where we wish to recover a rank-r

operator, allowing the solver to use more than r dyads can prove beneficial. This has been observed in the literature [BM04; BMP08; HV15]. We will see evidence of this when we examine self-calibration problems in Chapter 7.

5.5 The main results

In this section we use the results of the small $(4 \times 4 \otimes 4 \times 4)$ experiment and an additional, larger $(16 \times 16 \otimes 16 \times 16)$ experiment described in Section B.1.6 of the appendix. We aim to answer our original question:

> Are nuclear norms effective regularizers in promoting low-rank operators with various factor structures?

5.5.1 Factor structure

Table 5.3 shows the average gain (5.9) over 10 trials for all combinations of factor structure and nuclear norm tested in two different experiments. Higher gain indicates superior denoising performance. In each row, the bold number indicates the nuclear norm providing the highest average gain.

The top table shows the results from the smaller $(4 \times 4 \otimes 4 \times 4)$ experiment with rank-1 operators with SNR 15dB, solver rank 16, and convergence tolerance $\epsilon = 5 \times 10^{-4}$. The bottom shows the same for $16 \times 16 \otimes 16 \times 16$ rank-1 operators with solver rank 16 and convergence tolerance $\epsilon = 10^{-3}$. Both sets of results use the alternating minimization solver with the "sum" formulation (4.3).

First note that the $\ell_2 \otimes \ell_2$ nuclear norm (i.e., the trace norm) performs consistently well across all factor structures. We expect that the trace norm works to promote low-rank operators, and we see this. The trace norm does not care about the factor structures.

Looking at the top table we see that the highest gain for each factor structure occurs when the nuclear norm exactly matches the factor structure. That is, for (sparse, sparse) operators, the $\ell_1 \otimes \ell_1$ nuclear norm performs best, while for (orthogonal, orthogonal) operators, the $S_\infty \otimes S_\infty$ nuclear norm performs best. We also see that partially matching the factor structure gives better results than not.

In the bottom table we see similar results, but there are a few things worth noting. First we use a solve rank of 16, smaller than the cardinality of either factor, and

Operator shape: $4 \times 4 \otimes 4 \times 4$ **Rank:** 1 **SNR:** 15dB **Solver:** `altmin (NucNorm_Sum)` **Solver rank:** 16 **Tolerance:** $\epsilon = 5 \times 10^{-4}$

Factors	ℓ_1, ℓ_1	ℓ_1, ℓ_2	ℓ_1, ℓ_∞	ℓ_1, S_1	ℓ_1, S_∞	ℓ_2, ℓ_1	ℓ_2, ℓ_2	ℓ_2, ℓ_∞	ℓ_2, S_1	ℓ_2, S_∞	ℓ_∞, ℓ_1	ℓ_∞, ℓ_2	ℓ_∞, ℓ_∞	ℓ_∞, S_1	ℓ_∞, S_∞	S_1, ℓ_1	S_1, ℓ_2	S_1, ℓ_∞	S_1, S_1	S_1, S_∞	S_∞, ℓ_1	S_∞, ℓ_2	S_∞, ℓ_∞	S_∞, S_1	S_∞, S_∞
gaus, gaus	1.0	1.4	0.5	1.3	1.0	1.0	**6.1**	2.1	3.3	3.6	0.1	1.9	0.1	0.6	0.7	0.8	3.7	0.8	2.0	1.8	0.7	3.6	0.9	1.7	1.9
gaus, lr	1.2	1.1	-0.2	3.2	0.4	1.6	5.8	1.0	**7.4**	3.0	0.6	1.8	-0.2	2.7	0.3	1.3	3.6	0.1	5.0	1.0	1.1	3.3	0.1	4.4	1.1
gaus, orth	0.8	1.5	0.9	1.1	2.7	1.1	6.4	2.9	2.6	**7.8**	0.2	2.3	0.8	0.6	3.1	0.8	4.1	1.4	1.7	4.8	0.6	3.9	1.4	1.3	4.8
gaus, sign	0.4	1.4	3.8	1.3	1.1	0.6	6.2	**8.4**	3.7	4.5	-0.1	1.9	2.2	0.6	0.9	0.5	4.0	5.2	2.2	2.1	0.4	3.8	4.7	1.7	2.2
lr, lr	1.5	1.9	0.5	4.0	0.8	1.7	6.3	1.8	7.8	3.7	0.3	1.8	-0.2	2.7	0.2	3.4	7.9	2.8	**9.5**	4.8	0.6	3.9	0.4	4.8	1.0
lr, orth	1.3	1.8	1.1	1.4	3.2	1.2	6.3	2.9	2.9	7.4	0.2	2.1	0.5	0.5	2.9	2.8	7.7	3.9	4.2	**9.3**	0.5	4.0	1.2	1.2	4.7
orth, orth	1.0	1.6	0.9	1.1	3.0	1.3	6.7	3.0	2.9	8.1	0.7	2.9	1.2	1.1	4.0	0.7	3.6	1.3	1.6	5.0	2.4	7.0	3.9	4.1	**9.5**
sign, lr	0.7	0.8	-0.2	3.1	0.3	1.4	6.1	1.7	7.8	3.3	3.2	6.9	1.6	**8.7**	3.9	1.1	4.0	0.5	5.5	1.3	1.1	3.9	0.7	5.1	1.3
sign, orth	0.4	1.1	0.7	0.6	2.8	1.2	6.2	3.2	2.6	7.6	2.9	7.0	3.6	4.0	**8.5**	0.9	4.4	1.6	1.7	5.3	0.8	3.9	1.7	1.3	4.9
sign, sign	0.1	1.2	3.5	1.1	0.7	0.9	6.6	8.8	4.1	3.6	2.9	7.5	**11.7**	5.1	3.4	0.7	4.7	5.9	2.7	2.0	0.6	3.9	4.6	2.0	1.8
sparse, gaus	5.8	**8.5**	6.0	7.0	8.3	1.1	6.8	2.3	3.7	4.4	0.2	1.6	0.2	0.2	0.3	2.8	8.3	3.3	4.7	5.9	0.5	4.5	0.8	1.5	1.7
sparse, lr	5.9	7.7	5.6	**10.5**	7.0	1.3	6.1	1.9	7.9	3.4	-0.0	1.2	-0.0	2.0	-0.0	3.0	7.5	2.9	9.2	4.5	0.4	3.7	0.5	4.6	1.0
sparse, orth	5.3	7.6	6.5	6.0	**10.6**	0.8	6.0	2.9	2.6	7.3	-0.0	1.2	-0.0	-0.0	1.7	2.6	7.3	3.8	3.8	9.0	0.2	3.6	1.2	0.8	4.6
sparse, sign	5.4	7.7	**12.6**	6.5	7.4	0.6	6.0	8.1	3.3	3.9	0.0	1.3	1.9	0.0	0.1	2.5	7.3	10.0	4.3	5.1	0.1	3.7	4.7	1.2	1.5
sparse, sparse	**15.3**	8.2	4.0	10.8	7.4	8.6	6.6	0.7	8.1	4.1	4.2	1.3	-0.0	2.1	-0.0	10.7	8.1	1.9	9.6	4.8	7.8	3.9	-0.0	4.8	1.2

Operator shape: $16 \times 16 \otimes 16 \times 16$ **Rank:** 1 **SNR:** 10dB **Solver:** `altmin (NucNorm_Sum)` **Solver rank:** 16 **Tolerance:** $\epsilon = 1 \times 10^{-3}$

Factors	ℓ_1, ℓ_1	ℓ_1, ℓ_2	ℓ_1, ℓ_∞	ℓ_1, S_1	ℓ_1, S_∞	ℓ_2, ℓ_1	ℓ_2, ℓ_2	ℓ_2, ℓ_∞	ℓ_2, S_1	ℓ_2, S_∞	ℓ_∞, ℓ_1	ℓ_∞, ℓ_2	ℓ_∞, ℓ_∞	ℓ_∞, S_1	ℓ_∞, S_∞	S_1, ℓ_1	S_1, ℓ_2	S_1, ℓ_∞	S_1, S_1	S_1, S_∞	S_∞, ℓ_1	S_∞, ℓ_2	S_∞, ℓ_∞	S_∞, S_1	S_∞, S_∞
gaus, gaus	6.9	9.5	6.9	8.3	7.4	8.9	**16.4**	9.0	10.4	14.7	6.9	9.1	6.9	7.4	7.1	8.0	11.4	7.4	11.6	9.7	7.5	13.0	7.2	9.6	9.2
gaus, lr	6.9	9.5	6.9	13.1	6.9	9.2	16.3	8.5	**21.1**	11.2	6.9	9.2	6.9	12.2	6.9	8.5	11.5	6.9	17.5	8.6	8.0	13.0	6.9	20.6	7.6
gaus, orth	6.9	9.4	6.9	7.8	11.8	8.8	16.4	10.0	11.0	**21.6**	6.9	9.1	7.3	7.9	10.1	8.0	11.5	8.1	9.7	13.1	7.8	13.3	7.9	10.2	12.8
gaus, sign	6.9	9.4	11.4	8.2	7.8	9.3	16.2	**18.9**	10.3	15.3	6.9	9.0	9.0	7.6	7.4	7.9	11.5	13.4	11.5	9.9	7.5	13.1	10.2	9.6	9.2
lr, lr	7.5	9.8	6.9	13.0	6.9	9.4	16.4	8.3	21.0	11.3	6.9	8.1	6.9	11.1	6.9	11.9	21.0	12.4	**28.0**	17.0	6.9	11.4	6.9	20.5	6.9
lr, orth	6.9	9.7	7.0	8.2	11.0	8.9	16.5	10.0	10.9	21.7	6.9	8.4	6.9	6.9	9.1	11.9	20.9	13.7	16.0	**24.5**	6.9	11.3	6.9	7.5	11.0
orth, orth	6.9	9.4	6.9	7.8	11.9	8.8	16.2	9.7	10.9	**21.3**	6.9	9.4	7.4	8.2	10.4	7.5	12.5	8.3	10.7	14.7	11.0	16.7	10.7	13.2	**21.3**
sign, lr	6.9	9.9	6.9	13.8	6.9	9.4	16.2	8.1	**21.1**	11.2	10.7	14.6	6.9	17.9	10.1	8.7	11.5	6.9	17.3	8.7	8.3	13.2	6.9	20.4	7.7
sign, orth	6.9	10.1	6.9	7.2	12.5	8.9	16.3	9.9	11.0	**21.7**	10.7	14.7	8.7	14.7	15.4	8.0	11.5	8.3	9.7	13.5	7.7	13.1	7.7	10.0	11.9
sign, sign	6.9	10.0	13.3	8.1	7.3	9.4	16.0	18.8	10.5	14.5	14.9	14.3	**21.8**	11.8	9.1	7.8	11.5	13.4	11.6	9.8	7.1	12.7	9.6	9.7	8.8
sparse, gaus	15.8	**23.9**	15.9	16.4	23.1	8.9	18.7	9.4	10.4	15.6	6.9	6.9	6.9	6.9	6.9	12.9	22.9	13.3	15.4	20.8	6.9	16.0	6.9	8.2	8.3
sparse, lr	14.5	23.7	15.4	**30.7**	21.2	9.3	18.8	8.5	21.4	11.3	6.9	6.9	6.9	10.6	6.9	12.8	22.7	12.9	27.8	18.2	6.9	16.1	6.9	20.6	6.9
sparse, orth	15.9	23.5	17.0	16.8	**25.3**	8.9	18.8	10.2	11.1	21.8	6.9	7.0	6.9	6.9	7.8	13.0	22.7	14.1	16.1	24.3	6.9	16.1	6.9	7.6	11.0
sparse, sign	16.6	23.5	**27.1**	16.3	22.9	9.3	18.7	19.2	10.3	15.1	6.9	7.0	8.1	6.9	6.9	13.1	22.6	25.8	15.3	20.4	6.9	15.9	11.5	8.1	8.2
sparse, sparse	**40.6**	24.0	15.3	30.3	22.3	23.8	19.3	7.2	22.6	11.3	17.5	7.2	6.9	11.2	6.9	30.6	22.7	10.2	27.1	18.1	23.0	16.1	6.9	20.3	6.9

Table 5.3: **The main results.** These tables show the gains in dB (5.9) for denoising rank-1 operators at all tested combinations of factor structure and nuclear norm. Bold numbers indicate the highest value(s) in each row (i.e., the nuclear norm that empirically denoises the factor structure best). The top table shows results for $4 \times 4 \otimes 4 \times 4$ operators using the alternating minimization solver with 16 dyads. The bottom tables shows results for $16 \times 16 \otimes 16 \times 16$ operators also with 16 dyads in the alternating minimization solver. (Key: gaus = Gaussian, lr = low-rank, orth = orthogonal.)

so this rank constraint in the solver provides some denoising even when the nuclear norm does not. Indeed, we see around 7dB of gain in situations here that showed no gain in the table above.

Once again the $\ell_2 \otimes \ell_2$ nuclear norm performs consistently regardless of the factor structure. Generally, matching the nuclear norm to the factor structure provides the largest gains, but we do have some discrepancies. Notably, the $\ell_2 \otimes S_\infty$ nuclear norm not only performs best with (Gaussian, orthogonal) operators but also with (orthogonal, orthogonal) and (sign, orthogonal) operators. (In the case of (orthogonal, orthogonal) operators, it does equally as well as the $S_\infty \otimes S_\infty$ nuclear norm.) Also for (sign, low-rank), the $\ell_2 \otimes S_1$ nuclear norm outperforms the $\ell_2 \otimes S_1$ nuclear norm.

We posit that this occurs because of the lower solver rank combined with denoising a rank-1 operator. That is, the regularization becomes relatively less important in denoising this operator. As noted in the next section, these discrepancies lessen as the rank of the operator increases.

We conclude that

> *Nuclear norms are effective regularizers for structured low-rank operators.*

5.5.2 Operator rank

We also want to check the behavior of nuclear norm denoising as we increase the rank of the true operator $\mathcal{A}^\natural$. Figure 5.9 plots gain (5.9) versus operator rank using the ℓ_1 norm as a regularizer with $16 \times 16 \otimes 16 \times 16$ operators. As expected, the gain decreases as the operator rank increases.

We also notice that the gain curve for the alternating minimization solver does not track the shape of the convex solver's gain curve very well here. Again, we think this is due to the fact that we use only a solver rank of 16. We point out that with $4 \times 4 \otimes 4 \times 4$ operators, the alternating minimization solver with 16 dyads tracks the convex solver quite well. If we consider the $\ell_1 \otimes \ell_2$ nuclear norm instead (Figure 5.10), we do not see this oddity either.

In the last section, we saw that matching the nuclear norm to the factor structure of the true operator improves denoising performance. The results there only considered denoising rank-1 operators, and so we wish to ensure that similar

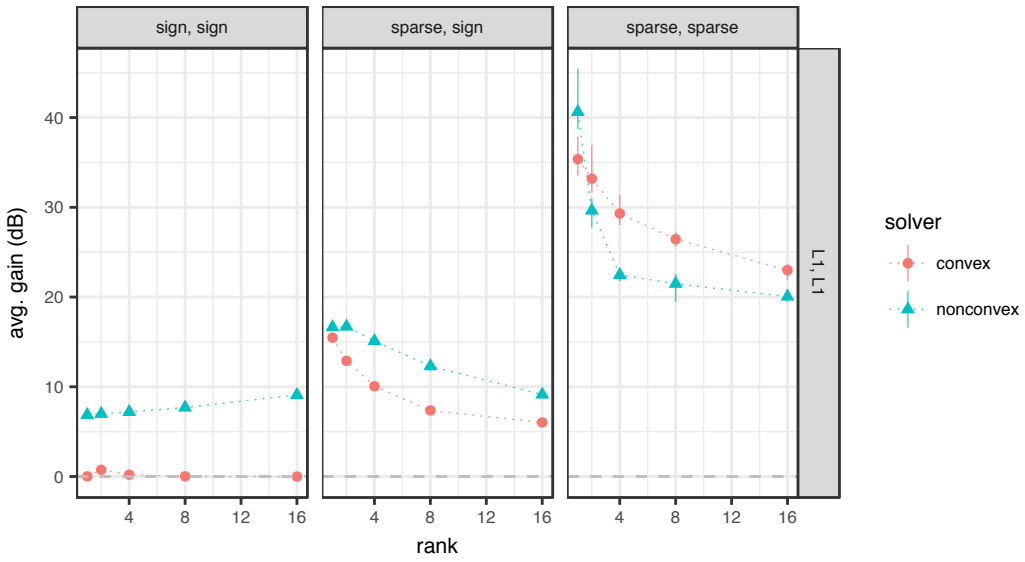


Figure 5.9: **Average gain vs. rank, ℓ_1 norm.** Each panel plots the average gain (5.9) of the denoising procedure over 10 trials versus the operator rank using the ℓ_1 norm as a regularizer with the color/shape indicating the solver. We test the convex solver `matsolve` and the nonconvex alternating minimization solver `altminsolve` (with solver rank 16). Alternating minimization uses solver rank 16. All operators have dimension $16 \times 16 \otimes 16 \times 16$, and the SNR (5.7) is 10dB. Error bars show the minimum and maximum gain over the trials. We facet the figure by factor structure.

behavior holds for higher-rank operators. Tables B.4 and B.5 in the appendix confirm this for $4 \times 4 \otimes 4 \times 4$ and $16 \times 16 \otimes 16 \times 16$ operators, respectively.

In fact, we see that the discrepancies between factor structure and nuclear norm that we noticed in Table 5.3[bottom] diminish as the operator rank increases. We conclude that even though the gains decrease with growing operator rank, nuclear norms still provide superior denoising performance when tuned to the factor structure of the operator.

5.5.3 Semidefinite relaxations

Finally, we want to consider the performance of the semidefinite relaxations for nuclear norms. In Section 3.6 we introduced these relaxations and stated that they are comparable to the true nuclear norms up to a dimension-independent multiplicative constant.

Figure 5.11 shows two experiments comparing the semidefinite relaxations to their true counterparts. In the top panel we show gain (5.9) plotted against

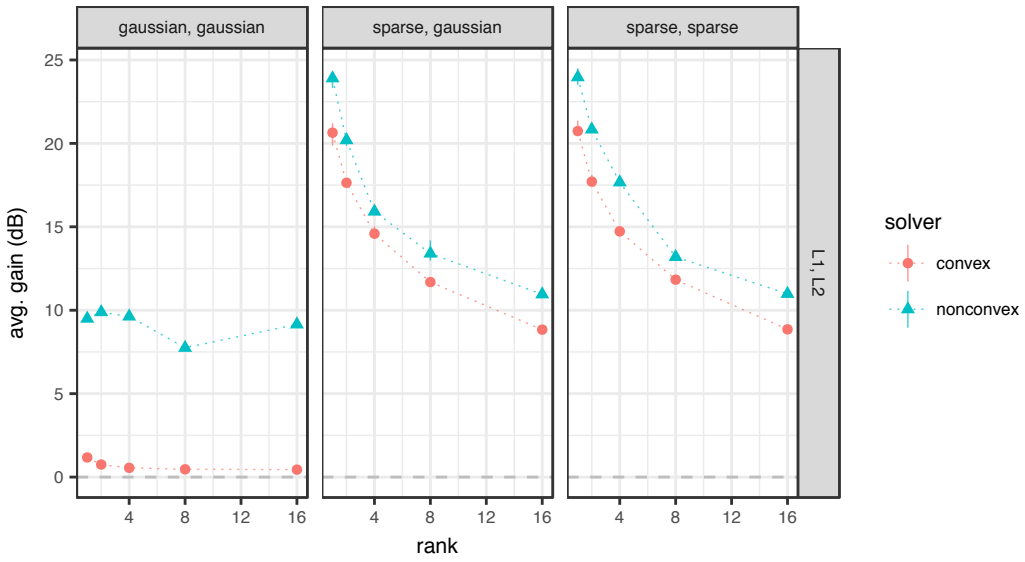


Figure 5.10: **Average gain vs. rank, $\ell_1 \otimes \ell_2$ norm.** Each panel plots the average gain (5.9) of the denoising procedure over 10 trials versus the operator rank using the $\ell_1 \otimes \ell_2$ nuclear norm as a regularizer with the color/shape indicating the solver. We test the convex solver `matsolve` and the nonconvex alternating minimization solver `altminsolve` (with solver rank 16). All operators have dimension $16 \times 16 \otimes 16 \times 16$, and the SNR (5.7) is 10dB. Error bars show the minimum and maximum gain over the trials. We facet the figure by factor structure.

operator rank for the smaller set of operators ($4 \times 4 \otimes 4 \times 4$) and with the alternating minimization solver set to use 16 dyads. In the bottom panel we create the same plot for our larger ($16 \times 16 \otimes 16 \times 16$) operators, also with solver rank 16. We facet by the nuclear norms and factor structures, and for comparison, we plot the gain for the trace norm as a dotted gray line.

In the top panel, we see that the true nuclear norm (with 16 dyads) generally outperforms the relaxed norm. While the difference is generally small, it is most noticeable at low operator ranks. The largest discrepancy appears when using the $\ell_\infty \otimes \ell_\infty$ nuclear norm to denoise (sign, sign) operators.

The situation differs somewhat when we consider the rank-constrained solutions in the bottom panel. Here, with the exception of the $\ell_\infty \otimes \ell_\infty$ nuclear norm, the formulations are generally close. While the true formulation tends to provide slightly higher gain than the relaxed one, this is not always the case.

Now, however, we see that the relaxed $\ell_\infty \otimes \ell_\infty$ nuclear norm (the max-norm) performs as well as the trace norm across the three factor structures tested.

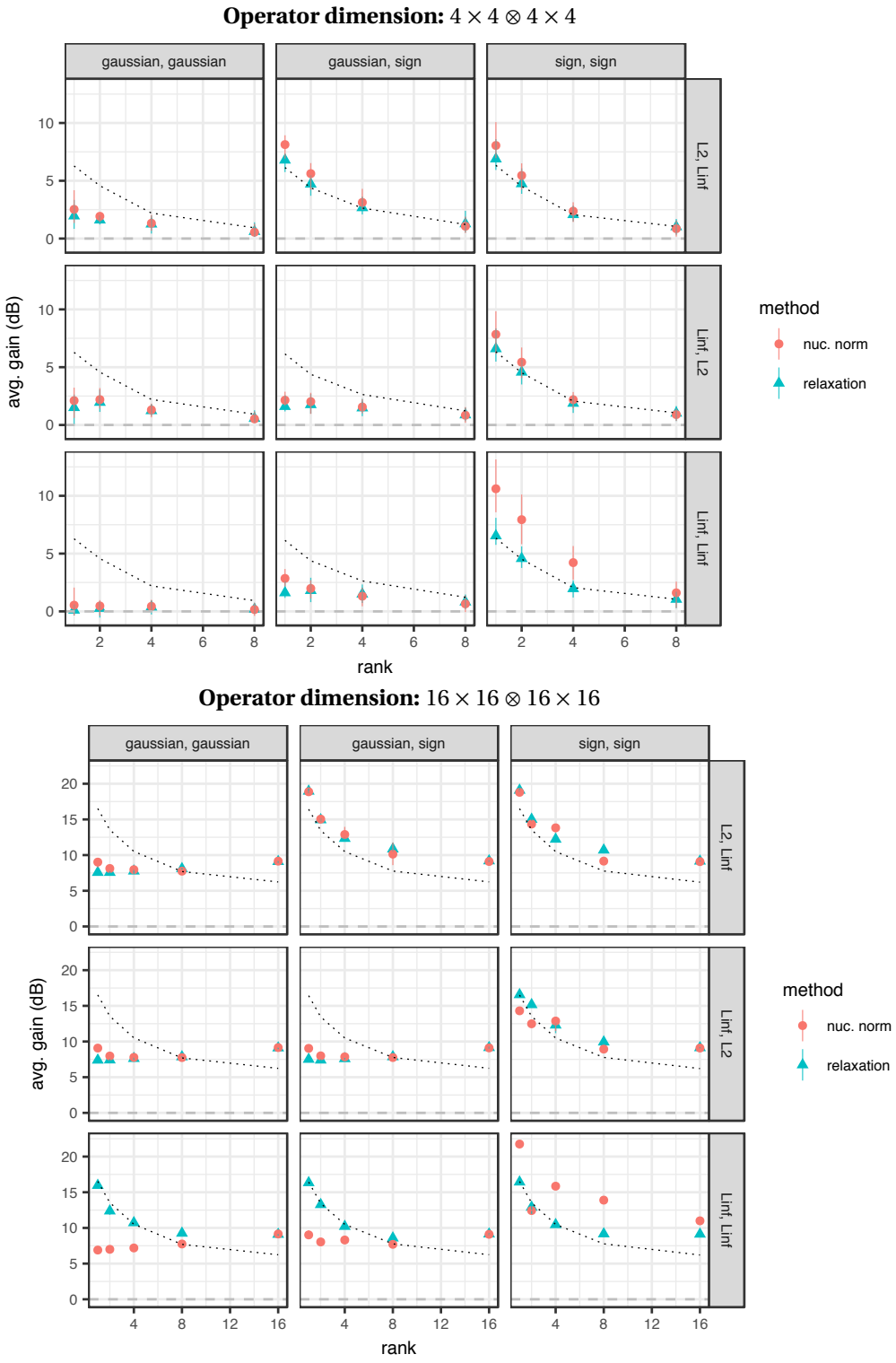


Figure 5.11: **Average gain vs. rank, semidefinite relaxations.** Each panel plots the avg. gain (5.9) of the denoising procedure over 10 trials vs. the operator rank with color/shape indicating the method. We test nuclear norms and their semidefinite relaxations. The dotted line shows the avg. gain of the $\ell_2 \otimes \ell_2$ nuclear norm for reference. All tests use the alternating minimization solver (with solver rank 16) and SNR (5.7) of 10dB. We include min/max error bars, and we facet by factor structure (columns) and nuclear norm (rows).

The true $\ell_\infty \otimes \ell_\infty$ nuclear norm norm only does so in the case of (sign, sign) operators—where we expect it to perform well anyway. This is in contrast to the unconstrained case in the top panel where both formulations of the $\ell_\infty \otimes \ell_\infty$ nuclear norm have little success in denoising factor structures other than (sign, sign).

5.5.4 Demixing

Finally, we wish to briefly address the question of demixing. The denoising problem itself simply asks us to approximate an operator from complete but noisy observations. Using our knowledge that the operators have a structured factorization, we employ nuclear norms to facilitate this approximation.

We may wonder, however, whether or not we can recover that factorization. This is itself a rather different problem, and it is an important one even in the case where we observe the true signal directly. Given that the alternating minimization solver already works in a factored form, we question whether it can indeed return structured factorizations.

To test this we consider a simple experiment. We generate the operator

$$\mathcal{A}^\natural = \sum_{i=1}^{4} \boldsymbol{X}_i \otimes \boldsymbol{Y}_i,$$

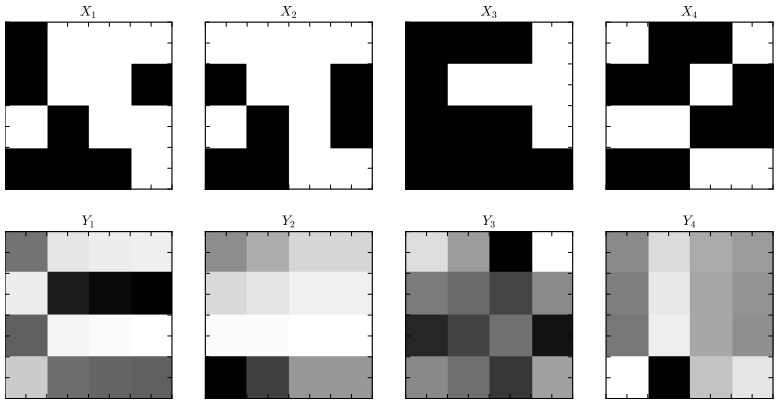
where the $\boldsymbol{X}_i \in \mathbb{M}^{4\times 4}$ are independent random sign matrices, and the $\boldsymbol{Y}_i \in \mathbb{M}^{4\times 4}$ are independent random rank-1 matrices.

We then add noise to achieve an SNR of 10dB and solve the denoising problem using the $\ell_\infty \otimes S_1$ nuclear norm in the alternating minimization solver. We set the solver rank to 16 and the convergence tolerance to $\epsilon = 1 \times 10^{-3}$. At convergence, the solver returns an operator with error resulting in an RSDR (5.8) of 14.0dB (a gain of 4dB).

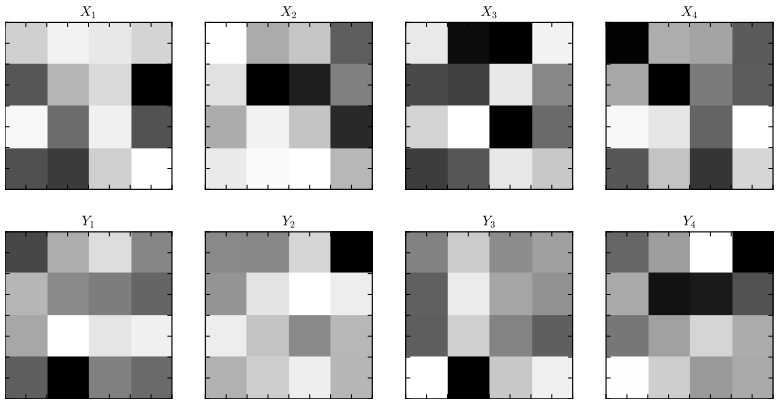
In Figure 5.12 we display the left and right factors of $\mathcal{A}^\natural$ along with the first four dyads of the recovered operator. The first set of panels shows the true factors with the left (sign) factors displayed above the right (rank-1) factors. The bottom set of panels shows the first 4 recovered factors using alternating minimization. Note that the recovered factors match the true factors extremely closely up to ordering and sign!

We also include the first four components of the truncated dyadic SVD as the middle set of panels. The right components serve as the initialization for the

Left and right factors of the true operator $\mathcal{A}^\natural$



Truncated dyadic SVD of the noisy observations



Alternating minimization with the $\ell_\infty \otimes S_1$ norm, first 4 components

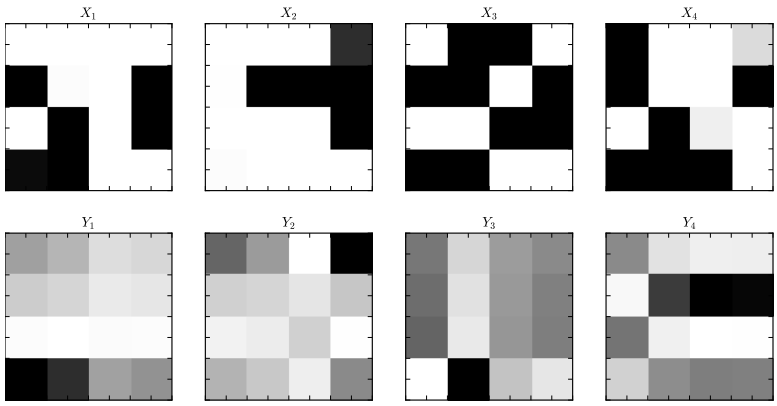


Figure 5.12: **Demixing with alternating minimization.** Each set of figures displays the left and right factors of a particular dyadic decomposition. The top row shows the dyads of a rank-4 (sign, low-rank) operator in $\mathbb{O}^{4\times 4\otimes 4\times 4}$. The middle row shows the top 4 components of the dyadic SVD applied to the noisy observations (SNR 10dB), and the bottom row shows the first 4 dyads returned by the alternating minimization solver using the $\ell_\infty \otimes S_1$ nuclear norm for denoising (gain: 4.1dB).

alternating minimization solver, and it is evident that their orientation has an effect on the ordering of the recovered components.

In any case, we present this example to suggest that solving nuclear norm problems with alternating minimization can also achieve structured factorizations. This is obviously an idealized example, and our work is generally focused on using nuclear norms to solve approximation problems. We leave this aspect for future study.

5.6 Summary

In this chapter, we performed numerical experiments to demonstrate the effectiveness of nuclear norms in denoising structured low-rank operators. At the same time we also had to demonstrate that our alternating minimization solver and hyperparameter choices produce accurate results. From these tests, we conclude that nuclear norms—used as regularizers—do indeed promote structured low-rank operators. Furthermore, we can tailor nuclear norms to match the underlying structures of those operators. In the next chapter we consider an application of nuclear norm denoising to hyperspectral imaging.

Chapter 6

Application: Hyperspectral image denoising

In the previous chapter we evaluated the numerical performance of nuclear norms in denoising synthetic operators. Here we apply the nuclear norm framework to denoising hyperspectral images making use of some real data.

6.1 Overview

Whereas a grayscale image records the total intensity of light at each pixel, a hyperspectral image (HSI) records the intensity of light at any number of specific wavelengths.[1] The RGB image we displayed in Figure 1.1, for instance, stored a color image in red, green, and blue components. More generally, we can think of hyperspectral images as a data cube with an arbitrary number of two-dimensional slices, one for each wavelength of light recorded, as shown in Figure 6.1.

In this way, hyperspectral images are three-dimensional arrays. For an ordered set of wavelengths, we can then write an HSI as an operator

$$\mathcal{A} = \sum_k \boldsymbol{X}_k \otimes \boldsymbol{e}_k,$$

where $\boldsymbol{X}_k$ is the matrix of intensities at wavelength k and the $\boldsymbol{e}_k$ are standard basis vectors.

[1] Note that the literature contains references to both hyperspectral and multispectral images with the distinction generally being the continuity of the measured spectral bands. This distinction does not concern us, and we will refer to all images with spectral data as hyperspectral.

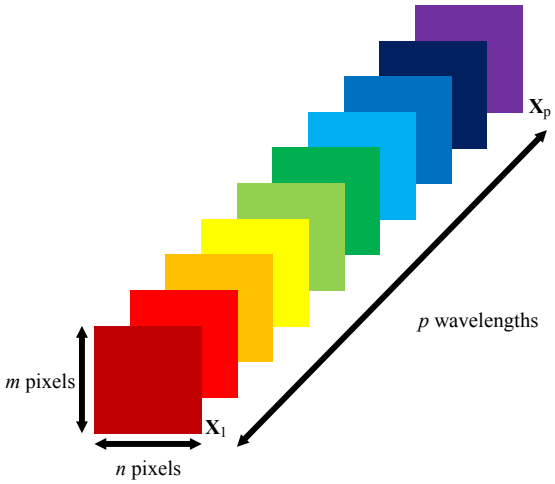


Figure 6.1: **HSI datcube.** We can represent a hyperspectral image (HSI) as an $m \times n \times p$ data cube, where each of the p slices is an $m \times n$ image recording the intensity of light at a specific wavelength. From the set of these p slices $\{\boldsymbol{X}_k\}_{k=1}^{p}$, we can construct the HSI as the operator $\mathcal{A} = \sum_{k=1}^{p} \boldsymbol{X}_k \otimes \boldsymbol{e}_k$.

The focus of this chapter is to apply nuclear norm denoising techniques from the previous chapter to hyperspectral images incorporating real data. We also provide a brief comparison with a recent approach by Zhao and Yang [ZY15] that applies techniques from dictionary learning to this problem. Note, however, that we intend this chapter to explore the use of nuclear norms in a setting that moves away from wholly synthetic data. Even though our approach does show some benefit, we must also discuss some notable current obstacles to large-scale implementation.

6.1.1 Roadmap

In Section 6.2 we discuss relevant work from the literature. We then test nuclear norm denoising on hyperspectral images generated using real spectral data in Section 6.3. Section 6.4 addresses pitfalls with the approach along with opportunity for future work.

6.2 Relevant work

Hyperspectral imaging has a decades-long history in the field of remote geological sensing. In the 1970s, Rowan and coauthors [Row+74; RGA77] used satellite imaging data to perform mineral identification. Gregg Vane and coauthors at the Jet Propulsion Laboratory pioneered airborne instruments for imaging

spectrometry in the 1980s [VG88; VG93]. More recent applications appear in diverse areas such as quality control in food processing [KCM01; Gow+07], astronomical surveys [Heg+03], counterfeit drug detection [Rod+05], and medical diagnosis [AKKT10].

In this section, we focus on a mixture model that provides a basis for exploration of HSI with nuclear norms.

6.2.1 A mixture model for HSI

Consider remote geological sensing where, for instance, airplanes fly over the ground measuring the reflected spectra of materials below. The resulting hyperspectral images will have a fixed spatial resolution, and the spectrum at a single pixel very likely results as a combination of several different materials on the ground.

The linear mixing model [KM02] assumes that the surface being imaged consists primarily of a small number of materials each having a roughly constant spectrum over the surface. Each of these spectra are called *endmembers*, and each pixel arises as a convex combination of the spectra. The weights in the convex combination are called the *abundances*. Mathematically, we can write the spectrum of a pixel $\boldsymbol{x} \in \mathbb{R}^L$ as

$$\boldsymbol{x} = \boldsymbol{S}\boldsymbol{a},$$

where $\boldsymbol{S}$ is an $L \times M$ matrix whose columns are the endmembers, and $\boldsymbol{a} \in \mathbb{R}^M$ gives the abundances. If the pixel spectrum $\boldsymbol{x}$ results from the combination of few materials, then the abundances $\boldsymbol{a}$ will be sparse.

To construct a hyperspectral image with N pixels, we then have the model

$$\boldsymbol{X} = \boldsymbol{S}\boldsymbol{A}, \tag{6.1}$$

where the columns of $\boldsymbol{X} \in \mathbb{M}^{L\times N}$ are the pixel spectra, and the columns of $\boldsymbol{A} \in \mathbb{M}^{M\times N}$ are the abundances of each pixel. Note that this resembles a sparse coding model.

The goal then is to receive a hyperspectral image and "unmix" its pixels to determine their constituent materials. The survey of Bioucas–Dias et al. [Bio+12] reviews techniques for solving this problem including approaches based on independent component analysis [BGC98; CZ99; Tu00], sparse regression [RRZF06;

IPB10; IBP11], and dictionary learning [COR11]. Key distinctions among these approaches include whether or not the spectral endmembers $\boldsymbol{S}$ are known.

6.2.2 Denoising vs. spectral unmixing

Consider again the linear mixture model (6.1). This is exactly a matrix factorization model, and the problems of denoising and spectral unmixing under this model are then the two central questions that opened this thesis.

Denoising requires finding an *approximation* of the hyperspectral image $\boldsymbol{X}$ given the knowledge that it has the factorization $\boldsymbol{X} = \boldsymbol{SA}$. Spectral unmixing, on the other hand, requires finding that factorization. While we focus on using nuclear norms to perform denoising under this model, we will also consider the unmixing problem.

We do note, however, that other approaches for denoising hyperspectral images exist including wavelet-based methods [ZG06], tensor filtering [LB08; RBB08], tensor decomposition [LBF12; LB13; Li+15], and low-rank matrix methods [Zha+14; HZZS16]. Again, our focus is the continued numerical testing of nuclear norms and not a broad comparison of HSI methods. To that end, we make a limited comparison between nuclear norms, the truncated dyadic SVD (3.7), and a dictionary learning-based method called Spa+Lr [ZY15].

6.2.3 Spa+Lr

The Spa+Lr method of Zhao and Yang [ZY15] combines a dictionary learning approach [EA06] with a low-rank assumption on the (matricized) hyperspectral image based on the linear mixture model. We use this method as a point of comparison for nuclear norm denoising to provide our results with some context from the HSI literature.

We make this choice for two reasons. First, they test hyperspectral images generated using a linear mixture model. This provides a good starting point for comparison with nuclear norms. Second, the unsupervised dictionary learning approach fits well into our discussion of bilinear problems as opposed to semisupervised techniques using a curated spectral library [IBP11]. We now summarize their approach.

First, they consider the hyperspectral image $\boldsymbol{X}^\natural$ as an $MN \times L$ matrix, where M, N are the spatial dimensions, and L is the spectral dimension. Following the

linear mixture model, they assume $\boldsymbol{X}^\natural$ has a representation

$$\boldsymbol{X}^\natural = \boldsymbol{A}\boldsymbol{S},$$

where the rows of $\boldsymbol{S} \in \mathbb{M}^{P\times L}$ are the endmembers, and the rows of $\boldsymbol{A} \in \mathbb{M}^{MN\times P}$ are the abundances for each of the MN pixels. (Note this model is the transpose of (6.1).)

They observe the noisy HSI

$$\boldsymbol{Y} = \boldsymbol{X}^\natural + \boldsymbol{W},$$

where $\boldsymbol{W}$ is Gaussian noise.

To approximate the original HSI $\boldsymbol{X}^\natural$, they consider the nonconvex problem

$$\underset{\boldsymbol{X},\boldsymbol{D},\boldsymbol{\alpha_i}}{\text{minimize}} \quad \gamma\|\boldsymbol{X}-\boldsymbol{Y}\|_{\ell_2}^2 + \sum_i \|\boldsymbol{R}_i\boldsymbol{X}-\boldsymbol{D}\boldsymbol{\alpha}_i\|_{\ell_2}^2 + \sum_i \eta\|\boldsymbol{a}_i\|_0 + \mu\,\text{rank}(\boldsymbol{X}).$$

The first term measures the fidelity between the decision variable $\boldsymbol{X}$ and the observed noisy HSI $\boldsymbol{Y}$. The next term concerns finding a dictionary $\boldsymbol{D}$ and coefficients $\boldsymbol{\alpha}_i$ such that each patch of the HSI $\boldsymbol{R}_i\boldsymbol{X}$ may be coded in terms of the dictionary, where the $\boldsymbol{R}_i$ are the operators that extract overlapping, rectangular patches of the HSI. The following term promotes sparse coefficient vectors $\boldsymbol{\alpha}_i$ so that we obtain a sparse coding for the patches. The final term penalizes HSI with high rank.

This formulation is sparse dictionary learning with an additional low-rank constraint. Due to the linear mixing assumption, they expect that $\boldsymbol{X}^\natural$ is indeed low-rank (or approximately so).

Before attempting to solve this problem they replace the hard rank constraint with the trace norm, introduce an auxiliary variable $\boldsymbol{U} = \boldsymbol{X}$, and write the auxiliary constraint as a penalty term:

$$\underset{\boldsymbol{X},\boldsymbol{D},\boldsymbol{\alpha}_i,\boldsymbol{U}}{\text{minimize}} \quad \gamma\|\boldsymbol{X}-\boldsymbol{Y}\|_{\ell_2}^2 + \sum_i \|\boldsymbol{R}_i\boldsymbol{X}-\boldsymbol{D}\boldsymbol{\alpha}_i\|_{\ell_2}^2 + \sum_i \eta\|\boldsymbol{a}_i\|_0 + \mu\|\boldsymbol{U}\|_{S_1} + \lambda\|\boldsymbol{X}-\boldsymbol{U}\|_{\ell_2}^2.$$

This problem, while still nonconvex, may be solved through alternating minimization.

The major steps are:

1. Fix $\boldsymbol{U}$, $\boldsymbol{X}$: Solve the dictionary learning/sparse coding problem in $\boldsymbol{D}$ and $\boldsymbol{\alpha}_i$ using K-SVD [AEB06].

2. Fix $\boldsymbol{X}$, $\boldsymbol{D}$, $\boldsymbol{\alpha}_i$: Solve for $\boldsymbol{U}$ by soft-thresholding the singular values of $\boldsymbol{X}$.

3. Fix $\boldsymbol{U}$, $\boldsymbol{D}$, $\boldsymbol{\alpha}_i$: Solve for $\boldsymbol{X}$ by performing the averaging step

$$\boldsymbol{X} = \left(\gamma \mathbf{I} + \sum_i \boldsymbol{R}_i^{\mathsf{t}} \boldsymbol{R}_i + \lambda \mathbf{I}\right)^{-1} \cdot \left(\gamma \boldsymbol{Y} + \sum_i \boldsymbol{R}_i^{\mathsf{t}} \boldsymbol{D} \boldsymbol{\alpha}_i + \lambda \boldsymbol{U}\right).$$

We implement Spa+Lr in Python using *scikit-learn* [Ped+11] for the dictionary learning routines. We discuss additional details in Section C.3 of the appendix.

6.3 Structured abundances

Now we turn to implementing HSI denoising as a nuclear norm problem and testing its performance with real spectra.

6.3.1 An operator mixture model

Of course we could write the linear mixture model (6.1) in our usual dyadic notation as

$$\boldsymbol{A} = \sum_i \boldsymbol{x}_i \otimes \boldsymbol{y}_i, \tag{6.2}$$

where the $\boldsymbol{y}_i \in \mathbb{R}^n$ are the endmembers (with spectral dimension n), and the $\boldsymbol{x}_i \in \mathbb{R}^m$ give the abundances of that endmember at each of the m pixels in the HSI. The result is a matrix $\boldsymbol{A} \in \mathbb{M}^{m \times n}$ with each row representing a pixel (spatial location), and each column a particular wavelength.

Notice though that the abundances in dyadic form need not be sparse. Indeed, if every pixel of an image is composed in part by endmember $\boldsymbol{y}_i$, then no entry of $\boldsymbol{x}_i$ will be zero. This is as we mentioned in Section 2.1.1 when discussing dictionary learning. The rows of the abundance matrix $\boldsymbol{X}$ whose columns are the $\boldsymbol{x}_i$ may be sparse, but the nuclear norm will not directly promote that structure.

There we discuss that lifting the dictionary learning problem to the operator space is a possibility, but our preliminary investigations revealed difficulties with this approach. Namely, the observed linear measurements in the lifted setting are insufficient to solve the dictionary learning problem using operator nuclear norms. In the HSI setting, however, we may use a slightly different approach.

Instead we consider the HSI $\mathcal{A} \in \mathbb{O}^{m\times n\otimes p}$ with spatial dimensions $m \times n$ and spectral dimensions p, and we write it as

$$\mathcal{A} = \sum_i \boldsymbol{X}_i \otimes \boldsymbol{y}_i,$$

where the $\boldsymbol{y}_i \in \mathbb{R}^p$ are the endmembers and the $\boldsymbol{X}_i \in \mathbb{M}^{m\times n}$ are their corresponding abundances at each pixel.

This small change allows us to take advantage of the two-dimensional structure of the abundances. For instance, if $\mathcal{A}$ is an HSI obtained through remote sensing, the abundances of materials may occur in contiguous patches, and we could then expect that the $\boldsymbol{X}_i$ are low-rank. Choosing a $S_1 \otimes Y$ nuclear norm could then promote this structure.

Alternatively, we may consider that the pattern of abundances themselves should resemble an image with few, sharp jumps in intensity. Total variation[2] may then be an appropriate regularizer on the $\boldsymbol{X}_i$. (And while we do recognize that we could apply one-dimensional TV to the matricized HSI (6.2), we prefer to retain the two-dimensional structure of the HSI throughout.)

6.3.2 Test images

To test nuclear norm denoising, we generate hyperspectral images using real spectra from the USGS Digital Spectral Library [Cla+07]. This database, referred to as splib06a, contains over 1300 spectra including minerals, organic compounds, vegetation, and man-made material. We follow the same generation procedure as Zhao and Yang [ZY15] that originates from Iordache et al. [IBP11].

First, we select 5 spectra from the database with some care taken to choose sufficiently different materials; the database includes many sets of extremely similar spectra. We perform some additional cleaning and smoothing of the spectra and provide those details in Section C.1 of the appendix. These steps undoubtedly assist the nuclear norm denoising procedure, but we still believe that this satisfies our desire to incorporate real data into our experiments (as opposed to the entirely synthetic tests of last chapter). Our experiments still use Spa+Lr as a point of comparison under the same circumstances.

[2] The two-dimensional total variation (2D TV) of a matrix $\boldsymbol{X} \in \mathbb{M}^{m\times n}$ is defined as $\|\boldsymbol{X}\|_{\mathrm{TV}} := \sum_{i,j} \sqrt{|x_{i+1,j} - x_{i,j}| + |x_{i,j+1} - x_{i,j}|}$. The use of total variation in image denoising was pioneered by Rudin et al. [ROF92]

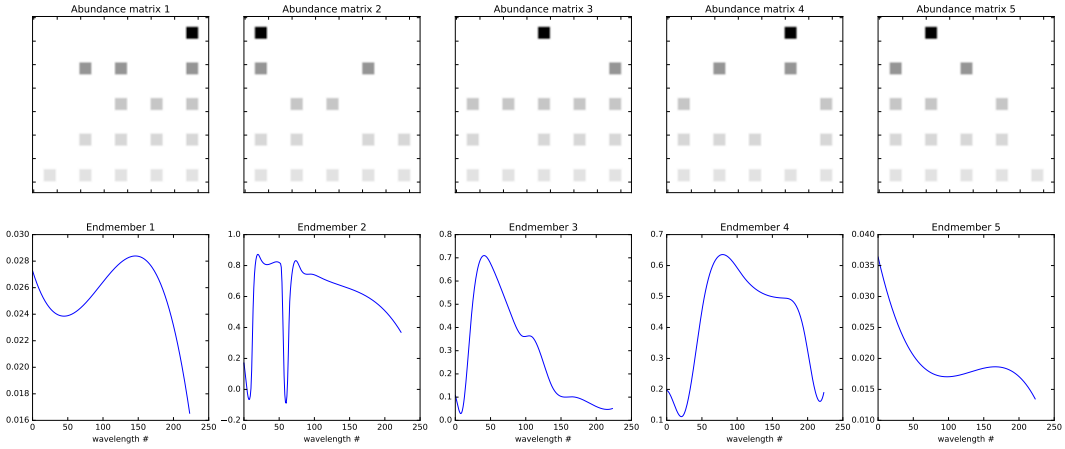


Figure 6.2: **HSI test image.** The abundances and endmembers of the test image $\mathcal{A} = \sum_{i=1}^{5} \boldsymbol{X}_i \otimes \boldsymbol{y}_i \in \mathbb{O}^{75\times75\otimes224\times1}$ generated using spectra from the USGS splib06a library according to the procedure of Iordache et al. [IBP11]. The top panels show the abundances $\boldsymbol{X}_i$, while the bottom panels plot the endmembers $\boldsymbol{y}_i$.

Using the selected spectra we generate a $75 \times 75 \times 224$ HSI such that there are 25 patches each of size 5×5 equally spaced in a square grid. The patches in the first row are composed of exactly one of the materials. In the second row, each patch is the equal combination of two materials, and so forth until we reach the last row where all patches are equal combinations of all five materials. We ensure that every material is represented in at least one patch of every row.

Figure 6.2 shows such an HSI $\mathcal{A} = \sum_{i=1}^{5} \boldsymbol{X}_i \otimes \boldsymbol{y}_i$, where the panels of the top row show each of the abundance matrices $\boldsymbol{X}_i$, and the bottom panels show the endmembers $\boldsymbol{y}_i$. Note that in the first row of the abundances, each patch only appears in one of the $\boldsymbol{X}_i$. In the second row, each patch appears in two of the $\boldsymbol{X}_i$, and so forth.

6.3.3 Numerical results

Let $\mathcal{A}^\natural \in \mathbb{O}^{75\times75\otimes224\times1} = \sum_{i=1}^{5} \boldsymbol{X}_i \otimes \boldsymbol{y}_i$ be the testing HSI generated in the previous section, and assume that we observe the noisy HSI

$$\mathcal{B} = \mathcal{A}^\natural + \sigma\mathcal{Z},$$

where the additive noise $\mathcal{Z}$ has independent standard normal entries.

We now test the ability of nuclear norms to denoise this HSI. For the sake of comparison we also consider using the truncated dyadic SVD (3.7) to compute the

Dyads	$\ell_1 \otimes \ell_2$	$\ell_1 \otimes$ TV	$S_1 \otimes \ell_2$	$S_1 \otimes$ TV	TV $\otimes \ell_2$	TV $\otimes$ TV	Dyadic SVD	Spa+Lr
5	21.4	17.7	23.0	18.6	**25.0**	21.0	15.7	
10	19.8	16.4	22.2	17.0	**23.8**	20.7	12.4	
N/A								**16.5**

Table 6.1: **Denoising the HSI test image, 10 dB SNR.** This table lists the average gains in dB (5.9) over 10 trials of the denoising experiment on the HSI test image. The bold figures indicate the largest value in each row.

best low-rank approximation to the noisy observation $\mathcal{B}$. By construction, the true HSI $\mathcal{A}^\natural$ has rank 5. We also test the Spa+Lr method of Zhao and Yang [ZY15] described in Section 6.2.3. The full experimental details are included in Section C.4 of the appendix.

Recall that abundance matrices $\boldsymbol{X}_i$ in the construction of the test HSI $\mathcal{A}^\natural$ have well-defined structure. In particular, they are low-rank and sparse. When viewed as images, they have continuous patches with well-defined edges. And so we believe that nuclear norms involving the ℓ_1 norm, trace norm, and total variation norm will serve well for the left factors.

For the endmembers (spectra), we consider the ℓ_2 and total variation norms. We expect that the ℓ_2 norm will perform better on the smoothed endmembers in this example, but we include the TV norm for comparison nonetheless.

Table 6.1 shows the results of the numerical experiment described above and in Section C.4 of the appendix. We show the average gain[3] in decibels for each combination of factor norms and with both 5 and 10 dyads allowed in the solver. The right-hand side of the table displays the gain using the truncated dyadic SVD and Spa+Lr to perform the denoising.

We see that using the TV $\otimes \ell_2$ nuclear norm yields the best performance at both 5 and 10 dyads. In both cases, the gain also outperforms the truncated dyadic SVD by a good margin. This is encouraging as the truncated SVD corresponds to solving a rank-constrained denoising problem with no regularization (Section 5.4.4.4).

As expected, using the ℓ_2 norm on the right factors (endmembers) does result in better performance than using total variation in this case. Using total variation, however, still allows the nuclear norm methods to outperform the baseline truncated SVD. Note that the ℓ_1 and S_1 norms still perform admirably in regularizing

[3] Recall that we defined gain (in decibels) as the recovery signal-to-distortion ratio (RSDR) less the signal-to-noise ratio (SNR) of the observations. See Section 5.4.2 for the definitions.

the abundances, but we expect that the ℓ_1 norm in particular benefits from the very special structure of our test image.

Finally, we observe that the nuclear norm approach here outperforms Spa+Lr. The use of Spa+Lr allows us to conclude that the gains from our method compare favorably to techniques in the HSI literature. We do not, however, make the claim here that nuclear norm denoising is strictly better than using Spa+Lr. Indeed, the nature of this test image suits itself well to our framework. We revisit this point later.

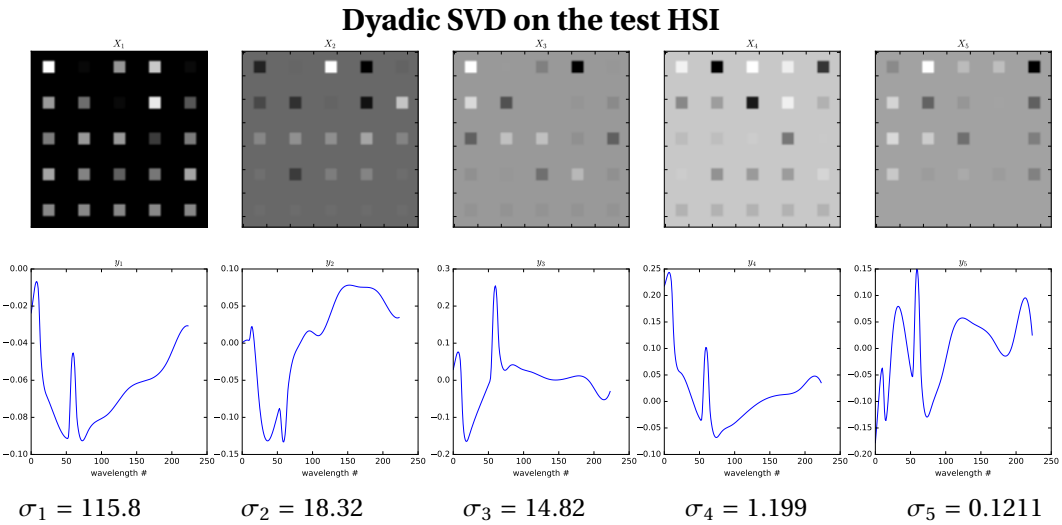
6.3.4 Unmixing

This chapter focuses on the denoising problem. That is, we see noisy observations of a hyperspectral image and we wish to approximate that image under the assumption that it has a low-rank representation as a linear mixture of materials. We do, however, want to briefly address the problem of spectral unmixing. In our framework this corresponds to actually obtaining a factorization of the HSI (as an operator) that corresponds to the linear mixture model. We previously saw a demixing example in Section 5.5.4.

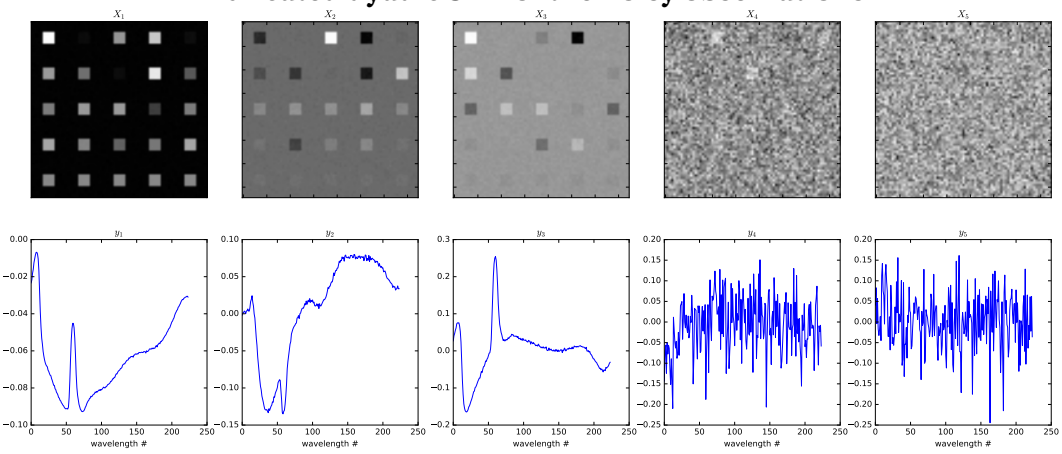
Given that we use alternating minimization to solve the nuclear norm denoising problem, we may wonder how well the recovered factors unmix the data. Before even looking at the results, however, we offer a strong word of caution. Figure 6.2 displays a factorization of our test HSI into abundances and endmembers. While we use nuclear norms to promote the different properties of the factors, we have no reason to believe here that this factorization is an optimal nuclear norm decomposition of the noiseless HSI $\mathcal{A}^\natural$. This alone should suggest the difficulty in using our approach to perform factorization in addition to approximation.

Nevertheless, we show the output of our solver in Figure 6.3. The top set of panels shows the dyadic SVD of the rank-5 test HSI along with their singular values. Of course, this decomposition does not match that of Figure 6.2. The middle set of panels shows truncated dyadic SVD applied to the noisy observations. We see that the components corresponding to the smaller singular values are more susceptible to noise. Finally, the last set of panels shows the output of the alternating minimization solver applied to the noisy observations using the $\mathrm{TV} \otimes \ell_2$ nuclear norm.

We note here that they strongly resemble the decomposition obtained by the



$\sigma_1 = 115.8$ $\sigma_2 = 18.32$ $\sigma_3 = 14.82$ $\sigma_4 = 1.199$ $\sigma_5 = 0.1211$



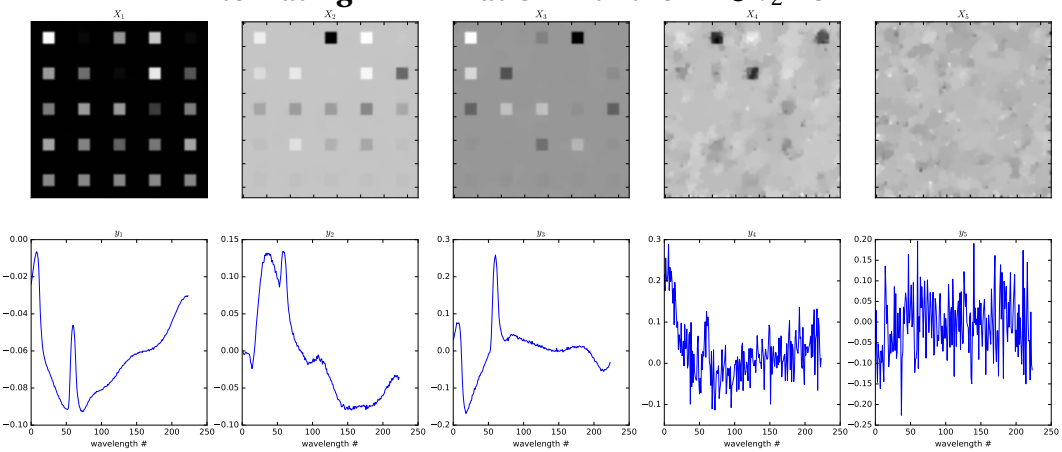


Figure 6.3: **Unmixing in HSI denoising.** Each set of figures displays the left and right factors of a particular dyadic decomposition. The top row shows the top 5 components of the dyadic SVD of the test HSI. The middle row shows the top 5 components of the dyadic SVD applied to the noisy observations (SNR 10dB), and the bottom row shows the 5 dyads returned by the alternating minimization solver using the TV $\otimes$ ℓ_2 nuclear norm for denoising.

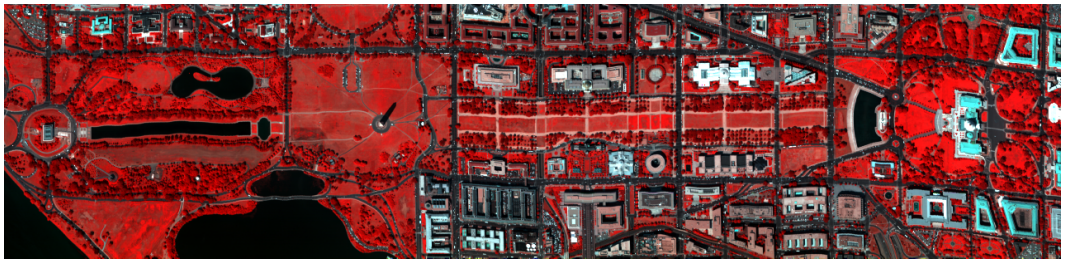


Figure 6.4: **Washington, D.C. Mall HYDICE image.** A false color reproduction of a $1280 \times 307 \times 191$ aerial hyperspectral image.

dyadic SVD. Note that, in particular, the fourth component shows a much better resemblance to the true HSI than the truncated dyadic SVD alone. This is the value of using the nuclear norm in the denoising procedure. After all, the default initialization for our alternating minimization solver *is* the truncated dyadic SVD of the noisy observations (Section 4.2.3). The nuclear norm allows us to improve on that and return more of the structure present in the original HSI. We do not, however, achieve any meaningful spectral unmixing.

6.4 Next steps

We would like to also apply our techniques to a real hyperspectral image. Figure 6.4 shows a false color version of a Hyperspectral Digital Imagery Collection Experiment (HYDICE) [Bas+93] image of the Washington DC Mall.[4] The HSI results from aerial imaging and has spatial dimension 1280×307 and spectral dimension 191.

To make the size more manageable, we conducted preliminary tests using a 128×128 patch centered on the Lincoln Memorial (on the left side of Figure 6.4). Quick testing with the truncated dyadic SVD showed that the resulting data cube is approximately low-rank with 5 dyads capturing over 99.9% of its variance. Denoising with the dyadic SVD at 10dB SNR consequently showed gains of 15.3dB with 5 dyads and 12.2 dB with 10 dyads. These results roughly match those of our HSI with synthetic abundances in Table 6.1. Further testing with Spa+Lr demonstrated a gain of 11.7dB, and this result agrees with the tests of Zhao and Yang [ZY15] on the same image (albeit a different patch).

Testing with the nuclear norm framework, however, did not yield gains signifi-

[4] Available at https://engineering.purdue.edu/~biehl/MultiSpec/hyperspectral.html under free license for testing and/or research.

cantly higher than the dyadic SVD when using 5 or 10 dyads in the alternating minimization solver. This suggests that the bulk of the denoising resulted from the reduction in solver dyads and the corresponding initialization using the dyadic SVD. We hypothesize that even though a linear mixing model may be appropriate to describe this image, our available nuclear norms are not able to promote this structure strongly enough here.

A possible solution is to follow work in spectral unmixing, such as [IBP11], where it is common to use a spectral library (such as splib06a) to specify the endmembers *a priori*. Solving such a problem, however, no longer requires the nuclear norm framework. A disadvantage to this approach is the need for a spectral library that is well-calibrated to a particular imaging setup.

Alternatively, we could consider an HSI model like

$$\mathcal{A} = \sum_i \boldsymbol{X}_i \otimes (\boldsymbol{D}\boldsymbol{y}_i),$$

where the $\boldsymbol{X}_i$ are again abundances, and the spectra are now $\boldsymbol{D}\boldsymbol{y}_i$ composed from a spectral dictionary $\boldsymbol{D}$. Our desire would be for the $\boldsymbol{y}_i$ to be sparse. We could instead consider the operator

$$\mathcal{A}^\natural = \sum_i \boldsymbol{X}_i \otimes \boldsymbol{y}_i,$$

where we have simply removed the spectral dictionary $\boldsymbol{D}$, and the measurement model

$$\boldsymbol{\mu}(\mathcal{A}) = \boldsymbol{\mu}\left(\sum_i \boldsymbol{X}_i \otimes \boldsymbol{y}_i\right) = \sum_i \boldsymbol{X}_i \otimes \boldsymbol{D}\boldsymbol{y}_i.$$

That is, we move the spectral dictionary into the measurements of our operator.

On test HSIs as in Section 6.3.2 with a small, spectral dictionary $\boldsymbol{D}$ and the TV$\otimes\ell_1$ nuclear norm, this approach does show some success. We must note, however, that the memory requirements under `operfact` for this measurement model are enormous. Any real test of this approach would necessarily require a larger dictionary $\boldsymbol{D}$, and therefore more memory. We present this here to provide a more complete catalog of our work, but we do not have the resources at present to pursue this further.

Note that we also did not utilize any nonnegativity constraints on the abundances or spectra, nor did we consider any constraints on the sum of the abundances at each pixel. These additional constraints may lead to better results, but at the moment, `operfact` is not equipped to handle them.

6.5 Discussion

Despite the difficulties in moving to a fully real hyperspectral image, the work with USGS spectral library data does demonstrate that nuclear norm denoising does work outside of fully synthetic data. Just as with the synthetic denoising problems, tuning the nuclear norm to the factor structure of the underlying operator is critical, and our tests here also display this.

We emphasize that this approach to HSI denoising is based on a truly convex problem while utilizing the linear mixture model in an unsupervised fashion (i.e., we use the spectral library splib06a to generate the images but not to denoise them). Even though we solve our problems using a nonconvex approach (alternating minimization), we believe that the power of using nuclear norms lies in the fact that we seek the solution to a convex problem at heart. Additionally, the models we consider here are enabled by the use of operators in the nuclear norm framework; we would not have been able to take advantage of the two-dimensional abundance structure using a simple matrix factorization.

In the next chapter, we move away from denoising problems to consider an application of nuclear norms in self-calibration problems.

Chapter 7

Application: Self-calibration

This chapter applies the nuclear norm framework to the self-calibration problem discussed in Section 2.1.2.4. In particular, we show how lifting the bilinear measurement model to the space of operators allows us to consider more complicated signal models than the matrix lifting schemes present in the literature. Nuclear norms then serve as natural regularizers in the lifted problem. Furthermore, we can improve the success rate of self-calibration by choosing nuclear norms that best match the structure of the lifted models.

7.1 Overview

Assume that we have observations of a signal $\boldsymbol{y} \in \mathbb{R}^N$ given by

$$\boldsymbol{b} = \boldsymbol{T}(\boldsymbol{x})\boldsymbol{y} + \boldsymbol{z}, \tag{7.1}$$

where $\boldsymbol{T}(\boldsymbol{x}) \in \mathbb{R}^{L\times N}$ is a linear map depending on a vector of parameters $\boldsymbol{x} \in \mathbb{R}^M$ and $\boldsymbol{z} \in \mathbb{R}^L$ is noise. We may think of $\boldsymbol{T}(\boldsymbol{x})$ as representing a sensor array that depends on calibration parameters $\boldsymbol{x}$.

The *self-calibration* problem is to recover the signal $\boldsymbol{y}$ without knowing the calibration parameters $\boldsymbol{x}$. That is, we wish to use the observation of the sensors without performing a calibration step first. As we discussed in Section 2.1.2.4, this problem has applications in direction-of-arrival estimation [FW88; FW91; See94; NS96; LY06; Liu+11], distributed sensing [BN07; BN08; WRS08; LB14], acoustic arrays [MDO11], and radar imaging [ZWBY14].

Note that if $\boldsymbol{T}(\boldsymbol{x})$ depends linearly on $\boldsymbol{x}$ then recovering $(\boldsymbol{x}, \boldsymbol{y})$ is a bilinear inverse problem. In this chapter, we consider such situations and show how the

`operfact` tool may be used with nuclear norms to perform self-calibration. Here we work with examples where $L < N$. That is, the number of measurements observed is smaller than the dimension of the signal. Even if we knew $\boldsymbol{T}(\boldsymbol{x})$, this problem would be underdetermined. We rely on structural assumptions combined with appropriate nuclear norms to recover the structured signal and calibration parameters.

7.1.1 Roadmap

We review the self-calibration literature in Section 7.2. Then in Section 7.3 we describe an operator lifting model for self-calibration and its implementation in `operfact`. We perform numerical experiments to demonstrate the effectiveness of the nuclear norm framework with single snapshots (Section 7.4.1), multiple snapshots (Section 7.4.2), and two-dimensional signals (Section 7.4.3).

7.2 Related work

In this section we discuss recent work on self-calibration to provide context for our numerical experiments.

7.2.1 Linear least squares

Bolzano and Nowak [BN07; BN08] consider the problem of self-calibration in sensor networks. They assume an array of n sensors that take measurements $\{x_j\}_{j=1}^n$. Each sensor, however, reports its measurements subject to an unknown gain α_j and offset β_j resulting in the observations

$$y_j = \frac{x_j - \beta_j}{\alpha_j}.$$

In vector notation this becomes

$$\boldsymbol{x} = \boldsymbol{Y}\boldsymbol{\alpha} + \boldsymbol{\beta},$$

where $\boldsymbol{Y}$ is an $n \times n$ diagonal matrix with entries y_j. They refer to this as a single *snapshot*—a report from all the sensors at a particular time.

To recover $\boldsymbol{x}$ from the observations $\boldsymbol{Y}$, they assume that the true signal $\boldsymbol{x}$ lies in a low-dimensional subspace. For instance, the signal may be bandlimited

and therefore lie in the span of a smaller number of sinusoids. Letting $\boldsymbol{P}$ be the orthogonal projector onto the complement of this subspace, we see that

$$\boldsymbol{P}\boldsymbol{x} = \boldsymbol{Y}\boldsymbol{\alpha} + \boldsymbol{\beta} = \boldsymbol{0}.$$

By considering k snapshots they obtain a linear system

$$\boldsymbol{P}(\boldsymbol{Y}_i\boldsymbol{\alpha} + \boldsymbol{\beta}) = \boldsymbol{0}, \text{ for } i = 1, \ldots, k.$$

If the signal subspace has dimension r, then this system represents $k(n-r)$ equations in $2n$ unknowns. They show that it is possible to perform self-calibration for k large enough. The subsequent work of Lipor and Balzano [LB14] allows for noisy measurements.

The very recent work of Ling and Strohmer [LS16] extends this approach of linearizing the bilinear measurements in various self-calibration models. They provide rigorous guarantees for recovery and address the question of stability with noisy measurements.

7.2.2 Calibrating compressed sensing

Consider the compressed sensing problem, where we observe

$$\boldsymbol{b} = \boldsymbol{T}\boldsymbol{y}, \tag{7.2}$$

with $\boldsymbol{T} \in \mathbb{R}^{L\times N}$ and $\boldsymbol{y} \in \mathbb{R}^n$ a sparse vector. A typical compressed sensing setting would assume that $\boldsymbol{T}$ is a known measurement matrix (with $L < N$) and attempt to recover the sparse vector $\boldsymbol{y}$. If, however, the sensing matrix $\boldsymbol{T}$ is unknown (or partially known) we have a calibration problem.

Gribonval et al. [GCD12] consider the case where $\boldsymbol{T}$ is known but subject to unknown gains. That is, the observations take the form

$$\boldsymbol{b} = \boldsymbol{D}\boldsymbol{T}\boldsymbol{y}, \tag{7.3}$$

where $\boldsymbol{D}$ is a diagonal matrix. In other words, each observation b_i in (7.3) corresponds to that of (7.2) multiplied by an unknown gain d_i.

To blindly calibrate this compressed sensing problem, they assume that they see the observations from multiple signals in the form

$$\boldsymbol{B} = \boldsymbol{D}\boldsymbol{T}\boldsymbol{Y},$$

where the matrix $\boldsymbol{Y} \in \mathbb{M}^{N\times Q}$ has Q unknown sparse signals in $\mathbb{R}^N$ as its columns.

Provided that none of the gains d_i is zero, the diagonal matrix $\boldsymbol{D}$ is invertible with $\boldsymbol{\Delta} = \boldsymbol{D}^{-1}$. Then the self-calibration problem becomes

$$\underset{(\boldsymbol{Y},\boldsymbol{\Delta})}{\text{minimize}} \quad \|\boldsymbol{Y}\|_{\ell_1} \quad \text{subject to} \quad \boldsymbol{\Delta B} = \boldsymbol{TY} \quad \text{and} \quad \operatorname{tr}(\boldsymbol{\Delta}) = \delta,$$

where the trace constraint serves to exclude the trivial solution $(\boldsymbol{0}, \boldsymbol{0})$. This problem is now convex, and they perform numerical experiments to demonstrate that this approach to self-calibration works for compressed sensing problems provided that the number Q of training signals is large enough.

Bilen et al. [BPGD14] extend this work to the space of complex signals. That is, they consider both the gain/amplitude calibration here as well as phase calibration. Recent work of Wang and Chi [Wan16] provide theoretical guarantees on the calibration procedure when the signals are sparse in the Fourier basis (i.e., $\boldsymbol{T}$ is a Fourier matrix).

7.2.3 A lifting approach

Our motivation is similar to the preceding work in that we wish to convexify the self-calibration problem. We, however, frame the problem by lifting to the space of operators. In particular, we follow the work of Ling and Strohmer [LS15b], and assume that the measurements (7.1) take the special form

$$\boldsymbol{b} = \boldsymbol{DTy} + \boldsymbol{z}, \quad \text{and} \quad \boldsymbol{D} = \operatorname{diag}(\boldsymbol{Sx}), \tag{7.4}$$

where we now assume that $\boldsymbol{T} \in \mathbb{R}^{L\times N}$ and $\boldsymbol{S} \in \mathbb{R}^{L\times M}$ are known. That is, the gains $\boldsymbol{D}$ belong to a low-dimensional subspace. The goal now is to recover the vector $\boldsymbol{x} \in \mathbb{R}^M$ that determines the gains along with the signal $\boldsymbol{y}$.

We see that this model is a generalization of the blind deconvolution model discussed in Section 2.1.2. Indeed, Ling and Strohmer took inspiration from the lifting procedure of Ahmed et al. [ARR14].

Assume that the measurements have no noise. We can write the entries of $\boldsymbol{b}$ as

$$b_l = \operatorname{tr}(\boldsymbol{M}_l^{\mathsf{t}}(\boldsymbol{xy}^{\mathsf{t}})) = \langle \boldsymbol{M}_l, \boldsymbol{xy}^{\mathsf{t}} \rangle, \quad \text{where} \quad \boldsymbol{M}_l = \boldsymbol{s}_l \boldsymbol{t}_l^{\mathsf{t}},$$

where $\boldsymbol{s}_l$ is the lth row of $\boldsymbol{S}$, and $\boldsymbol{T}_l$ is the lth row of $\boldsymbol{T}$.

If we set $\boldsymbol{A} = \boldsymbol{xy}^{\mathsf{t}}$, then we can write the measurement vector $\boldsymbol{b}$ as

$$\boldsymbol{b} = \boldsymbol{\mu}(\boldsymbol{A}),$$

where $\boldsymbol{\mu}$ is a linear measurement map such that the lth entry of $\boldsymbol{\mu}(\boldsymbol{A})$ is $\operatorname{tr}(\boldsymbol{M}_l^{\mathsf{t}}\boldsymbol{A})$. That is, the bilinear measurement model becomes a linear measurement model on matrices.

They assume that the true signal $\boldsymbol{y}$ is sparse and solve the self-calibration problem

$$\underset{\boldsymbol{A}}{\text{minimize}} \quad \|\boldsymbol{A}\|_{\ell_1} \quad \text{subject to} \quad \boldsymbol{\mu}(\boldsymbol{A}) = \boldsymbol{b}. \tag{7.5}$$

They call this procedure SparseLift.

They also provide a result on the probability of recovery.

Theorem 7.2.1 (ℓ_1-minimization for self-calibration [LS15b, Thm. 3.1])**.** *In the model (7.4), assume that $\boldsymbol{S} \in \mathbb{R}^{L\times M}$ satisfies $\boldsymbol{S}^*\boldsymbol{S} = \mathbf{I}_M$ (with $M \le L$), $\boldsymbol{x} \in \mathbb{R}^M$ is sparse, and $\boldsymbol{y} \in \mathbb{R}^n$ is s-sparse. Further assume that $\boldsymbol{T} \in \mathbb{M}^{L\times N}$ with $L < N$ has independent standard Gaussian entries. Then the solution $\widehat{\boldsymbol{A}}$ to (7.5) equals $\boldsymbol{x}\boldsymbol{y}^{\mathsf{t}}$ with probability at least $1 - \mathcal{O}(L^{-\alpha+1})$ provided that*

$$\frac{L}{\log^2 L} \ge C_\alpha \mu_{max}^2 M s \log(Ms),$$

where the constant C_α grows linearly with α, and $\mu_{\max}$ is the largest absolute entry in $\sqrt{L}\boldsymbol{S}$.

Therefore, the number L of measurements required scales roughly with Ms, the size of the subspace containing the unknown gains multiplied by the sparsity of the signal. A similar result holds when $\boldsymbol{T}$ is a partial Fourier matrix.

In their numerical experiments, they also consider using the $\|\cdot\|_{1,2}$ norm—the sum of the Euclidean norms of the rows—to enforce column-sparsity of $\boldsymbol{A}$. In the nuclear norm framework this is the $\ell_2 \otimes \ell_1$ nuclear norm, and its use makes sense as $\boldsymbol{A} = \boldsymbol{x}\boldsymbol{y}^{\mathsf{t}} = \boldsymbol{x} \otimes \boldsymbol{y}$, where $\boldsymbol{y}$ is sparse but $\boldsymbol{x}$ is not.

Flinth [Fli16] extends the theoretical results of Ling and Strohmer [LS15b; LS15a] to handle self-calibration and demixing problems with the $\|\cdot\|_{1,2}$ norm. He obtains qualitatively similar guarantees, but his results on stability suggest that the mixed norm will perform better than the ℓ_1 norm.

7.2.4 Our work

We also follow the lifting approach of Ling and Strohmer [LS15b], but we consider a lifting to the operator space to allow for two different extensions: multiple snapshots and two-dimensional signals. Under the lifted operator model, we

can replace the ℓ_1 regularizer in (7.5) with a nuclear norm that can account for the shared structure between snapshots, or the two-dimensional structure of a signal. While the above-mentioned approach in [GCD12; BPGD14] may be able to accommodate such structured signals, the authors do not discuss this possibility.

The remainder of this chapter discusses the operator lifting model and its implementation in `operfact` along with numerical experiments to demonstrate its use.

7.3 The operator measurement model

In this section we consider the lifting approach from Ling and Strohmer [LS15b] in the operator setting. We show how we can then use it to model self-calibration problems in `operfact`.

7.3.1 Assumptions

Let $\boldsymbol{Y} \in \mathbb{M}^{p\times q}$ be a set of q signals in $\mathbb{R}^p$, and assume that we observe

$$\boldsymbol{B} = \boldsymbol{T}(\boldsymbol{x})\boldsymbol{Y} + \boldsymbol{Z} \in \mathbb{M}^{d\times q}, \tag{7.6}$$

where $\boldsymbol{T}(\boldsymbol{x}) \in \mathbb{R}^{d\times p}$ is a linear operator depending on a vector of parameters $\boldsymbol{x} \in \mathbb{R}^m$, and $\boldsymbol{z} \in \mathbb{M}^{d\times q}$ is noise.

Following Ling and Strohmer [LS15b], we assume that the sensing operator $\boldsymbol{T}(\boldsymbol{x})$ takes the form

$$\boldsymbol{T}(\boldsymbol{x})\colon \boldsymbol{x} \mapsto \operatorname{diag}(\boldsymbol{S}\boldsymbol{x})\boldsymbol{T}_0,$$

where $\boldsymbol{S} \in \mathbb{R}^{d\times m}$ and $\boldsymbol{T}_0 \in \mathbb{R}^{d\times p}$ are known linear transformations. This assumption corresponds to using a known sensing matrix $\boldsymbol{T}_0$ with unknown gains $\operatorname{diag}(\boldsymbol{S}\boldsymbol{x})$ that belong to the subspace given by $\operatorname{range}(\boldsymbol{S})$. For simplicity, we henceforth refer to $\boldsymbol{T}_0$ simply as $\boldsymbol{T}$.

For the moment take the noise $\boldsymbol{Z} = \boldsymbol{0}$, and so we can then write the lth column of $\boldsymbol{B}$ as

$$\boldsymbol{b}_l = \operatorname{diag}(\boldsymbol{S}\boldsymbol{x})\boldsymbol{T}\boldsymbol{y}_l,$$

where $\boldsymbol{y}_l$ is the lth column of $\boldsymbol{Y}$.

Hence the jl-entry of $\boldsymbol{B}$ is

$$\begin{aligned} b_{jl} &= [\operatorname{diag}(\boldsymbol{S}\boldsymbol{x})]_j \cdot [\boldsymbol{T}\boldsymbol{y}_l]_j \\ &= \langle \boldsymbol{s}_{j:}, \boldsymbol{x} \rangle \langle \boldsymbol{t}_{j:}, \boldsymbol{y}_l \rangle, \end{aligned}$$

where $\boldsymbol{s}_{j:}$ is the jth row of $\boldsymbol{S}$ and $\boldsymbol{t}_{j:}$ is the jth row of $\boldsymbol{T}$.

Now let $\boldsymbol{T}_{jl}$ be the matrix whose lth column is $\boldsymbol{t}_{j:}$ and whose remaining entries are zero. Then we define the operator $\mathcal{M}_{jl} \in \mathbb{O}^{m \times 1 \otimes p \times q}$ such that

$$\mathcal{M}_{jl} = \boldsymbol{s}_{j:} \otimes \boldsymbol{T}_{jl}.$$

Let $\mathcal{A}^\natural = \boldsymbol{x} \otimes \boldsymbol{Y} \in \mathbb{O}^{m \times 1 \otimes p \times q}$, and observe that

$$\langle \mathcal{M}_{jl}, \mathcal{A}^\natural \rangle = \langle \boldsymbol{s}_{j:}, \boldsymbol{x} \rangle \langle \boldsymbol{T}_{jl}, \boldsymbol{Y} \rangle = \langle \boldsymbol{s}_{j:}, \boldsymbol{x} \rangle \langle \boldsymbol{t}_{j:}, \boldsymbol{y}_l \rangle = b_{jl},$$

where we have used the inner product identity (3.8).

In this way, we see that the observations (7.6) may be written as the result of applying a linear map to the rank-1 operator $\mathcal{A}^\natural$. That is,

$$\boldsymbol{B} = \operatorname{diag}(\boldsymbol{S}\boldsymbol{x})\boldsymbol{T}\boldsymbol{Y} + \boldsymbol{Z} = \boldsymbol{\mu}(\mathcal{A}^\natural) + \boldsymbol{Z} \quad \text{where} \quad [\boldsymbol{\mu}(\mathcal{A}^\natural)]_{jl} = \langle \mathcal{M}_{jl}, \mathcal{A}^\natural \rangle. \tag{7.7}$$

The adjoint is then

$$\mu^*(\boldsymbol{B}) = \sum_{jl} b_{jl} \mathcal{M}_{jl}. \tag{7.8}$$

To perform self-calibration we then solve the convex problem

$$\underset{\mathcal{A}}{\text{minimize}} \quad \frac{1}{2} \|\boldsymbol{B} - \mu(\mathcal{A})\|_{\ell_2}^2 + \lambda N_{X,Y}(\mathcal{A}), \tag{7.9}$$

where $N_{X,Y}$ is a nuclear norm chosen to match the structure of $\boldsymbol{x} \otimes \boldsymbol{Y}$, and $\lambda \geq 0$ is a penalty constant.

7.3.2 Implementation in `operfact`

To implement the measurements (7.7) in `operfact` we create a custom `Measurement` class (see Section 4.5). Note that `operfact` expects a vector of observations, and so the implementation returns $\operatorname{vec}(\boldsymbol{B})$ instead of $\boldsymbol{B}$ itself.

The following listing shows a partial implementation `SelfCalibMeasurement`, the custom measurement class.

```
import scipy.fftpack
from operfact import operators, measurements

class SelfCalibMeasurement(measurements.Measurement):
    """Implements the self-calibration measurement map"""
    def __init__(self, d, m, p, q):
        self.dim_amb = d
        self.shape = (m, 1, p, q)
        self.nmeas = d * q
        self.S = scipy.fftpack.dct(np.eye(d), norm='ortho')[:, 0:m
            ]
        self.T = rand_gaussianmat((d, p), False)

    def apply(self, oper):
        assert isinstance(oper, operators.DyadsOperator)
        # NB: broadcasting elementwise multiplication to do diag(
            vec) @ matrix
        temp = sum([(self.S @ oper.lfactors[r]) * (self.T @ oper.
            rfactors[r])
                    for r in range(oper.nfactors)])
        return temp.flatten(order='F')  # return a vector

    def matapply(self, mat):
        assert (self.shape[3] == 1)
        return np.vstack([np.matrix(self.S[l,:])*(mat*np.matrix(
            self.T[l,:]).T) for l in range(self.nmeas)])

    def initfrommeas(self, meas):
        out = operators.ArrayOperator(np.zeros(self.shape))
        for i in range(self.dim_amb):
            s_i = self.S[i:i+1, :].T  # force return 2D array
            a_i = self.T[i:i+1, :].T  # force return 2D array
            for j in range(self.shape[3]):
                Yij = np.zeros(self.shape[2:4])
                Yij[:, j:j+1] = a_i  # a_i is 2D array
                Aij = operators.DyadsOperator([s_i, ], [Yij, ])
                out += meas[j*self.dim_amb + i] * Aij.
                    asArrayOperator()
        return out
```

The implementation here is straightforward and includes an `apply` function that works on an operator in dyadic form as well as a `matapply` function that works

on the lifted matrix.[1] The `initfrommeas` function implements the adjoint (7.8) that we use to initialize the alternating solver.

Note that this implementation assumes that $\boldsymbol{T}$ is Gaussian and $\boldsymbol{S}$ is a partial DCT matrix. We use the DCT since CVXPY does not natively handle complex variables.

7.4 Numerical results

In this section we examine three measurement models and perform numerical experiments to explore the use of nuclear norms in self-calibration problems.

7.4.1 Single snapshot

First we consider the case of a single snapshot (i.e., $q = 1$) with $d = 128$ uncalibrated measurements of an s-sparse signal $\boldsymbol{y} \in \mathbb{R}^{256}$. We allow both the sparsity s and the dimension of the parameter vector $\boldsymbol{x} \in \mathbb{R}^m$ to range independently from 0 to 15. As mentioned in the last section, we take $\boldsymbol{S} \in \mathbb{R}^{128\times m}$ to be a partial DCT matrix, and $\boldsymbol{T} \in \mathbb{R}^{128\times 256}$ to be a Gaussian matrix. This is the same setting as Ling and Strohmer [LS15b].

We solve the self-calibration problem (7.9) using the $\ell_1 \otimes \ell_1$ and $\ell_2 \otimes \ell_1$ nuclear norms as regularizers. Full experimental details appear in Section D.1 of the appendix.

Figure 7.1 shows the phase transitions for successful self-calibration at various noise levels using the convex solver `matsolve`. In each facet we display the results in a grid, with each square representing a choice for the number s of nonzeros in $\boldsymbol{y}$, and the length m of $\boldsymbol{x}$. The intensity of the square corresponds to the "success percentage" over 10 random trials: white represents a 100% success rate while black represents 0%. We address the noiseless and noisy cases individually.

[1] The `matapply` function here assumes a single snapshot (i.e., $q = 1$), but it could be modified to handle the lifted operator case with multiple snapshots. In that case, however, we tend to use nuclear norms that do not work with the direct convex solver.

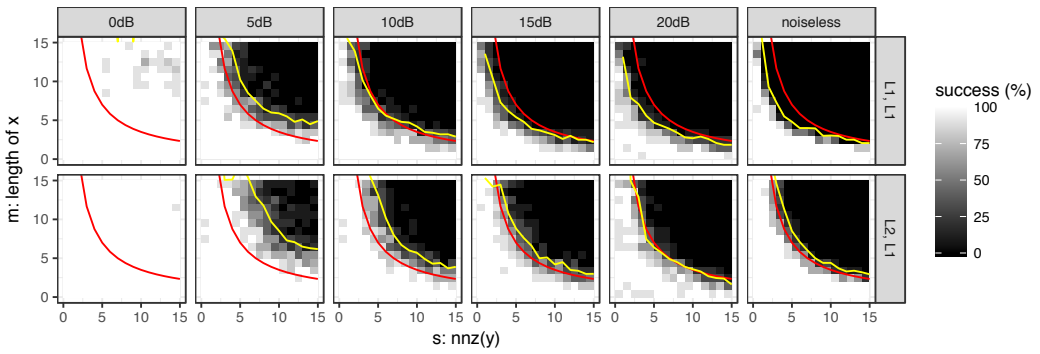


Figure 7.1: **Phase transitions for single-snapshot self-calibration by SNR.** Each plot shows the average success rate for the self-calibration problem over 10 trials as a function of the length m of the parameters $\boldsymbol{x}$ and the number s of nonzeros in the sparse signal $\boldsymbol{y}$. The success rate is displayed as a grayscale gradient from black (0%) to white (100%). The red curve indicates the phase transition observed by Ling and Strohmer [LS15b], while the yellow curve gives our empirical 50% success rate computed by logistic regression. We facet our plot on the SNR (columns) and nuclear norm (rows). All experiments use the convex solver `matsolve`.

7.4.1.1 The noiseless case

In the noiseless case, we define success as Ling and Strohmer do,

$$\frac{\|\widehat{\mathcal{A}} - \mathcal{A}^\natural\|_{\ell_2}}{\|\mathcal{A}^\natural\|_{\ell_2}} \leq 0.01,$$

where $\mathcal{A}^\natural = \boldsymbol{x} \otimes \boldsymbol{y}$ is the ground truth, and $\widehat{\mathcal{A}}$ is the solution from the solver. We project $\widehat{\mathcal{A}}$ to a rank-1 operator (here, a matrix). Note that recovery of $\boldsymbol{x}$ and $\boldsymbol{y}$ is accomplished by the truncated SVD, and we cannot avoid a scaling ambiguity between the factors.

Ling and Strohmer [LS15b] empirically observed success when $ms < 70$. Their experiments considered partial Fourier $\boldsymbol{S}$, and according to Theorem 7.2.1, we expect that using a partial DCT $\boldsymbol{S}$ will require twice as many measurements.[2] And so we plot a red curve $ms = 35$ indicating the rough location of the expected phase transition. A yellow curve indicates our empirical 50% success rate. Our empirical results indeed match closely with theirs in the noiseless case.

[2] The quantity $\mu^2_{\max}$ for a Fourier matrix is 1, while it is 2 for a partial DCT.

7.4.1.2 The noisy case

For the noisy cases, we let $\boldsymbol{Z}$ in (7.7) have independent NORMAL$(0, \sigma^2)$ entries, and we choose

$$\sigma^2 = 10 \log_{10} \left(\frac{10^{-\mathrm{SNR}/10} \|\mathcal{A}^\natural\|_{\ell_2}^2}{mpq} \right),$$

where SNR is a desired signal-to-noise ratio in decibels (dB). Note that we measure the SNR of this problem as if we were corrupting the entries of $\mathcal{A}^\natural$ with noise just as in Section 5.4.2 of the denoising experiments. (We also calculate the penalty constant λ using the procedure detailed in Section B.2 of the appendix.) The SNR is independent of the number of measurements.

We measure the reconstruction signal-to-distortion ratio (RSDR) as

$$\mathrm{RSDR} := 10 \log_{10} \left(\frac{\|\mathcal{A}^\natural\|_{\ell_2}^2}{\|\widehat{\mathcal{A}} - \mathcal{A}^\natural\|_{\ell_2}^2} \right). \tag{7.10}$$

We define success as having RSDR $>$ SNR. That is, success corresponds with displaying robustness to noise.

The remaining panels in Figure 7.1 show the performance of the single-snapshot self-calibration procedure with various noise levels. As in the denoising experiments, we see that the phase transition at higher SNRs (i.e., lower noise) roughly matches the noiseless setting and that the relative performance improves as the SNR falls (see Section 5.4.2). Again, this does not mean that we recover the signal better in lower-SNR regimes; it means that the recovery is better relative to the noise level.

7.4.1.3 Solution with alternating minimization

While the previous experiments can be completed using the convex solver `matsolve`, we will require the use of our alternating minimization solver `altminsolve` as we consider different nuclear norms. Here we wish to determine the effect of the solver rank on the performance. Since we know that the true solution of the problem should be rank-1, we would like to use fewer dyads in the solver.

Figure 7.2 shows the effect of solving (7.9) with alternating minimization at an SNR of 15dB with different solver ranks. Using 16 dyads results in empirical success curves that resemble those of the convex solver. Restricting the number of dyads, however, does result in successes for combinations of larger m and s.

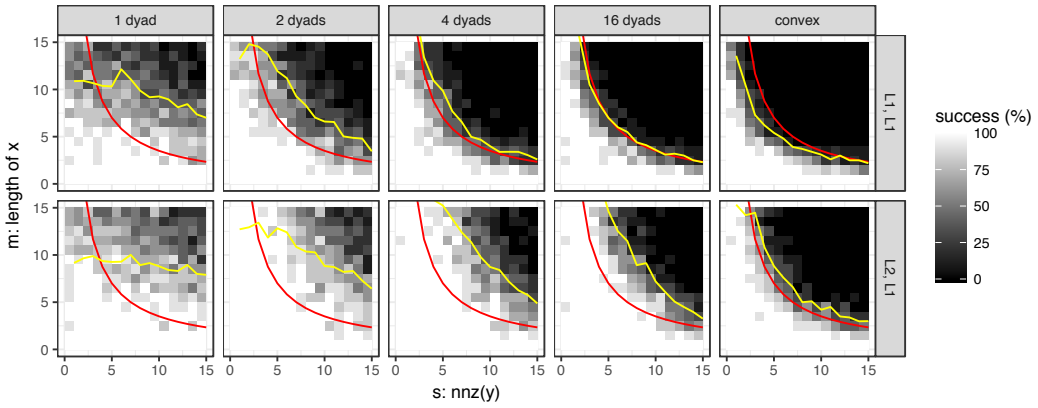


Figure 7.2: **Phase transitions for single-snapshot self-calibration by solver rank.** Each plot shows the average success rate for the self-calibration problem over 10 trials as a function of the length m of the parameters $\boldsymbol{x}$ and the number s of nonzeros in the sparse signal $\boldsymbol{y}$. The success rate is displayed as a grayscale gradient from black (0%) to white (100%). The red curve indicates the phase transition observed by Ling and Strohmer [LS15b], while the yellow curve gives our empirical 50% success rate computed by logistic regression. We facet our plot on the solver rank (columns) and nuclear norm (rows). The "convex" column indicates the convex solver `matsolve`. All experiments use an SNR of 15dB.

At the other end, with one dyad, the solver no longer shows a sharp phase transition and does not perform as well for lower s, the number of nonzeros in $\boldsymbol{y}$. Even though we expect the solution to be rank-1, using even one extra dyad considerably improves the behavior. We suspect this reflects results in low-rank optimization that show reductions (or eliminations) in spurious local minima for high enough solver rank (see [BM03; BM04]).

7.4.2 Multiple snapshots

We now turn our attention to the multiple-snapshot case (i.e., $q > 1$). That is, we have the $128 \cdot q$ uncalibrated measurements

$$\boldsymbol{B} = \operatorname{diag}(\boldsymbol{S}\boldsymbol{x})\boldsymbol{T}\boldsymbol{Y} + \boldsymbol{Z} = \boldsymbol{\mu}(\mathcal{A}^\natural) + \boldsymbol{Z} \in \mathbb{M}^{128\times q},$$

where the q columns of $\boldsymbol{Y}$ are s-sparse signals in $\mathbb{R}^{256}$. Again, both the length of the parameter vector $\boldsymbol{x} \in \mathbb{R}^m$ and the sparsity s vary independently. We still construct $\boldsymbol{S} \in \mathbb{M}^{128\times m}$ as a partial DCT matrix and $\boldsymbol{T} \in \mathbb{R}^{128\times 256}$ as a Gaussian matrix.

To perform self-calibration, we solve (7.9) using a nuclear norm $N_{X,Y}$ chosen specifically for the structure of $\mathcal{A}^\natural = \boldsymbol{x} \otimes \boldsymbol{Y}$. We test three different models for generating the set of snapshots $\boldsymbol{Y}$.

- **Independent sparse snapshots.** Under this model, the columns of $\boldsymbol{Y}$ each have s nonzeros with their locations chosen uniformly at random and magnitudes drawn independently from a standard normal distribution. This is the model considered in [GCD12; BPGD14]. The matrix $\boldsymbol{Y}$ will have sq nonzeros, and we believe that the $\ell_2 \otimes \ell_1$ nuclear norm is therefore suited to recover $\mathcal{A}^\natural = \boldsymbol{x} \otimes \boldsymbol{Y}$.
- **Simultaneous sparsity.** Here the columns of $\boldsymbol{Y}$ have s nonzeros in identical locations with the entries drawn independently from a standard normal distribution. The matrix $\boldsymbol{Y}$ will then be row-sparse with standard normal rows, and we believe that the $\ell_2 \otimes (\ell_1 \otimes \ell_2)$ nuclear norm will suitably recover $\mathcal{A}^\natural$.
- **Identical sparse snapshots.** Finally we consider the case where the columns of $\boldsymbol{Y}$ are identical s-sparse vectors whose entries have been drawn independently from a standard normal distribution. The matrix $\boldsymbol{Y}$ will again be row-sparse but with rows having identical entries, and we therefore believe that the $\ell_2 \otimes (\ell_1 \otimes \ell_\infty)$ nuclear norm will best match the structure of $\mathcal{A}^\natural$.

We consider separately the effects of changing the nuclear norm and the number of snapshots. Full experimental details appear in Section D.2 of the appendix.

7.4.2.1 Matching nuclear norms to signal models

Figure 7.3 shows the results of a numerical experiment to test the efficacy of each of the nuclear norms mentioned above with each of the models for generating the snapshots $\boldsymbol{Y}$. Each panel shows the success rate under a single combination of signal generation and regularizer with 8 snapshots (i.e., $q = 8$). All experiments used the alternating minimization solver with 4 dyads at an SNR of 15dB. Success again is determined by the condition RSDR > SNR, where the recovery signal-to-distortion ratio (RSDR) is given by (7.10).

We can see that in the case where the columns of $\boldsymbol{Y}$ are independent, promoting row-sparsity of $\boldsymbol{Y}$ proves detrimental. As we expect, the $\ell_2 \otimes \ell_1$ nuclear norm

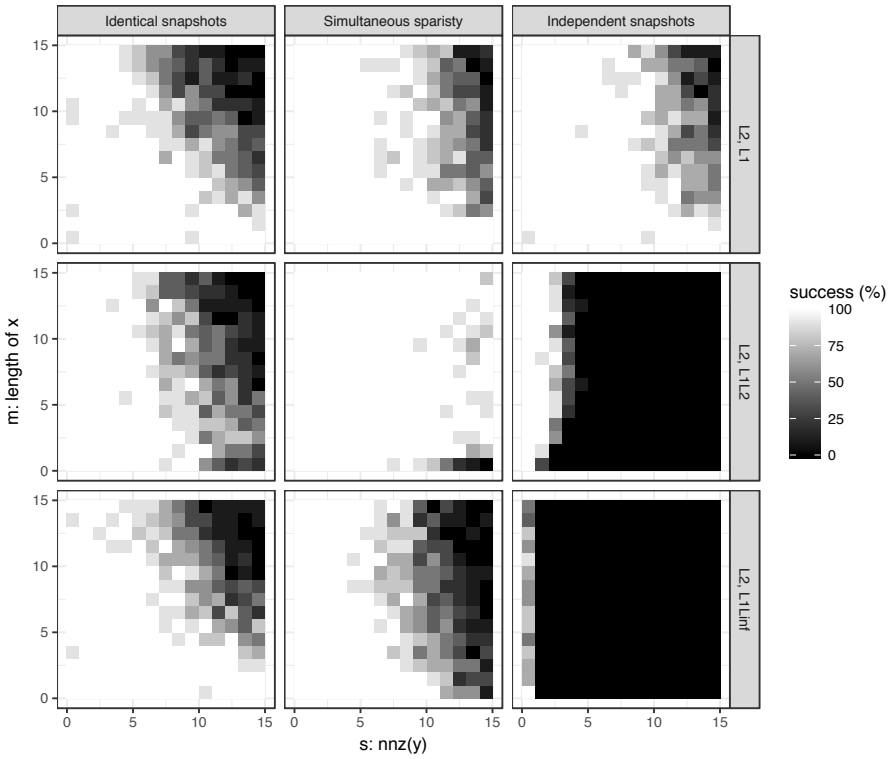


Figure 7.3: **Phase transitions for multiple-snapshot self-calibration by signal model and regularizer.** Each plot shows the average success rate for the self-calibration problem over 10 trials as a function of the length m of the parameters $\boldsymbol{x}$ and the number s of nonzeros in each snapshot $\boldsymbol{y}_i$. The success rate is displayed as a grayscale gradient from black (0%) to white (100%). We facet our plot on the signal model (columns) and nuclear norm (rows). All experiments use an SNR of 15dB.

performs best with this model. Note also that we see an improvement in success rates versus the single snapshot setting of the previous section. We will return to this point shortly.

When the snapshots are simultaneously sparse, the $\ell_2 \otimes (\ell_1 \otimes \ell_2)$ nuclear norm performs somewhat better, particularly as the dimension m of $\boldsymbol{x}$ grows. That is, we see some benefit from including the assumption of row-sparsity on $\boldsymbol{Y}$ when the calibration must account for a greater number of parameters.

Finally, when the columns of $\boldsymbol{Y}$ are identical, both the $\ell_2 \otimes \ell_1$ and $\ell_2 \otimes (\ell_1 \otimes \ell_\infty)$ nuclear norms perform well. Further examination shows that the $\ell_2 \otimes (\ell_1 \otimes \ell_\infty)$ nuclear norm has more overall success, but the difference is slight and a pattern to the differences is not clear.

Overall we see that the identical sparsity pattern model (under the $\ell_2 \otimes (\ell_1 \otimes \ell_2)$

nuclear norm) has the highest success for larger values of s and m. Indeed, we expect that the row-sparsity of the signal matrix $\boldsymbol{Y}$ is a "stronger" structure than the case of independent columns. We must also consider, however, that identical snapshots model also has row-sparse $\boldsymbol{Y}$. We suspect that the diversity of the entries allows for better opportunity to learn the parameters $\boldsymbol{x}$. While the matrix $\boldsymbol{Y}$ may be "simpler" in the identical snapshot case, it is less helpful in performing calibration.

7.4.2.2 Varying the number of snapshots

We now turn our attention to the effect of the number q of snapshots on the success rate. Figure 7.4 shows the results of using various values of q under the different snapshot models. For each model we regularize the self-calibration problem with the nuclear norms we already identified as providing the best performance.[3] As before we use the alternating minimization solver with 4 dyads and 15dB SNR.

In each instance, we see that increasing the number of snapshots generally improves the success rate of the self-calibration problem. That is, more snapshots allow for performing self-calibration as the number s of nonzeros in each snapshot increases and as the length m of the calibration parameters $\boldsymbol{x}$ grows.

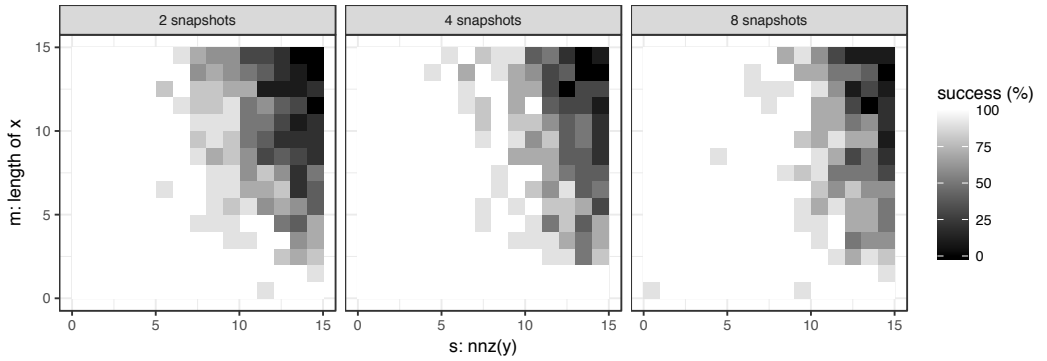
Consider fixed values for s and the length of $\boldsymbol{x}$. The intuition here is that even though we have more snapshots—and thus more entries of $\boldsymbol{Y}$ to recover—the number of measurements grows linearly with the number q of snapshots. Meanwhile, the number of calibration parameters in $\boldsymbol{x}$ remains unchanged. The additional snapshots give us more opportunity to correctly identify the entries of $\boldsymbol{x}$.

We do see, however, that the effect is somewhat less pronounced in the case of independent snapshots. Our intuition is that the shared structure between the columns in the other two models is relatively more beneficial as the number of snapshots grows.

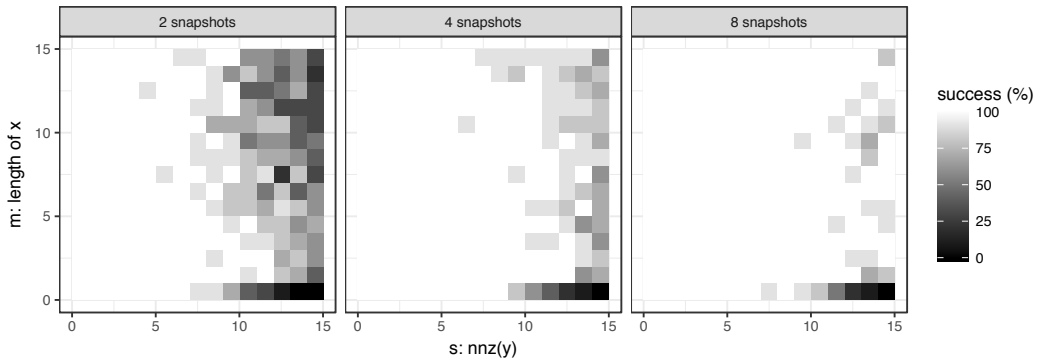
7.4.3 Two-dimensional signals

Our last numerical experiment concerns a self-calibration problem where the true signal has two-dimensional structure. We consider the signal $\boldsymbol{Y} \in \mathbb{M}^{p\times q}$

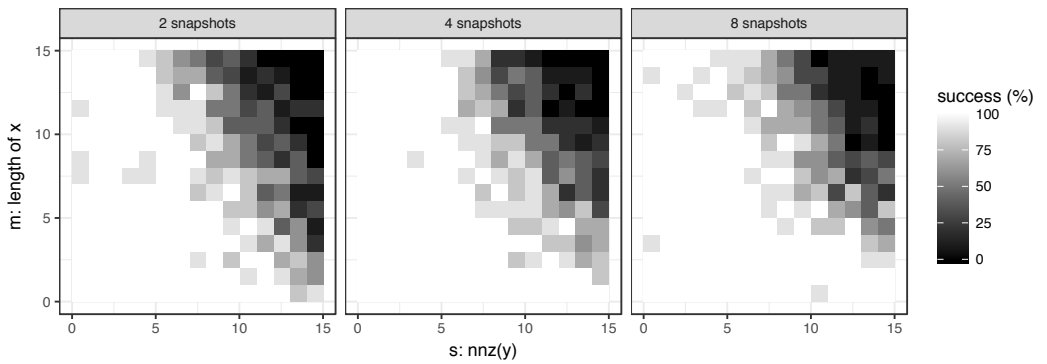
[3] In the case of identical snapshots, we chose to use the $\ell_2 \otimes (\ell_1 \otimes \ell_\infty)$ nuclear norm even though the performance difference was slight compared to the $\ell_2 \otimes \ell_1$ nuclear norm.



(a) Snapshots $\boldsymbol{y}_i$ are independent with the $\ell_2 \otimes \ell_1$ nuclear norm.



(b) Snapshots $\boldsymbol{y}_i$ are simultaneously sparse with the $\ell_2 \otimes (\ell_1 \otimes \ell_2)$ nuclear norm.



(c) Snapshots $\boldsymbol{y}_i$ are identical with the $\ell_2 \otimes (\ell_1 \otimes \ell_\infty)$ nuclear norm.

Figure 7.4: **Phase transitions for multiple-snapshot self-calibration by number of snapshots.** Each plot shows the average success rate for the self-calibration problem over 10 trials as a function of the length m of the parameters $\boldsymbol{x}$ and the number s of nonzeros in each snapshot $\boldsymbol{y}_i$. The success rate is displayed as a grayscale gradient from black (0%) to white (100%). We divide our results into three subfigures, each examining a single signal model and nuclear norm, and we facet each subfigure on the number q of snapshots observed. All experiments use an SNR of 15dB.

and the (noisy) measurements

$$\boldsymbol{b} = \operatorname{diag}(\boldsymbol{S}\boldsymbol{x})\boldsymbol{T}\operatorname{vec}(\boldsymbol{Y}) + \boldsymbol{z},$$

where the $\boldsymbol{S} \in \mathbb{M}^{d\times m}$, $\boldsymbol{x} \in \mathbb{R}^m$, and $\boldsymbol{T} \in \mathbb{M}^{d\times p}$ are as before. This is simply the single snapshot problem but instead of considering the true operator

$$\mathcal{A} = \boldsymbol{x} \otimes \operatorname{vec}(\boldsymbol{Y}) \in \mathbb{O}^{m\times 1\otimes pq\times 1},$$

we let

$$\mathcal{A} = \boldsymbol{x} \otimes \boldsymbol{Y} \in \mathbb{O}^{m\times 1\otimes p\times q}.$$

This distinction requires little computational change; we reuse the `operfact` implementation from Section 7.3.2 with an additional flag to indicate the use of a 2D signal. The benefit is that we can now apply nuclear norms that promote desired *matrix* structure in the signal $\boldsymbol{Y}$ as opposed to vector structure as in Section 7.4.1.

For this experiment, we observe 2048 uncalibrated measurements (15dB SNR) of a random rank-r matrix $\boldsymbol{Y} \in \mathbb{M}^{64\times 64}$. Again, $\boldsymbol{S} \in \mathbb{R}^{2048\times m}$ is a partial DCT matrix, and $\boldsymbol{T} \in \mathbb{R}^{2048\times 4096}$ Gaussian matrix. We allow the parameters m and r to vary independently and test the performance of self-calibration using alternating minimization (with 1 dyad) and the $\ell_2 \otimes S_1$ nuclear norm. Full experimental details appear in Section D.3 of the appendix.

Given our 2048 measurements and the fact that our structured operator $\mathcal{A} = \boldsymbol{x} \otimes \boldsymbol{Y}$ has $m + r(p+q) = m + 128r$ degrees of freedom, we have hope that self-calibration is possible for smaller values of m and r. Figure 7.5 shows the results of the experiment, and we indeed see success as we range m and r over small values. This small experiment provides encouraging results, but more efficient computational methods are necessary for a study of large two-dimensional signals.

7.5 Summary

In this chapter, we developed an operator lifting procedure that extends SparseLift [LS15b] to handle multiple snapshots and two-dimensional signals. This operator approach allows us to formulate convex programs—using nuclear norms—that can account for the shared structure between snapshots and the two-dimensional complexity of single snapshots. We see that natural pairings of nuclear norms to data models provide good recovery performance.

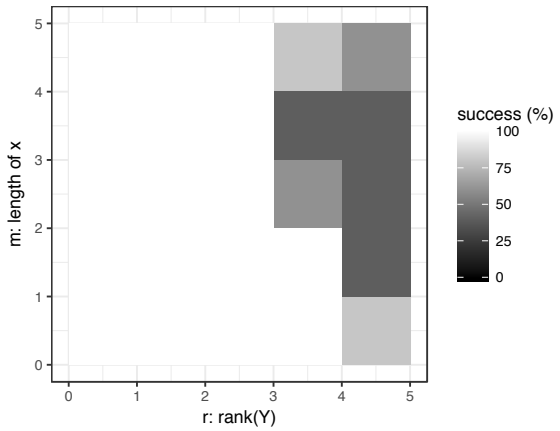


Figure 7.5: **Phase transition for 2D-signal self-calibration.** This plot shows the average success rate for the self-calibration problem (SNR 15dB) over 10 trials as a function of the length m of the parameters $\boldsymbol{x}$ and the rank r of the two-dimensional signal $\boldsymbol{Y}$. The success rate is displayed as a grayscale gradient from black (0%) to white (100%).

Our numerical approach utilized the extensibility of `operfact` to create problem-specific measurement objects. After all, this is a main reason for creating the software: rapid prototyping and testing of models.

Taken together with our experiments in denoising, we provide strong evidence that nuclear norms successfully promote the individual factor structures of matrices and operators. We can formulate convex programs to approximate operators having such distinguished structure. In some cases we can even use the alternating minimization to retrieve factorizations. The successes of the nuclear norm framework leave open the possibility for future theoretical and empirical study.

Bibliography

[AAJN16] A. Agarwal, A. Anandkumar, P. Jain, and P. Netrapalli. "Learning Sparsely Used Overcomplete Dictionaries via Alternating Minimization". In: *SIAM Journal on Optimization* 26.4 (Dec. 2016), pp. 2775–2799.

[AAN16] A. Agarwal, A. Anandkumar, and P. Netrapalli. "A Clustering Approach to Learning Sparsely-Used Overcomplete Dictionaries". In: *IEEE Transactions on Information Theory* (Sept. 2016), pp. 1–1.

[ACD15] A. Ahmed, A. Cosee, and L. Demanet. "A convex approach to blind deconvolution with diverse inputs". In: *Proceedings of the IEEE 6th International Workshop on Computational Advances in Multi-Sensor Adaptive Processing*. 2015, pp. 5–8.

[AEB06] M. Aharon, M. Elad, and A. Bruckstein. "K-SVD: An Algorithm for Designing Overcomplete Dictionaries for Sparse Representation". In: *IEEE Transactions on Signal Processing* 54.11 (Nov. 2006), pp. 4311–4322.

[AGKM16] S. Arora, R. Ge, R. Kannan, and A. Moitra. "Computing a Nonnegative Matrix Factorization—Provably". In: *SIAM Journal on Computing* 45.4 (Aug. 2016), pp. 1582–1611.

[AGM12] S. Arora, R. Ge, and A. Moitra. "Learning Topic Models – Going beyond SVD". In: *Proceedings of the IEEE 53rd Annual Symposium on Foundations of Computer Science*. IEEE, Dec. 2012, pp. 1–10.

[AGM14] S. Arora, R. Ge, and A. Moitra. "New Algorithms for Learning Incoherent and Overcomplete Dictionaries". In: *Proceedings of the 27th Conference on Learning Theory*. 2014, pp. 779–806.

[AGMM15] S. Arora, R. Ge, T. Ma, and A. Moitra. "Simple, Efficient, and Neural Algorithms for Sparse Coding". In: *Proceedings of the 28th Conference on Learning Theory*. 2015, pp. 113–149.

[AKKT10] H. Akbari, Y. Kosugi, K. Kojima, and N. Tanaka. "Detection and Analysis of the Intestinal Ischemia Using Visible and Invisible

Hyperspectral Imaging". In: *IEEE Transactions on Biomedical Engineering* 57.8 (2010), pp. 2011–2017.

[ALMT14] D. Amelunxen, M. Lotz, M. B. McCoy, and J. A. Tropp. "Living on the edge: phase transitions in convex programs with random data". In: *Information and Inference* 3.3 (2014), pp. 224–294.

[AN06] N. Alon and A. Naor. "Approximating the Cut-Norm via Grothendieck's Inequality". In: *SIAM Journal on Computing* 35.4 (Jan. 2006), pp. 787–803.

[ARR14] A. Ahmed, B. Recht, and J. Romberg. "Blind Deconvolution Using Convex Programming". In: *IEEE Transactions on Information Theory* 60.3 (Mar. 2014), pp. 1711–1732.

[AT06] A. Auslender and M. Teboulle. "Interior gradient and proximal methods for convex and conic optimization". In: *SIAM Journal on Optimization* 16.3 (2006), pp. 697–725.

[Bac13] F. Bach. "Convex relaxations of structured matrix factorizations". In: *arXiv.org* (Sept. 2013). arXiv: 1309.3117.

[Bas+93] R. Basedow et al. "The HYDICE instrument design and its application to planetary instruments". In: *Proceedings of the Workshop on Advanced Technologies for Planetary Instruments*. 1993.

[Bat82] R. H. T. Bates. "Astronomical speckle imaging". In: *Physics Reports* 90.4 (Oct. 1982), pp. 203–297.

[BBCE09] R. Balan, B. G. Bodmann, P. G. Casazza, and D. Edidin. "Painless Reconstruction from Magnitudes of Frame Coefficients". In: *Journal of Fourier Analysis and Applications* 15.4 (Mar. 2009), pp. 488–501.

[BCG11] S. R. Becker, E. J. Candès, and M. C. Grant. "Templates for convex cone problems with applications to sparse signal recovery". In: *Mathematical Programming Computation* 3.3 (2011), pp. 165–218.

[BDDW08] R. Baraniuk, M. Davenport, R. DeVore, and M. Wakin. "A Simple Proof of the Restricted Isometry Property for Random Matrices". In: *Constructive Approximation* 28.3 (Jan. 2008), pp. 253–263.

[BGC98] J. D. Bayliss, J. A. Gualtieri, and R. F. Cromp. "Analyzing hyperspectral data with independent component analysis". In: *Proceedings of the 26th AIPR Workshop: Exploiting New Image Sources and Sensors*. Ed. by J. M. Selander. SPIE, Mar. 1998, pp. 133–143.

[BGKP16] C. Bhattacharya, N. Goyal, R. Kannan, and J. Pani. "Non-negative Matrix Factorization under Heavy Noise". In: *Proceedings of the 33rd International Conference on Machine Learning*. 2016, pp. 1426–1434.

[Bio+12] J. M. Bioucas-Dias et al. “Hyperspectral Unmixing Overview: Geometrical, Statistical, and Sparse Regression-Based Approaches”. In: *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* 5.2 (Apr. 2012), pp. 354–379.

[BM03] S. Burer and R. D. C. Monteiro. “A nonlinear programming algorithm for solving semidefinite programs via low-rank factorization”. In: *Mathematical Programming* 95.2, Ser. B (2003), pp. 329–357.

[BM04] S. Burer and R. D. C. Monteiro. “Local Minima and Convergence in Low-Rank Semidefinite Programming”. In: *Mathematical Programming* 103.3 (Dec. 2004), pp. 427–444.

[BMMN11] M. Braverman, K. Makarychev, Y. Makarychev, and A. Naor. “The Grothendieck Constant is Strictly Smaller than Krivine’s Bound”. In: *Proceedings of the IEEE 52nd Annual Symposium on Foundations of Computer Science*. IEEE, 2011, pp. 453–462.

[BMP08] F. Bach, J. Mairal, and J. Ponce. “Convex Sparse Matrix Factorizations”. In: *arXiv.org* (Dec. 2008). arXiv: 0812.1869.

[BN01] A. Ben-Tal and A. Nemirovski. *Lectures on Modern Convex Optimization*. Analysis, Algorithms, and Engineering Applications. Philadelphia: Society for Industrial and Applied Mathematics, Jan. 2001.

[BN07] L. Balzano and R. Nowak. “Blind Calibration of Sensor Networks”. In: *Proceedings of the 6th International Symposium on Information Processing in Sensor Networks*. IEEE, 2007, pp. 79–88.

[BN08] L. Balzano and R. Nowak. “Blind Calibration of Networks of Sensors: Theory and Algorithms”. In: *Networked Sensing Information and Control*. Boston: Springer, 2008, pp. 9–37.

[BPGD14] C. Bilen, G. Puy, R. Gribonval, and L. Daudet. “Convex Optimization Approaches for Blind Sensor Calibration Using Sparsity”. In: *IEEE Transactions on Signal Processing* 62.18 (July 2014), pp. 4847–4856.

[BT97] D. P. Bertsekas and J. N. Tsitsiklis. *Parallel and Distributed Computation*. Numerical Methods. Belmont: Athena Scientific, Jan. 1997.

[BTCB14] J. J. Bruer, J. A. Tropp, V. Cevher, and S. R. Becker. “Time–Data Tradeoffs by Aggressive Smoothing”. In: *Advances in Neural Information Processing Systems 27 (NIPS 2014)*. 2014, pp. 1664–1672.

[BTCB15] J. J. Bruer, J. A. Tropp, V. Cevher, and S. R. Becker. "Designing Statistical Estimators That Balance Sample Size, Risk, and Computational Cost". In: *IEEE Journal of Selected Topics in Signal Processing* 9.4 (2015), pp. 612–624.

[BTR13] B. N. Bhaskar, G. Tang, and B. Recht. "Atomic Norm Denoising With Applications to Line Spectral Estimation". In: *IEEE Transactions on Signal Processing* 61.23 (Dec. 2013), pp. 5987–5999.

[Bun+07] O. Bunk et al. "Diffractive imaging for periodic samples: retrieving one-dimensional concentration profiles across microfluidic channels". In: *Acta Crystallographica Section A: Foundations of Crystallography* 63.4 (July 2007), pp. 306–314.

[CC70] J. D. Carroll and J.-J. Chang. "Analysis of individual differences in multidimensional scaling via an n-way generalization of "Eckart-Young" decomposition". In: *Psychometrika* 35.3 (Sept. 1970), pp. 283–319.

[CDS98] S. S. Chen, D. L. Donoho, and M. A. Saunders. "Atomic decomposition by basis pursuit". In: *SIAM Journal on Scientific Computing* 20.1 (1998), pp. 33–61.

[CESV13] E. J. Candès, Y. C. Eldar, T. Strohmer, and V. Voroninski. "Phase Retrieval via Matrix Completion". In: *SIAM Journal on Imaging Sciences* 6.1 (Feb. 2013), pp. 199–225.

[CJ13] V. Chandrasekaran and M. I. Jordan. "Computational and statistical tradeoffs via convex relaxation". In: *Proceedings of the National Academy of Sciences of the United States of America* 110.13 (2013), E1181–E1190.

[CJ16a] V. Cambareri and L. Jacques. "A Greedy Blind Calibration Method for Compressed Sensing with Unknown Sensor Gains". In: *arXiv.org* (Oct. 2016). arXiv: 1610.02851.

[CJ16b] V. Cambareri and L. Jacques. "Through the Haze: A Non-Convex Approach to Blind Calibration for Linear Random Sensing Models". In: *arXiv.org* (Oct. 2016). arXiv: 1610.09028.

[CJ95] J. Cadima and I. T. Jolliffe. "Loading and correlations in the interpretation of principle compenents". In: *Journal of Applied Statistics* 22.2 (Jan. 1995), pp. 203–214.

[Cla+07] R. N. Clark et al. *USGS digital spectral library splib06a*. Sept. 2007. URL: http://speclab.cr.usgs.gov/spectral.lib06.

[CLMW11] E. J. Candès, X. Li, Y. Ma, and J. Wright. "Robust principal component analysis?" In: *Journal of the ACM* 58.3 (May 2011), pp. 11–37.

[CLS15] E. J. Candès, X. Li, and M. Soltanolkotabi. “Phase Retrieval via Wirtinger Flow: Theory and Algorithms”. In: *IEEE Transactions on Information Theory* 61.4 (Apr. 2015), pp. 1985–2007.

[CMP11] A. Chai, M. Moscoso, and G. Papanicolaou. “Array imaging using intensity-only measurements”. In: *Inverse Problems* 27.1 (Jan. 2011), p. 015005.

[COR11] A. S. Charles, B. A. Olshausen, and C. J. Rozell. “Learning Sparse Codes for Hyperspectral Imagery”. In: *IEEE Journal of Selected Topics in Signal Processing* 5.5 (2011), pp. 963–978.

[CP10] E. J. Candès and Y. Plan. “Matrix Completion With Noise”. In: *Proceedings of the IEEE* 98.6 (Apr. 2010), pp. 925–936.

[CR02] S. F. Cotter and B. D. Rao. “Sparse channel estimation via matching pursuit with application to equalization”. In: *IEEE Transactions on Communications* 50.3 (Aug. 2002), pp. 374–377.

[CR09] E. J. Candès and B. Recht. “Exact Matrix Completion via Convex Optimization”. In: *Foundations of Computational Mathematics* 9.6 (Apr. 2009), pp. 717–772.

[CREK05] S. F. Cotter, B. D. Rao, K. Engan, and K. Kreutz-Delgado. “Sparse solutions to linear inverse problems with multiple measurement vectors”. In: *IEEE Transactions on Signal Processing* 53.7 (June 2005), pp. 2477–2488.

[CRPW12] V. Chandrasekaran, B. Recht, P. A. Parrilo, and A. S. Willsky. “The Convex Geometry of Linear Inverse Problems”. In: *Foundations of Computational Mathematics* 12.6 (2012), pp. 805–849.

[CRT06a] E. J. Candès, J. Romberg, and T. Tao. “Robust uncertainty principles: exact signal reconstruction from highly incomplete frequency information”. In: *IEEE Transactions on Information Theory* 52.2 (Jan. 2006), pp. 489–509.

[CRT06b] E. J. Candès, J. K. Romberg, and T. Tao. “Stable signal recovery from incomplete and inaccurate measurements”. In: *Communications on Pure and Applied Mathematics* 59.8 (Aug. 2006), pp. 1207–1223.

[CSPW11] V. Chandrasekaran, S. Sanghavi, P. A. Parrilo, and A. S. Willsky. “Rank-Sparsity Incoherence for Matrix Decomposition”. In: *SIAM Journal on Optimization* 21.2 (Apr. 2011), pp. 572–596.

[CSV13] E. J. Candès, T. Strohmer, and V. Voroninski. “PhaseLift: Exact and Stable Signal Recovery from Magnitude Measurements via Convex Programming”. In: *Communications on Pure and Applied Mathematics* 66.8 (Aug. 2013), pp. 1241–1274.

[CT05] E. J. Candès and T. Tao. “Decoding by Linear Programming”. In: *IEEE Transactions on Information Theory* 51.12 (Dec. 2005), pp. 4203–4215.

[CT10] E. J. Candès and T. Tao. “The Power of Convex Relaxation: Near-Optimal Matrix Completion”. In: *IEEE Transactions on Information Theory* 56.5 (2010), pp. 2053–2080.

[CW15] Y. Chen and M. J. Wainwright. “Fast low-rank estimation by projected gradient descent: General statistical and algorithmic guarantees”. In: *arXiv.org* (Sept. 2015). arXiv: 1509.03025.

[CZ99] C. H. Chen and X. Zhang. “Independent component analysis for remote sensing study”. In: *Image and Signal Processing for Remote Sensing V*. Ed. by S. B. Serpico. SPIE, Dec. 1999, pp. 150–158.

[DB16] S. Diamond and S. Boyd. “CVXPY: A Python-Embedded Modeling Language for Convex Optimization”. In: *Journal of Machine Learning Research* 17.83 (2016), pp. 1–5.

[dEJL07] A. d’Aspremont, L. El Ghaoui, M. I. Jordan, and G. R. G. Lanckriet. “A Direct Formulation for Sparse PCA Using Semidefinite Programming”. In: *SIAM Review* 49.3 (Jan. 2007), pp. 434–448.

[DF87] J. C. Dainty and J. R. Feinup. “Phase retrieval and image reconstruction for astronomy”. In: *Image recovery*. Ed. by H. Stark. Orlando: Academic Press, 1987.

[DGFS08] J. Diestel, A. Grothendieck, J. H. Fourie, and J. Swart. *The Metric Theory of Tensor Products*. Grothendieck’s Résumé Revisited. Providence: American Mathematical Society, Jan. 2008.

[DHS05] C. Ding, X. He, and H. D. Simon. “On the Equivalence of Nonnegative Matrix Factorization and Spectral Clustering”. In: *Proceedings of the SIAM International Conference on Data Mining*. Philadelphia: Society for Industrial and Applied Mathematics, 2005, pp. 606–610.

[DJM13] D. L. Donoho, I. Johnstone, and A. Montanari. “Accurate Prediction of Phase Transitions in Compressed Sensing via a Connection to Minimax Denoising”. In: *IEEE Transactions on Information Theory* 59.6 (2013), pp. 3396–3433.

[DKK12] D. M. Dunlavy, T. G. Kolda, and W. P. Kegelmeyer. “Multilinear Algebra for Analyzing Data with Multiple Linkages”. In: *Graph Algorithms in the Language of Linear Algebra*. Philadelphia: Society for Industrial and Applied Mathematics, Mar. 2012, pp. 85–114.

[DMA97] G. Davis, S. Mallat, and M. Avellaneda. “Adaptive greedy approximations”. In: *Constructive Approximation* 13.1 (Mar. 1997), pp. 57–98.

[Don06] D. L. Donoho. “Compressed sensing”. In: *IEEE Transactions on Information Theory* 52.4 (Apr. 2006), pp. 1289–1306.

[Dro89] S. N. Drossos. “Fast artifact free reconstruction algorithm for limited data (PET) using constrained optimisation”. In: *Proceedings of the 3rd International Conference on Image Processing and its Applications*. 1989, pp. 367–372.

[DS89] D. L. Donoho and P. B. Stark. “Uncertainty Principles and Signal Recovery”. In: *SIAM Journal on Applied Mathematics* 49.3 (June 1989), pp. 906–931.

[DT96] R. A. DeVore and V. N. Temlyakov. “Some remarks on greedy algorithms”. In: *Advances in Computational Mathematics* 5.1 (Dec. 1996), pp. 173–187.

[EA06] M. Elad and M. Aharon. “Image Denoising Via Sparse and Redundant Representations Over Learned Dictionaries”. In: *IEEE Transactions on Image Processing* 15.12 (Dec. 2006), pp. 3736–3745.

[EAH99] K. Engan, S. O. Aase, and J. Hakon Husoy. “Method of optimal directions for frame design”. In: *Proceedings of the 1999 IEEE International Conference on Acoustics, Speech, and Signal Processing*. IEEE, 1999, 2443–2446 vol.5.

[Faz02] M. Fazel. “Matrix rank minimization with applications”. PhD thesis. Stanford University, 2002.

[Fie78] J. R. Fienup. “Reconstruction of an object from the modulus of its Fourier transform”. In: *Optics Letters* 3.1 (July 1978), pp. 27–29.

[Fie82] J. R. Fienup. “Phase retrieval algorithms: a comparison”. In: *Applied Optics* 21.15 (Aug. 1982), pp. 2758–2769.

[FKL89] K. Faulkner, C. J. Kotre, and M. Louka. “Veiling glare deconvolution of images produced by X-ray image intensifiers”. In: *Proceedings of the 3rd International Conference on Image Processing and its Applications*. 1989, pp. 669–673.

[Fli16] A. Flinth. “Sparse Blind Deconvolution and Demixing Through $\ell_{1,2}$-Minimization”. In: *arXiv.org* (Sept. 2016). arXiv: 1609.06357.

[FR08] M. Fornasier and H. Rauhut. “Recovery Algorithms for Vector-Valued Data with Joint Sparsity Constraints”. In: *SIAM Journal on Numerical Analysis* 46.2 (Feb. 2008), pp. 577–613.

[FS14] B. Friedlander and T. Strohmer. “Bilinear compressed sensing for array self-calibration”. In: *Proceedings of the 48th Asilomar Conference on Signals, Systems and Computers*. IEEE, 2014, pp. 363–367.

[FW88] B. Friedlander and A. J. Weiss. “Eigenstructure methods for direction finding with sensor gain and phase uncertainties”. In: *Proceedings of the 1988 IEEE International Conference on Acoustics, Speech, and Signal Processing*. IEEE, 1988, pp. 2681–2684.

[FW91] B. Friedlander and A. J. Weiss. “Direction finding in the presence of mutual coupling”. In: *IEEE Transactions on Antennas and Propagation* 39.3 (Mar. 1991), pp. 273–284.

[GB08] M. C. Grant and S. P. Boyd. “Graph Implementations for Nonsmooth Convex Programs”. In: *Recent Advances in Learning and Control*. London: Springer, 2008, pp. 95–110.

[GB14] M. Grant and S. Boyd. *CVX: Matlab Software for Disciplined Convex Programming*. Mar. 2014. URL: http://cvxr.com/cvx.

[GBY06] M. Grant, S. Boyd, and Y. Ye. “Disciplined Convex Programming”. In: *Global Optimization*. Boston: Kluwer Academic Publishers, 2006, pp. 155–210.

[GCD12] R. Gribonval, G. Chardon, and L. Daudet. “Blind calibration for compressed sensing by convex optimization”. In: *Proceedings of the 2012 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2012, pp. 2713–2716.

[GGR95] I. F. Gorodnitsky, J. S. George, and B. D. Rao. “Neuromagnetic source imaging with FOCUSS: A recursive weighted minimum norm algorithm”. In: *Electroencephalography and Clinical Neurophysiology* 95.4 (Nov. 1995), pp. 231–51.

[GLM16] R. Ge, J. D. Lee, and T. Ma. “Matrix Completion has No Spurious Local Minimum”. In: *Advances in Neural Information Processing Systems 29 (NIPS 2016)*. 2016, pp. 2973–2981.

[Gow+07] A. Gowen et al. “Hyperspectral imaging – an emerging process analytical tool for food quality and safety control”. In: *Trends in Food Science & Technology* 18.12 (Dec. 2007), pp. 590–598.

[Gri02] R. Gribonval. “Sparse decomposition of stereo signals with Matching Pursuit and application to blind separation of more than two sources from a stereo mixture”. In: *Proceedings of the 2002 IEEE International Conference on Acoustics, Speech, and Signal Processing*. IEEE, May 2002, pp. III–3057–III–3060.

[Gro53] A. Grothendieck. “Résumé de la théorie métrique des produits tensoriels topologiques”. In: *Bol. Soc. Mat. São Paulo* 8 (1953), pp. 1–79.

[GS72] R. W. Gerchberg and W. O. Saxton. “A practical algorithm for the determination of the phase from image and diffraction plane pictures”. In: *Optik* 35 (1972), pp. 237–246.

[GW95] M. X. Goemans and D. P. Williamson. “Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming”. In: *Journal of the ACM* 42.6 (Nov. 1995), pp. 1115–1145.

[Har14] M. Hardt. “Understanding Alternating Minimization for Matrix Completion”. In: *Proceedings of the IEEE 55th Annual Symposium on Foundations of Computer Science*. IEEE, Oct. 2014, pp. 651–660.

[Har70] R. A. Harshman. “Foundations of the PARAFAC Procedure”. In: *UCLA Working Papers in Phonetics* 16 (1970), pp. 1–84.

[Har93] R. W. Harrison. “Phase problem in crystallography”. In: *Journal of the Optical Society of America A* 10.5 (May 1993), pp. 1046–1055.

[Hau82] R. E. Hausman. “Constrained multivariate analysis”. In: *Optimisation in Statistics*. 1982, pp. 137–151.

[Heg+03] E. K. Hege et al. “Hyperspectral imaging for astronomy and space surviellance”. In: *Imaging Spectrometry IX*. Ed. by S. S. Shen and P. E. Lewis. SPIE, Aug. 2003, p. 380.

[Hit27] F. L. Hitchcock. “The Expression of a Tensor or a Polyadic as a Sum of Products”. In: *Studies in Applied Mathematics* 6.1-4 (Apr. 1927), pp. 164–189.

[Hit28] F. L. Hitchcock. “Multiple Invariants and Generalized Rank of a P-Way Matrix or Tensor”. In: *Studies in Applied Mathematics* 7.1-4 (Apr. 1928), pp. 39–79.

[HMT11] N. Halko, P. G. Martinsson, and J. A. Tropp. “Finding Structure with Randomness: Probabilistic Algorithms for Constructing Approximate Matrix Decompositions”. In: *SIAM Review* 53.2 (May 2011).

[Hoy02] P. O. Hoyer. “Non-negative sparse coding”. In: *Proceedings of the 2002 12th IEEE Workshop on Neural Networks for Signal Processing*. IEEE, Nov. 2002, pp. 557–565.

[Hoy04] P. O. Hoyer. “Non-negative Matrix Factorization with Sparseness Constraints”. In: *Journal of Machine Learning Research* 5 (Dec. 2004), pp. 1457–1469.

[HV15] B. D. Haeffele and R. Vidal. “Global Optimality in Tensor Factorization, Deep Learning, and Beyond”. In: *arXiv.org* (June 2015). arXiv: 1506.07540.

[HYV14] B. D. Haeffele, E. D. Young, and R. Vidal. “Structured low-rank matrix factorization”. In: *Proceedings of the 31st International Conference on Machine Learning*. 2014.

[HZZS16] W. He, H. Zhang, L. Zhang, and H. Shen. "Total-Variation-Regularized Low-Rank Matrix Factorization for Hyperspectral Image Restoration". In: *IEEE Transactions on Geoscience and Remote Sensing* 54.1 (Jan. 2016), pp. 178–188.

[IBP11] M.-D. Iordache, J. M. Bioucas-Dias, and A. Plaza. "Sparse Unmixing of Hyperspectral Data". In: *IEEE Transactions on Geoscience and Remote Sensing* 49.6 (June 2011), pp. 2014–2039.

[IPB10] M.-D. Iordache, A. Plaza, and J. Bioucas-Dias. "On the use of spectral libraries to perform sparse unmixing of hyperspectral data". In: *Proceedings of the 2nd Workshop on Hyperspectral Image and Signal Processing: Evolution in Remote Sensing* (2010), pp. 1–4.

[Jac91] J. E. Jackson. *A User's Guide to Principal Components*. Hoboken: John Wiley & Sons, Inc., 1991.

[Jam87] G. J. O. Jameson. *Summing and Nuclear Norms in Banach Space Theory*. Cambridge: Cambridge University Press, July 1987.

[JC93] S. M. Jefferies and J. C. Christou. "Restoration of Astronomical Images by Iterative Blind Deconvolution". In: *The Astrophysical Journal* 415 (Sept. 1993), p. 862.

[Jef67] J. N. R. Jeffers. "Two Case Studies in the Application of Principal Component Analysis". In: *Applied Statistics* 16.3 (1967), p. 225.

[JNS12] P. Jain, P. Netrapalli, and S. Sanghavi. "Low-rank Matrix Completion using Alternating Minimization". In: *arXiv.org* (Dec. 2012). arXiv: 1212.0467.

[Jol95] I. T. Jolliffe. "Rotation of principal components: choice of normalization constraints". In: *Journal of Applied Statistics* 22.1 (1995), pp. 29–35.

[JTU03] I. T. Jolliffe, N. T. Trendafilov, and M. Uddin. "A Modified Principal Component Technique Based on the LASSO". In: *Journal of Computational and Graphical Statistics* 12.3 (Sept. 2003), pp. 531–547.

[KB09] T. G. Kolda and B. W. Bader. "Tensor Decompositions and Applications". In: *SIAM Review* 51.3 (Aug. 2009), pp. 455–500.

[KBV09] Y. Koren, R. Bell, and C. Volinsky. "Matrix Factorization Techniques for Recommender Systems". In: *Computer* 42.8 (Aug. 2009), pp. 30–37.

[KCM01] M. S. Kim, Y. R. Chen, and P. M. Mehl. "Hyperspectral reflectance and fluorescence imaging system for food quality and safety". In: *Transactions of the American Society of Agricultural Engineers* 44.3 (May 2001), pp. 721–729.

[KM02] N. Keshava and J. F. Mustard. “Spectral unmixing”. In: *IEEE Signal Processing Magazine* 19.1 (2002), pp. 44–57.

[KMO10a] R. H. Keshavan, A. Montanari, and S. Oh. “Matrix Completion From a Few Entries”. In: *IEEE Transactions on Information Theory* 56.6 (May 2010), pp. 2980–2998.

[KMO10b] R. H. Keshavan, A. Montanari, and S. Oh. “Matrix Completion from Noisy Entries”. In: *Journal of Machine Learning Research* 11.Jul (2010), pp. 2057–2078.

[KN12] S. Khot and A. Naor. “Grothendieck-Type Inequalities in Combinatorial Optimization”. In: *Communications on Pure and Applied Mathematics* 65.7 (July 2012), pp. 992–1035.

[KO00] T. G. Kolda and D. P. O’Leary. “Algorithm 805: computation and uses of the semidiscrete matrix decomposition”. In: *ACM Transactions on Mathematical Software* 26.3 (Sept. 2000), pp. 415–435.

[Kor09] Y. Koren. *The BellKor Solution to the Netflix Grand Prize*. Tech. rep. 2009.

[KP07] H. Kim and H. Park. “Sparse non-negative matrix factorizations via alternating non-negativity-constrained least squares for microarray data analysis”. In: *Bioinformatics* 23.12 (June 2007), pp. 1495–1502.

[KP08] J. Kim and H. Park. *Sparse Nonnegative Matrix Factorization for Clustering*. Tech. rep. 36. Georgia Institute of Technology, 2008.

[Kri+92] V. Krishnamurthi et al. “Blind deconvolution of 2-D and 3-D fluorescent micrographs”. In: *Biomedical Image Processing and Three-Dimensional Microscopy*. Ed. by R. S. Acharya, C. J. Cogswell, and D. B. Goldgof. San Jose: SPIE, June 1992, pp. 95–102.

[Kri77] J.-L. Krivine. “Sur la constante de Grothendieck”. In: *CR Acad. Sci. Paris Ser. AB* 284.8 (1977), A445–A446.

[LB08] D. Letexier and S. Bourennane. “Noise Removal From Hyperspectral Images by Multidimensional Filtering”. In: *IEEE Transactions on Geoscience and Remote Sensing* 46.7 (July 2008), pp. 2061–2069.

[LB13] T. Lin and S. Bourennane. “Survey of hyperspectral image denoising methods based on tensor decompositions”. In: *EURASIP Journal on Advances in Signal Processing* 2013.1 (2013), pp. 1–11.

[LB14] J. Lipor and L. Balzano. “Robust blind calibration via total least squares”. In: *Proceedings of the 2014 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2014, pp. 4244–4248.

[LBF12] X. Liu, S. Bourennane, and C. Fossati. “Denoising of Hyperspectral Images Using the PARAFAC Model and Statistical Performance Analysis”. In: *IEEE Transactions on Geoscience and Remote Sensing* 50.10 (Oct. 2012), pp. 3717–3724.

[Lee+10] J. D. Lee et al. “Practical Large-Scale Optimization for Max-norm Regularization”. In: *Advances Neural Information Processing Systems 23 (NIPS 2010)*. 2010, pp. 1297–1305.

[LGLS06] Z.-Q. Luo, M. Gastpar, J. Liu, and A. Swami. “Distributed signal processing in sensor networks”. In: *IEEE Signal Processing Magazine* 23.4 (July 2006), pp. 14–15.

[Li+15] C. Li et al. “Hyperspectral image denoising using the robust low-rank tensor recovery”. In: *Journal of the Optical Society of America A* 32.9 (Sept. 2015), pp. 1604–1612.

[Lin07] C.-J. Lin. “Projected Gradient Methods for Nonnegative Matrix Factorization”. In: *Neural Computation* 19.10 (Oct. 2007), pp. 2756–2779.

[Liu+11] A. Liu et al. “An Eigenstructure Method for Estimating DOA and Sensor Gain-Phase Errors”. In: *IEEE Transactions on Signal Processing* 59.12 (2011), pp. 5944–5956.

[LLJB15] K. Lee, Y. Li, M. Junge, and Y. Bresler. “Stability in blind deconvolution of sparse signals and reconstruction by alternating minimization”. In: *Proceedings of the 2015 International Conference on Sampling Theory and Applications*. IEEE, 2015, pp. 158–162.

[LLR16] Y. Li, Y. Liang, and A. Risteski. “Recovery Guarantee of Nonnegative Matrix Factorization via Alternating Updates”. In: *Advances in Neural Information Processing Systems 29 (NIPS 2016)*. Nov. 2016.

[LLSW16] X. Li, S. Ling, T. Strohmer, and K. Wei. “Rapid, Robust, and Reliable Blind Deconvolution via Nonconvex Optimization”. In: *arXiv.org* (June 2016). arXiv: 1606.04933.

[Lov79] L. Lovász. “On the Shannon capacity of a graph”. In: *IEEE Transactions on Information Theory* 25.1 (Jan. 1979), pp. 1–7.

[LP68] J. Lindenstrauss and A. Pełczyński. “Absolutely summing operators in L_p-spaces and their applications”. In: *Studia Mathematica* 29.3 (1968), pp. 275–326.

[LS01] D. D. Lee and H. S. Seung. “Algorithms for Non-negative Matrix Factorization”. In: *Advances in Neural Information Processing Systems 13 (NIPS 2000)*. 2001, pp. 556–562.

[LS15a] S. Ling and T. Strohmer. “Blind Deconvolution Meets Blind Demixing: Algorithms and Performance Bounds”. In: *arXiv.org* (Dec. 2015). arXiv: 1512.07730.

[LS15b] S. Ling and T. Strohmer. “Self-calibration and biconvex compressive sensing”. In: *Inverse Problems* 31.11 (Sept. 2015), p. 115002.

[LS16] S. Ling and T. Strohmer. “Self-Calibration via Linear Least Squares”. In: *arXiv.org* (Nov. 2016). arXiv: 1611.04196.

[LS97] D. D. Lee and H. S. Seung. “Unsupervised Learning by Convex and Conic Coding”. In: *Advances in Neural Information Processing Systems 9 (NIPS 1996)*. 1997, pp. 515–521.

[LS99] D. D. Lee and H. S. Seung. “Learning the parts of objects by non-negative matrix factorization”. In: *Nature* 401.6755 (Oct. 1999), pp. 788–791.

[LSJR16] J. D. Lee, M. Simchowitz, M. I. Jordan, and B. Recht. “Gradient Descent Only Converges to Minimizers”. In: *Proceedings of the 29th Conference on Learning Theory*. 2016, pp. 1246–1257.

[LY06] M. Lin and L. Yang. “Blind Calibration and DOA Estimation With Uniform Circular Arrays in the Presence of Mutual Coupling”. In: *Antennas and Wireless Propagation Letters* 5.1 (2006), pp. 315–318.

[Mai+08] J. Mairal et al. “Discriminative Sparse Image Models for Class-Specific Edge Detection and Image Interpretation”. In: *Computer Vision – ECCV 2008*. Berlin, Heidelberg: Springer, Oct. 2008, pp. 43–56.

[Mal09] S. Mallat. *A Wavelet Tour of Signal Processing*. 3rd edition. The Sparse Way. Burlington: Academic Press, 2009.

[McC13] M. B. McCoy. “A geometric analysis of convex demixing”. PhD thesis. California Institute of Technology, 2013.

[MCKS99] J. Miao, P. Charalambous, J. Kirz, and D. Sayre. “Extending the methodology of X-ray crystallography to allow imaging of micrometre-sized non-crystalline specimens”. In: *Nature* 400.6742 (July 1999), pp. 342–344.

[MCW03] D. M. Malioutov, M. Cetin, and A. S. Willsky. “Source localization by enforcing sparsity through a laplacian prior: an SVD-based approach”. In: *Proceedings of the 2003 IEEE Workshop on Statistical Signal Processing*. IEEE, 2003, pp. 573–576.

[MCW05] D. Malioutov, M. Cetin, and A. S. Willsky. “A sparse signal reconstruction perspective for source localization with sensor arrays”. In: *IEEE Transactions on Signal Processing* 53.8 (July 2005), pp. 3010–3022.

[MDO11] R. Mignot, L. Daudet, and F. Ollivier. “Compressed sensing for acoustic response reconstruction: Interpolation of the early part”. In: *Proceedings of the 2011 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*. IEEE, Oct. 2011, pp. 225–228.

[Mez07] F. Mezzadri. “How to generate random matrices from the classical compact groups”. In: *Notices of the American Mathematical Society* 54.5 (2007), pp. 592–604.

[Mia+02] J. Miao et al. “High Resolution 3D X-Ray Diffraction Microscopy”. In: *Physical Review Letters* 89.8 (Aug. 2002), p. 088303.

[Mil90] R. P. Millane. “Phase retrieval in crystallography and optics”. In: *Journal of the Optical Society of America A* 7.3 (Mar. 1990), pp. 394–411.

[MISE08] J. Miao, T. Ishikawa, Q. Shen, and T. Earnest. “Extending X-Ray Crystallography to Allow the Imaging of Noncrystalline Materials, Cells, and Single Protein Complexes”. In: *Annual Review of Physical Chemistry* 59.1 (May 2008), pp. 387–410.

[MN36] F. J. Murray and J. von Neumann. “On Rings of Operators”. In: *Annals of Mathematics* 37.1 (Jan. 1936), p. 116.

[MT11] M. McCoy and J. A. Tropp. “Two proposals for robust PCA using semidefinite programming”. In: *Electronic Journal of Statistics* 5 (2011), pp. 1123–1160.

[Nes04] Y. Nesterov. *Introductory lectures on convex optimization*. Vol. 87. Applied Optimization. Boston: Kluwer Academic Publishers, 2004.

[Nes05] Y. Nesterov. “Smooth minimization of non-smooth functions”. In: *Mathematical Programming* 103.1 (2005), pp. 127–152.

[Nes07] Y. Nesterov. *Gradient Methods for Minimizing Composite Objective Function*. Tech. rep. 2007/76. Louvain-la-Neuve, Belgium: CORE, Université catholique de Louvain, 2007.

[Nes98] Y. Nesterov. “Semidefinite relaxation and nonconvex quadratic optimization”. In: *Optimization Methods and Software* 9.1-3 (Jan. 1998), pp. 141–160.

[NJS15] P. Netrapalli, P. Jain, and S. Sanghavi. “Phase Retrieval Using Alternating Minimization”. In: *IEEE Transactions on Signal Processing* 63.18 (Sept. 2015), pp. 4814–4826.

[NS96] B. C. Ng and C. M. S. See. “Sensor-array calibration using a maximum-likelihood approach”. In: *IEEE Transactions on Antennas and Propagation* 44.6 (June 1996), pp. 827–835.

[NW14] A. Natarajan and Y. Wu. "Computational Complexity of Certifying Restricted Isometry Property". In: *Leibniz International Proceedings in Informatics*. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2014, p. 380.

[OCPB16] B. O'Donoghue, E. Chu, N. Parikh, and S. Boyd. "Conic Optimization via Operator Splitting and Homogeneous Self-Dual Embedding". In: *Journal of Optimization Theory and Applications* 169.3 (Feb. 2016), pp. 1042–1068.

[OF97] B. A. Olshausen and D. J. Field. "Sparse coding with an overcomplete basis set: A strategy employed by V1?" In: *Vision Research* 37.23 (Dec. 1997), pp. 3311–3325.

[OH15] S. Oymak and B. Hassibi. "Sharp MSE Bounds for Proximal Denoising". In: *Foundations of Computational Mathematics* 16.4 (Oct. 2015), pp. 965–1029.

[Old34] R. Oldenburger. "Composition and Rank of n-Way Matrices and Multilinear Forms". In: *Annals of Mathematics* 35.3 (July 1934), p. 622.

[Old36] R. Oldenburger. "Non-Singular Multilinear Forms and Certain p-Way Matrix Factorizations". In: *Transactions of the American Mathematical Society* 39.3 (May 1936), p. 422.

[Oll15] E. Ollila. "Nonparametric simultaneous sparse recovery: An application to source localization". In: *Proceedings of the 23rd European Signal Processing Conference*. IEEE, 2015, pp. 509–513.

[Pad94] J. F. Padgett. *Marriage and Elite Structure in Renaissance Florence, 1281–1500*. Oct. 1994. URL: https://uchicago.app.box.com/s/izitmyicp0vjj4ajxomembzn8p0xot5s.

[Ped+11] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12.Oct (2011), pp. 2825–2830.

[Phi+05] C. Phillips et al. "An empirical Bayesian solution to the source reconstruction problem in EEG". In: *NeuroImage* 24.4 (Feb. 2005), pp. 997–1011.

[Pie07] A. Pietsch. *History of Banach Spaces and Linear Operators*. Basel: Birkhäuser, 2007.

[Pis12] G. Pisier. "Grothendieck's Theorem, past and present". In: *Bulletin of the American Mathematical Society* 49.2 (2012), pp. 237–323.

[Pis86] G. Pisier. *Factorization of Linear Operators and Geometry of Banach Spaces*. Vol. 60. Washington: Conference Board of the Mathematical Sciences, 1986.

[PRK93] Y. C. Pati, R. Rezaiifar, and P. S. Krishnaprasad. “Orthogonal matching pursuit: recursive function approximation with applications to wavelet decomposition”. In: *Proceedings of the 27th Asilomar Conference on Signals, Systems and Computers*. IEEE Comput. Soc. Press, Nov. 1993, pp. 40–44.

[PT94] P. Paatero and U. Tapper. “Positive matrix factorization: A nonnegative factor model with optimal utilization of error estimates of data values”. In: *Environmetrics* 5.2 (June 1994), pp. 111–126.

[RBB08] N. Renard, S. Bourennane, and J. Blanc-Talon. “Denoising and Dimensionality Reduction Using Multilinear Tools for Hyperspectral Images”. In: *IEEE Geoscience and Remote Sensing Letters* 5.2 (Apr. 2008), pp. 138–142.

[Rec11] B. Recht. “A Simpler Approach to Matrix Completion”. In: *Journal of Machine Learning Research* 12.Dec (2011), pp. 3413–3430.

[RFP10] B. Recht, M. Fazel, and P. A. Parrilo. “Guaranteed Minimum-Rank Solutions of Linear Matrix Equations via Nuclear Norm Minimization”. In: *SIAM Review* 52.3 (2010), pp. 471–501.

[RGA77] L. C. Rowan, A. F. H. Goetz, and R. P. Ashley. “Discrimination of Hydrothermally Altered and Unaltered Rocks in Visible and Near-infrared Multispectral Images”. In: *Geophysics* 42.3 (Apr. 1977), pp. 522–535.

[Rod+05] O. Y. Rodionova et al. “NIR spectrometry for counterfeit drug detection”. In: *Analytica Chimica Acta* 549.1–2 (Sept. 2005), pp. 151–158.

[ROF92] L. I. Rudin, S. Osher, and E. Fatemi. “Nonlinear total variation based noise removal algorithms”. In: *Physica D: Nonlinear Phenomena* 60.1-4 (Nov. 1992), pp. 259–268.

[Row+74] L. C. Rowan et al. *Discrimination of rock types and detection of hydrothermally altered areas in south-central Nevada by the use of computer-enhanced ERTS images*. Tech. rep. 1974.

[RR13] B. Recht and C. Re. “Parallel stochastic gradient algorithms for large-scale matrix completion”. In: *Mathematical Programming Computation* 5.2 (Apr. 2013), pp. 201–226.

[RRTB12] B. Recht, C. Re, J. A. Tropp, and V. Bittorf. “Factoring nonnegative matrices with linear programs”. In: *Advances in Neural Information Processing Systems 25 (NIPS 2012)*. 2012, pp. 1214–1222.

[RRWN11] B. Recht, C. Re, S. Wright, and F. Niu. “Hogwild: A Lock-Free Approach to Parallelizing Stochastic Gradient Descent”. In: *Advances in Neural Information Processing Systems 24 (NIPS 2011)*. 2011, pp. 693–701.

[RRZF06] D. M. Rogge, B. Rivard, J. Zhang, and J. Feng. “Iterative Spectral Unmixing for Optimizing Per-Pixel Endmember Sets”. In: *IEEE Transactions on Geoscience and Remote Sensing* 44.12 (2006), pp. 3725–3736.

[Rya02] R. A. Ryan. *Introduction to Tensor Products of Banach Spaces*. London: Springer-Verlag, 2002.

[Sch43] R. Schatten. “On the direct product of Banach spaces”. In: *Transactions of the American Mathematical Society* 53.2 (1943), pp. 195–217.

[Sch46] R. Schatten. “The Cross-Space of Linear Transformations”. In: *Annals of Mathematics* 47.1 (Jan. 1946), pp. 73–84.

[Sch50] R. Schatten. *A theory of cross-spaces*. Princeton: Princeton University Press, 1950.

[SDB13] S. Sahnoun, E.-H. Djermoune, and D. Brie. “Sparse modal estimation of 2-D NMR signals”. In: *Proceedings of the 2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, May 2013, pp. 8751–8755.

[SDBC17] S. Sahnoun, E.-H. Djermoune, D. Brie, and P. Comon. “A Simultaneous Sparse Approximation Method for Multidimensional Harmonic Retrieval”. In: *Signal Processing* 131 (Feb. 2017), pp. 36–48.

[See94] C. M. S. See. “Sensor array calibration in the presence of mutual coupling and unknown sensor gains and phases”. In: *Electronics Letters* 30.5 (Mar. 1994), pp. 373–374.

[SH08] H. Shen and J. Z. Huang. “Sparse principal component analysis via regularized low rank matrix approximation”. In: *Journal of Multivariate Analysis* 99.6 (July 2008), pp. 1015–1034.

[She+16] X. Shen et al. “Disciplined Multi-Convex Programming”. In: *arXiv.org* (Sept. 2016). arXiv: 1609.03285.

[SL15] R. Sun and Z.-Q. Luo. “Guaranteed Matrix Completion via Non-convex Factorization”. In: *Proceedings of the IEEE 56th Annual Symposium on Foundations of Computer Science*. IEEE, Dec. 2015, pp. 270–289.

[SN46] R. Schatten and J. von Neumann. “The Cross-Space of Linear Transformations. II”. In: *Annals of Mathematics* 47.3 (July 1946), pp. 608–630.

[SN48] R. Schatten and J. von Neumann. “The Cross-Space of Linear Transformations. III”. In: *Annals of Mathematics* 49.3 (July 1948), pp. 557–582.

[SQW16] J. Sun, Q. Qu, and J. Wright. "A geometric analysis of phase retrieval". In: *Proceedings of the 2016 IEEE International Symposium on Information Theory*. IEEE, July 2016, pp. 2379–2383.

[SQW17a] J. Sun, Q. Qu, and J. Wright. "Complete Dictionary Recovery over the Sphere I: Overview and the Geometric Picture". In: *IEEE Transactions on Information Theory* 63.2 (2017), pp. 853–884.

[SQW17b] J. Sun, Q. Qu, and J. Wright. "Complete Dictionary Recovery over the Sphere II: Recovery by Riemannian Trust-region Method". In: *IEEE Transactions on Information Theory* 63.2 (2017), pp. 885–914.

[SRJ05] N. Srebro, J. Rennie, and T. S. Jaakkola. "Maximum-Margin Matrix Factorization". In: *Advances in Neural Information Processing Systems 17 (NIPS 2004)*. 2005, pp. 1329–1336.

[SS05] N. Srebro and A. Shraibman. "Rank, Trace-Norm and Max-Norm". In: *Learning Theory*. Berlin, Heidelberg: Springer, June 2005, pp. 545–560.

[TGS06] J. A. Tropp, A. C. Gilbert, and M. J. Strauss. "Algorithms for simultaneous sparse approximation. Part I: Greedy pursuit". In: *Signal Processing* 86.3 (Mar. 2006), pp. 572–588.

[Tib96] R. Tibshirani. "Regression shrinkage and selection via the Lasso". In: *Journal of the Royal Statistical Society: Series B (Methodology)* 58.1 (1996), pp. 267–288.

[TP14] A. M. Tillmann and M. E. Pfetsch. "The Computational Complexity of the Restricted Isometry Property, the Nullspace Property, and Related Concepts in Compressed Sensing". In: *IEEE Transactions on Information Theory* 60.2 (Feb. 2014), pp. 1248–1259.

[Tro03] J. A. Tropp. *Literature Survey: Non-Negative Matrix Factorization*. 2003. URL: http://users.cms.caltech.edu/~jtropp/notes/Tro03-Literature-Survey.pdf.

[Tro04] J. A. Tropp. "Topics in sparse approximation". PhD thesis. 2004.

[Tro06] J. A. Tropp. "Algorithms for simultaneous sparse approximation. Part II: Convex relaxation". In: *Signal Processing* 86.3 (Mar. 2006), pp. 589–602.

[Tro12] J. A. Tropp. "The nuclear option". Oct. 2012.

[Tu00] T.-M. Tu. "Unsupervised signature extraction and separation in hyperspectral images: a noise-adjusted fast independent component analysis approach". In: *Optical Engineering* 39.4 (Apr. 2000), pp. 897–906.

[TXHK95] L. Tong, G. Xu, B. Hassibi, and T. Kailath. “Blind Channel Identification Based on Second-Order Statistics: A Frequency-Domain Approach”. In: *IEEE Transactions on Information Theory* 41.1 (1995), pp. 329–334.

[Ude+14] M. Udell et al. “Convex optimization in Julia”. In: *Proceedings of the First Workshop for High Performance Technical Computing in Dynamic Languages (HPTCDL)*. IEEE Press, Nov. 2014, pp. 18–28.

[Van00] C. F. Van Loan. “The ubiquitous Kronecker product”. In: *Journal of Computational and Applied Mathematics* 123.1-2 (2000), pp. 85–100.

[VG88] G. Vane and A. F. H. Goetz. “Terrestrial Imaging Spectroscopy”. In: *Remote Sensing of Environment* 24.1 (Feb. 1988), pp. 1–29.

[VG93] G. Vane and A. F. H. Goetz. “Terrestrial imaging spectrometry: Current status, future trends”. In: *Remote Sensing of Environment* 44.2-3 (May 1993), pp. 117–126.

[Vin00] S. K. Vines. “Simple principal components”. In: *Journal of the Royal Statistical Society: Series C (Applied Statistics)* 49.4 (Jan. 2000), pp. 441–451.

[Wal63] A. Walther. “The Question of Phase Retrieval in Optics”. In: *Optica Acta: International Journal of Optics* 10.1 (Jan. 1963), pp. 41–49.

[Wan16] L. Wang. “Blind Deconvolution from Multiple Sparse Inputs”. In: *IEEE Signal Processing Letters* 23.10 (Oct. 2016), pp. 1384–1388.

[Wat11] J. Watrous. *CS 766/QIC 820 Theory of Quantum Information*. 2011. URL: https://cs.uwaterloo.ca/~watrous/CS766/LectureNotes/all.pdf.

[WBSJ15] G. Wunder, H. Boche, T. Strohmer, and P. Jung. “Sparse Signal Processing Concepts for Efficient 5G System Design”. In: *IEEE Access* 3 (2015), pp. 195–208.

[WH90] T. Wilson and S. J. Hewlett. “Imaging strategies in three-dimensional confocal microscopy”. In: *Biomedical Image Processing*. Ed. by A. C. Bovik and W. E. Higgins. Santa Clara: International Society for Optics and Photonics, May 1990, pp. 35–45.

[WP98] X. Wang and H. V. Poor. “Blind equalization and multiuser detection in dispersive CDMA channels”. In: *IEEE Transactions on Communications* 46.1 (1998), pp. 91–103.

[WRS08] C. Wang, P. Ramanathan, and K. K. Saluja. “Calibrating Nonlinear Mobile Sensors”. In: *Proceedings of the 5th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks*. IEEE, June 2008, pp. 533–541.

[WTH09] D. M. Witten, R. Tibshirani, and T. Hastie. “A penalized matrix decomposition, with applications to sparse principal components and canonical correlation analysis.” In: *Biostatistics* 10.3 (July 2009), pp. 515–534.

[XCS12] H. Xu, C. Caramanis, and S. Sanghavi. “Robust PCA via Outlier Pursuit”. In: *IEEE Transactions on Information Theory* 58.5 (May 2012), pp. 3047–3064.

[XY13] Y. Xu and W. Yin. “A Block Coordinate Descent Method for Regularized Multiconvex Optimization with Applications to Nonnegative Tensor Factorization and Completion”. In: *SIAM Journal on Imaging Sciences* 6.3 (July 2013), pp. 1758–1789.

[YWHM10] J. Yang, J. Wright, T. S. Huang, and Y. Ma. “Image Super-Resolution Via Sparse Representation”. In: *IEEE Transactions on Image Processing* 19.11 (May 2010), pp. 2861–2873.

[ZG06] A. Zelinski and V. Goyal. “Denoising Hyperspectral Imagery and Recovering Junk Bands using Wavelets and Sparse Approximation”. In: *Proceedings of the 2006 IEEE International Symposium on Geoscience and Remote Sensing* (2006), pp. 387–390.

[ZH05] H. Zou and T. Hastie. “Regularization and variable selection via the elastic net”. In: *Journal of the Royal Statistical Society: Series B (Methodology)* 67 (2005), pp. 301–320.

[Zha+02] H. Zha et al. “Spectral Relaxation for K-means Clustering”. In: *Advances in Neural Information Processing Systems 14 (NIPS 2001)*. 2002, pp. 1057–1064.

[Zha+14] H. Zhang et al. “Hyperspectral Image Restoration Using Low-Rank Matrix Recovery”. In: *IEEE Transactions on Geoscience and Remote Sensing* 52.8 (Aug. 2014), pp. 4729–4743.

[ZHT06] H. Zou, T. Hastie, and R. Tibshirani. “Sparse Principal Component Analysis”. In: *Journal of Computational and Graphical Statistics* 15.2 (June 2006), pp. 265–286.

[ZL15] Q. Zheng and J. Lafferty. “A Convergent Gradient Descent Algorithm for Rank Minimization and Semidefinite Programming from Random Linear Measurements”. In: *Advances in Neural Information Processing Systems 28 (NIPS 2015)*. Montreal, June 2015, pp. 109–117.

[ZLS09] M. Zinkevich, J. Langford, and A. J. Smola. “Slow Learners are Fast”. In: *Advances in Neural Information Processing Systems 22 (NIPS 2009)*. 2009, pp. 2331–2339.

[ZWBY14] L. Zhao, L. Wang, G. Bi, and L. Yang. “An Autofocus Technique for High-Resolution Inverse Synthetic Aperture Radar Imagery”. In: *IEEE Transactions on Geoscience and Remote Sensing* 52.10 (Jan. 2014), pp. 6392–6403.

[ZWL15] T. Zhao, Z. Wang, and H. Liu. “A Nonconvex Optimization Framework for Low Rank Matrix Estimation”. In: *Advances in Neural Information Processing Systems 28 (NIPS 2015)*. 2015, pp. 559–567.

[ZWLS10] M. Zinkevich, M. Weimer, L. Li, and A. J. Smola. “Parallelized Stochastic Gradient Descent”. In: *Advances Neural Information Processing Systems 23 (NIPS 2010)*. 2010, pp. 2595–2603.

[ZWSP08] Y. Zhou, D. Wilkinson, R. Schreiber, and R. Pan. “Large-Scale Parallel Collaborative Filtering for the Netflix Prize”. In: *Algorithmic Aspects in Information and Management*. Ed. by R. Fleischer and J. Xu. Berlin, Heidelberg: Springer, 2008, pp. 337–348.

[ZY15] Y.-Q. Zhao and J. Yang. “Hyperspectral Image Denoising via Sparse Representation and Low-Rank Constraint”. In: *IEEE Transactions on Geoscience and Remote Sensing* 53.1 (Jan. 2015), pp. 296–308.

Appendix A

Proofs of results in Chapter 3

This appendix provides the proofs of some results regarding nuclear norms from Chapter 3.

A.1 Proof of Proposition 3.3.3

In this section we restate and prove Proposition 3.3.3 regarding properties of the nuclear norm.

Proposition A.1.1 (Properties of the nuclear norm)**.** *Let* $X = (\mathbb{M}^{m\times n}, \|\cdot\|_X)$ *and* $Y = (\mathbb{M}^{p\times q}, \|\cdot\|_Y)$ *be normed vector spaces. The nuclear norm* $N_{X,Y}$ *satisfies the following.*

1. *The nuclear norm* $N_{X,Y}$ *is a norm on* $\mathbb{O}^{m\times n\otimes p\times q}$.
2. $N_{X,Y}$ *is a crossnorm.*
3. $N_{X,Y}$ *dominates all other crossnorms on* $X \otimes Y$ *uniformly.*

Proof. (1) We prove that the nuclear norm $N_{X,Y}$ is in fact a norm through a geometric argument. Recall that we denote the set of dyads with unit-norm factors (3.12) as $D_{X\otimes Y}$, and we call its absolutely convex hull $S_{X\otimes Y}$ the nuclear unit ball (3.13). For this argument, we endow the factor spaces ($\mathbb{M}^{m\times n}$ and $\mathbb{M}^{p\times q}$) and the operator space ($\mathbb{O}^{m\times n\otimes p\times q}$) with Euclidean topology. We observe that:

- The set $S_{X\otimes Y}$ is *symmetric* and *convex* since it is an absolutely convex hull.

- The set $S_{X\otimes Y}$ is *closed* and *bounded*. To see this, we first note that the sets $\{\boldsymbol{X} : \|\boldsymbol{X}\|_X = 1\} \subseteq \mathbb{R}^{m\times n}$ and $\{\boldsymbol{Y} : \|\boldsymbol{Y}\|_Y = 1\} \subseteq \mathbb{R}^{p\times q}$ are compact. Therefore the set $D_{X\otimes Y}$ is also compact as it is a continuous image of the direct product of these two compact sets. Finally, $S_{X\otimes Y}$ is the convex hull of a compact set ($D_{X\otimes Y}$) in Euclidean space, and so we conclude that it is compact as well.

- The set $S_{X\otimes Y}$ is *absorbing*. We can write every operator $\mathcal{A} \in \mathbb{O}^{m\times n\otimes p\times q}$ as a linear combination $\mathcal{A} = \sum_{i=1}^{r} \lambda_i \boldsymbol{X}_i \otimes \boldsymbol{Y}_i$ with the $\boldsymbol{X}_i \otimes \boldsymbol{Y}_i \in D_{X\otimes Y}$. Thus $\mathcal{A} \in t \cdot S_{X\otimes Y}$ whenever $t \sum_{i=1}^{r} |\lambda_i| \leq 1$.

Therefore the nuclear unit ball $S_{X\otimes Y}$ satisfies the topological requirements to be the closed unit ball of a norm. Proposition 3.3.6 shows that the nuclear unit ball coincides with $\{\mathcal{A} : N_{X,Y}(\mathcal{A}) \leq 1\}$, and we conclude that the nuclear norm $N_{X,Y}$ is indeed a norm.

(2) To show that $N_{X,Y}$ is additionally a crossnorm, we must demonstrate that

$$N_{X,Y}(\boldsymbol{X} \otimes \boldsymbol{Y}) = \|\boldsymbol{X}\|_X \|\boldsymbol{Y}\|_Y,$$

for all dyads $\boldsymbol{X} \otimes \boldsymbol{Y}$. By the definition (3.10) of the nuclear norm $N_{X,Y}$, it is clear that $N_{X,Y}(\boldsymbol{X} \otimes \boldsymbol{Y}) \leq \|\boldsymbol{X}\|_X \|\boldsymbol{Y}\|_Y$.

Now we introduce the function

$$f(\mathcal{A}) := \max\{\langle \mathcal{A}, \boldsymbol{X}' \otimes \boldsymbol{Y}' \rangle : \|\boldsymbol{X}'\|_{X^*} \leq 1, \|\boldsymbol{Y}'\|_{Y^*} \leq 1\},$$

where $\|\cdot\|_{X^*}$ and $\|\cdot\|_{Y^*}$ are the dual norms of the X and Y norms. This is the maximum of a linear functional over a compact set, and therefore the value is indeed attained.

Observe, however, that we can bound the absolute value of the linear functional in the objective using the identity (3.8) that makes explicit the connection between operators and bilinear forms. That is, for $\mathcal{A} = \sum_i \boldsymbol{X}_i \otimes \boldsymbol{Y}_i$,

$$\begin{aligned}
|\langle \mathcal{A}, \boldsymbol{X}' \otimes \boldsymbol{Y}' \rangle| &= \left| \left\langle \sum_i \boldsymbol{X}_i \otimes \boldsymbol{Y}_i, \boldsymbol{X}' \otimes \boldsymbol{Y}' \right\rangle \right| \\
&= \left| \sum_i \langle \boldsymbol{X}_i, \boldsymbol{X}' \rangle \langle \boldsymbol{Y}_i, \boldsymbol{Y}' \rangle \right| \\
&\leq \sum_i |\langle \boldsymbol{X}_i, \boldsymbol{X}' \rangle| |\langle \boldsymbol{Y}_i, \boldsymbol{Y}' \rangle| \\
&\leq \|\boldsymbol{X}'\|_{X^*} \|\boldsymbol{Y}'\|_{Y^*} \sum_i \|\boldsymbol{X}_i\|_X \|\boldsymbol{Y}_i\|_Y.
\end{aligned}$$

Therefore, $f(\sum_i \boldsymbol{X}_i \otimes \boldsymbol{Y}_i) \leq \sum_i \|\boldsymbol{X}_i\|_X \|\boldsymbol{Y}_i\|_Y$, and we conclude that $f(\mathcal{A}) \leq N_{X,Y}(\mathcal{A})$.

Furthermore, we see that

$$\begin{aligned} f(\boldsymbol{X} \otimes \boldsymbol{Y}) &= \max\{\langle \boldsymbol{X} \otimes \boldsymbol{Y}, \boldsymbol{X}' \otimes \boldsymbol{Y}' \rangle : \|\boldsymbol{X}'\|_{X^*} \leq 1, \|\boldsymbol{Y}'\|_{Y^*} \leq 1\} \\ &= \max\{\langle \boldsymbol{X}, \boldsymbol{X}' \rangle \langle \boldsymbol{Y}, \boldsymbol{Y}' \rangle : \|\boldsymbol{X}'\|_{X^*} \leq 1, \|\boldsymbol{Y}'\|_{Y^*} \leq 1\} \\ &= \|\boldsymbol{X}\|_X \|\boldsymbol{Y}\|_Y. \end{aligned}$$

We conclude that

$$f(\boldsymbol{X} \otimes \boldsymbol{Y}) = \|\boldsymbol{X}\|_X \|\boldsymbol{Y}\|_Y \leq N_{X,Y}(\boldsymbol{X} \otimes \boldsymbol{Y}) \leq \|\boldsymbol{X}\|_X \|\boldsymbol{Y}\|_Y,$$

and so $N_{X,Y}(\boldsymbol{X} \otimes \boldsymbol{Y}) = \|\boldsymbol{X}\|_X \|\boldsymbol{Y}\|$. The nuclear norm $N_{X,Y}$ is a crossnorm.

(3) Finally, let $|||\cdot|||$ be any crossnorm on $\mathbb{O}^{m\times n\otimes p\times q}$. For any operator $\mathcal{A} = \sum_i \boldsymbol{X}_i \otimes \boldsymbol{Y}_i \in \mathbb{O}^{m\times n\otimes p\times q}$, we have that

$$|||\mathcal{A}||| = \left|\left|\left| \sum_i \boldsymbol{X}_i \otimes \boldsymbol{Y}_i \right|\right|\right| \leq \sum_i \|\boldsymbol{X}_i\|_X \|\boldsymbol{Y}_i\|_Y.$$

Therefore $|||\mathcal{A}||| \leq N_{X,Y}(\mathcal{A})$. □

A.2 Proof of Proposition 3.3.4

In this section we restate and prove Proposition 3.3.4 concerning the optimal decompositions in Definition 3.3.2 of the nuclear norm.

Proposition A.2.1 (Optimal decompositions)**.** *Let $X = (\mathbb{M}^{m\times n}, \|\cdot\|_X)$ and $Y = (\mathbb{M}^{p\times q}, \|\cdot\|_Y)$ be normed vector spaces. The nuclear norm $N_{X,Y}$ satisfies the following.*

1. *The infima in Definition 3.3.2 of $N_{X,Y}$ are attained.*
2. *The number of dyads at such an optimal decomposition is no more than $mnpq$.*

Proof. We first consider the alternative definition of the nuclear norm as the gauge function of $S_{X\otimes Y}$, the nuclear unit ball (3.14). That is,

$$N_{X,Y}(\mathcal{A}) = \inf\{t : t \geq 0, \mathcal{A} \in t \cdot S_{X\otimes Y}\}.$$

This is the infimum of a linear function over a compact set, and therefore it is attained.

Furthermore, the optimal $t^* = N_{X,Y}(\mathcal{A}) > 0$ whenever $\mathcal{A} \neq \mathbf{0}$, and then $(1/t^*)\mathcal{A}$ lies on the boundary of $S_{X\otimes Y}$. Using the definition (3.14) of $S_{X\otimes Y} \subseteq \mathbb{O}^{m\times n\otimes p\times q}$ as the convex hull of $D_{X\otimes Y}$ (the set (3.12) of dyads with unit-norm factors), we can write $(1/t^*)\mathcal{A}$ as the convex combination

$$\frac{1}{t^*}\mathcal{A} = \sum_{i=1}^{mnpq} \lambda_i \boldsymbol{X}_i \otimes \boldsymbol{Y}_i,$$

where $mnpq$ is the dimension of $\mathbb{O}^{m\times n\otimes p\times q}$, $\|\boldsymbol{X}_i\|_X = 1$, and $\|\boldsymbol{Y}_i\|_Y = 1$.

We claim that $\mathcal{A} = \sum_{i=1}^{mnpq}(t^*\lambda_i \boldsymbol{X}_i) \otimes \boldsymbol{Y}_i$ is an optimal decomposition in the definition (3.10). Indeed,

$$\sum_{i=1}^{mnpq} = \|t^*\lambda_i \boldsymbol{X}_i\|_X \|\boldsymbol{Y}_i\|_Y = \sum_{i=1}^{mnpq} |t^*\lambda_i| = t^* = N_{X,Y}(\mathcal{A}),$$

where we have used the fact that the λ_i are the coefficients of a convex combination and that t^* is positive.

Similarly, $\mathcal{A} = \sum_{i=1}^{mnpq} t^*\lambda_i(\boldsymbol{X}_i \otimes \boldsymbol{Y}_i)$ is the equivalent optimal decomposition in the definition (3.11). In the case where $\mathcal{A} = \mathbf{0}$, it is clear that $\mathcal{A} = \mathbf{0}\otimes\mathbf{0} = 1\cdot\mathbf{0}\otimes\mathbf{0}$ is an optimal decomposition. Therefore, the number of dyads in any optimal decomposition is at most $mnpq$, the dimension of $\mathbb{O}^{m\times n\otimes p\times q}$. □

Appendix B

Denoising experiments

This appendix provides the details of the denoising experiments in Chapter 5.

B.1 The synthetic denoising experiments

In this section we describe the protocol for our synthetic denoising experiments, discussing the parameters that we may vary at each step of the procedure. Our subsequent numerical experiments result from a principled exploration of this parameter space, and we will provide more detailed information about our particular choices when discussing those specific experiments.

B.1.1 Overview

All of our synthetic denoising experiments result from creating and solving individual denoising problems while systematically varying the parameters used therein. In this section we summarize the procedure used to create single denoising problems as well as our method for hierarchically testing the parameters.

B.1.1.1 A single denoising problem

We split the creation and solution of a synthetic denoising problem into three main stages, and we summarize the procedure:

1. **Operator generation:** We construct the true signal $\mathcal{A}^\natural \in \mathbb{O}^{m\times n\otimes p\times q}$, a low-rank operator whose factors have distinguished structure.

a) We choose the dimensions m, n, p, q of the true signal $\mathcal{A}^\natural$ and its rank r.

b) We generate left factors $\{\boldsymbol{X}_i\}_{i=1}^r$ (in $\mathbb{M}^{m\times n}$) having a chosen structure, and we do the same for the right factors $\{\boldsymbol{Y}_i\}_{i=1}^r$ (in $\mathbb{M}^{p\times q}$).

c) We combine the factors to make the true signal $\mathcal{A}^\natural = \sum_{i=1}^r \boldsymbol{X}_i \otimes \boldsymbol{Y}_i$.

d) We verify that $\mathcal{A}^\natural$ indeed has rank r. If not, we repeat the generation and combination steps.

e) In order to facilitate comparisons between different signals, we normalize $\mathcal{A}^\natural$ by dividing each factor by $1/\sqrt{\|\mathcal{A}^\natural\|_{\ell_2}}$. For simplicity, we take $\mathcal{A}^\natural$ to mean the normalized signal.

2. **Noise generation:** We corrupt the true signal with noise.

   a) Generate a random operator $\mathcal{Z} \in \mathbb{O}^{m\times n\otimes p\times q}$ with independent standard normal entries, and select a noise level σ.

   b) Create the noisy measurements $\mathcal{B} = \mathcal{A}^\natural + \sigma\mathcal{Z}$.

3. **Solver options and solution:** We choose a regularizer and call on `altminsolve`—and, if applicable, `matsolve` and `sdpsolve`—to compute the estimate $\widehat{\mathcal{A}}$.

   a) Select a regularizer.

   b) Compute a range of appropriate penalty constants.

   c) Specify options to pass to `altminsolve`, `matsolve`, and `sdpsolve`.

   d) Call the solver(s) to compute the estimate $\widehat{\mathcal{A}}$, and compute the desired error metrics.

Each of these steps has a set of parameters associated with it, and the following sections describe them in more detail.

B.1.1.2 Testing the parameter space

We test the parameters in a hierarchical fashion. That is, for each combination of operator generation parameters, we create one operator $\mathcal{A}^\natural$. And then for each noise power we form the noisy observation $\mathcal{B}$. We denoise this $\mathcal{B}$ using all combinations of the solver options.

While we could generate a different $\mathcal{A}^\natural$ for each call to the solvers, this hierarchical approach saves some time. It also matches our desires in testing. We think that choosing the regularizer to match the operator structure will result in better performance, and this procedure indeed tests all of the regularizers for each generated $\mathcal{A}^\natural$.

Note that we only test the combinations of factor structures that are unique up to ordering. That is, if we test sparse left factors with Gaussian right factors, we do not test Gaussian left factors with sparse right factors. We do, however, test all combinations of nuclear norms.

Additionally, we only test cases where the number of dyads in the solver is at least as large as the true operator rank. This makes sense as we wish to look at what happens when we use fewer dyads than the maximal operator rank, but we do not want the number of dyads to be smaller than the number actually necessary to represent the true signal.

B.1.2 Operator generation

We choose dimensions for $\mathcal{A}^\natural \in \mathbb{O}^{m\times n\otimes p\times q}$ and its rank r. Recall that the rank[1] of $\mathcal{A}^\natural$ is equivalent to the rank of the matrix $\mathrm{mat}(\mathcal{A}^\natural)$ (see Section 3.2.2). Therefore the maximal rank of $\mathcal{A}^\natural$ is $\min\{mn, pq\}$.

Finally, we must specify how exactly we will generate the dyads used to construct $\mathcal{A}^\natural$. We select one structure that is shared by all of the left factors $\boldsymbol{X}_i$ and another (possibly same) structure that is shared by all of the right factors $\boldsymbol{Y}_i$. In these experiments we consider the following choices:

- **Random Gaussian matrix:** The entries of the matrix are independent $\textsc{normal}(0, 1)$ variates.
- **Random 1-sparse matrix:** We set one randomly chosen entry of the matrix to $+1$ or -1 with equal probability. The remaining entries are identically zero.
- **Random sign matrix:** Each entry is independently filled by either $+1$ or -1 with equal probability.

[1] We take the *rank* of an operator to mean the smallest number of dyads that sum to that operator. Sometimes *rank* is used interchangeably with *order* when referring to tensors.

- **Random orthogonal matrix:** We generate these matrices using a QR decomposition of random Gaussian matrices. Some care must be taken to ensure that we sample uniformly from the distribution of orthogonal matrices. See [Mez07] for details.

- **Random rank-1 matrix:** We generate a random rank-1 matrix in $\mathbb{M}^{m\times n}$ as the outer product of random unit vectors $\boldsymbol{u} \in \mathbb{R}^m$ and $\boldsymbol{v} \in \mathbb{R}^n$.

We consider these structures precisely for their connection with the ℓ_2, ℓ_1, ℓ_∞, S_∞, and S_1 norms respectively (see Section 3.3.6 and Table 3.1).

Note that just because a matrix can be written as the sum of r dyads does *not* mean that it has rank equal to r. In some cases, choosing the factors of the r dyads independently can easily lead to operators with rank smaller than r. Sparse factors, in particular, are susceptible to this when the desired rank r is close to its maximal value. We guard against this by checking the rank of the generated $\mathcal{A}^\natural$ and regenerating it if necessary.

B.1.3 Noise generation

For our experiments, we only generate random noise operators $\mathcal{Z}$ with independent standard normal entries. So all that remains is for us to choose the noise power σ^2. It is more useful for us, however, to consider the signal-to-noise ratio (SNR) measured in decibels (dB):

$$\mathrm{SNR} := 10\log_{10}\left(\frac{\|\mathcal{A}^\natural\|_{\ell_2}^2}{\|\sigma\mathcal{Z}\|_{\ell_2}^2}\right) \approx 10\log_{10}\left(\frac{\|\mathcal{A}^\natural\|_{\ell_2}^2}{\sigma^2\cdot mnpq}\right).$$

In particular, since we normalize the true operator so that $\|\mathcal{A}^\natural\|_{\ell_2} = 1$, we can find the noise power σ^2 for a target SNR as

$$\sigma^2 = \frac{10^{-\mathrm{SNR}/10}}{mnpq}.$$

B.1.4 Solver options

For each problem, we need to specify a nuclear norm as a regularizer. In `operfact`, these are objects of type `NucNorm_Sum`, `NucNorm_Prod`, and `NucNorm_SDR`. We describe them in Section 4.6.2.

Of particular interest to us are the nuclear norms involving ℓ_2, ℓ_1, ℓ_∞, S_∞, and S_1 norms since these correspond to the factor structures under consideration. We also consider the relaxed nuclear norms involving the superquadratic ℓ_2 and ℓ_∞ norms.

In Section 5.3.2 we discuss the computation of the optimal penalty constant when the regularizer is a norm, and we discuss the function that implements this computation in Section B.2 of this appendix. Since these are estimates of the optimal regularization constant, however, we also introduce an "offset" parameter that allows us to test a range of constants. If we let λ_0 denote the computed constant, we pick offsets j and compute $\lambda_j = 2^j \lambda_0$. Note that $j = -\infty$ (i.e., $\lambda_{-\infty} = 0$) corresponds to using no regularization.

We then choose a solver from our `operfact.solvers` module (Section 4.7). This choice is at least partially determined by the regularizer we use: in most cases the alternating minimization solver is the only one available. Some nuclear norms, however, admit simple implementations with the convex solver `matsolve`. We also use `sdpsolve` for the semidefinite relaxations.

Since each of our outer solvers relies on the Python package CVXPY [DB16] to construct and solve the optimization problem, we must specify which external solver CVXPY should use. We call this the "inner" solver, and we use SCS [OCPB16] for our experiments. This first-order solver has the advantage of allowing us to test higher-dimensional problems than the default interior-point solver CVXOPT. Because of the relatively slow convergence of first-order methods, we pay for the better scaling by solving the problem to lower accuracy.

Each of the solvers that CVXPY calls has its own set of hyperparameters that we may specify. In these experiments we use the default options for SCS, but we enable warm-starting for our alternating minimization solver.

In problems where we use the alternating minimization solver `altminsolve`, we have some additional parameters to specify. Most importantly, this solver allows us explicitly specify the number of dyads we use in the solution. We will explore the effect of this hyperparameter through experimentation, but we should note now that if we restrict `altminsolve` to using a smaller number of dyads than the rank of the true signal $\mathcal{A}^\natural$, it could not reconstruct this signal even in the case of no noise. This follows directly from the definition of operator rank. Therefore, we do not test combinations where the solver rank is less than

Operator generation	
Operator shape	$4 \times 4 \otimes 4 \times 4$
Operator rank	1, 2, 4, 8
Factor structures	all combinations of random 1-sparse, Gaussian, sign, rank-1, and orthogonal matrices
Noise generation	
Noise level	0dB, 5dB, 10dB, 15dB, 20dB
Solver options	
Regularizer	all nuclear norms involving ℓ_1, ℓ_2, ℓ_∞, S_1, and S_∞ semidefinite relaxations for $\ell_2 \otimes \ell_\infty$, $\ell_\infty \otimes \ell_2$, and $\ell_\infty \otimes \ell_\infty$
Regularization constant	offsets $= -\infty, -3, -2, -1, 0, 1$
Outer solver	`altminsolve`; `matsolve` and `sdpsolve`, where applicable
Inner solver	SCS
Inner solver options	default
`altminsolve` only:	
Number of dyads in solution	1, 2, 4, 8, 16
Relative convergence threshold	1×10^{-1}, 5×10^{-2}, 1×10^{-2}, 5×10^{-3}, 1×10^{-3}, 5×10^{-4}
Max. outer iterations	25

Table B.1: **Denoising parameters,** $(4 \times 4 \otimes 4 \times 4)$**.** This table lists the parameters tested in our synthetic denoising experiment for operators of size $(4 \times 4 \otimes 4 \times 4)$.

the operator rank.

The alternating minimization solver terminates when the relative change in the objective between outer iterations is below a given threshold or after we have completed a maximal number of outer iterations. We supply both of these thresholds to the solver.

B.1.5 Small experiment

The combinatorial nature of testing our parameter space requires that we control the amount of time it takes to solve any one particular problem. To that end, we use small dimensions $(4 \times 4 \otimes 4 \times 4)$ to perform initial tests that inform us on how to effectively limit the parameter space when running larger tests. Table B.1 summarizes the full set of parameters we test over 10 trials in this small-scale experiment.

B.1.6 A larger experiment

Following our initial experimentation, we narrow the parameter space and denoise larger operators. Table B.2 details these choices, and we again performed 10 trials of this experiment.

Operator generation	
Operator shape	$16 \times 16 \otimes 16 \times 16$
Operator rank	1, 2, 4, 8, 16
Factor structures	all combinations of random 1-sparse, Gaussian, sign, rank-1, and orthogonal matrices
Noise generation	
Noise level	10dB
Solver options	
Regularizer	all nuclear norms involving ℓ_1, ℓ_2, ℓ_∞, S_1, and S_∞ semidefinite relaxations for $\ell_2 \otimes \ell_\infty$, $\ell_\infty \otimes \ell_2$, and $\ell_\infty \otimes \ell_\infty$
Regularization constant	offsets $= -\infty, -3, -2, -1, 0, 1$
Outer solver	`altminsolve`; `matsolve` and `sdpsolve`, where applicable
Inner solver	SCS
Inner solver options	default
`altminsolve` only:	
Number of dyads in solution	1, 4, 16
Relative convergence threshold	1×10^{-1}, 5×10^{-2}, 1×10^{-2}, 5×10^{-3}, 1×10^{-3}
Max. outer iterations	25

Table B.2: **Denoising parameters,** $(16 \times 16 \otimes 16 \times 16)$**.** This table lists the parameters tested in our synthetic denoising experiment for operators of size $(16 \times 16 \otimes 16 \times 16)$.

B.2 The `penconst_denoise` function

The `penconst_denoise` function computes or estimates an appropriate penalty constant $\lambda_0 = \mathbb{E}\, N^*_{X,Y}(\mathcal{Z})$, where $N^*_{X,Y}$ is the dual norm and $\mathcal{Z}$ is random noise (see Section 5.3.2). We assume that $\mathcal{Z} \in \mathbb{O}^{m \times n \otimes p \times q}$ has independent standard normal entries.

Table B.3 lists the output of this function for various combinations of factor spaces X and Y. If a particular combination is not listed, the default $\lambda_0 = \sqrt{mn} + \sqrt{pq}$ is used.

We note that in some cases the dual norm has a closed form, and we can use Monte Carlo to estimate λ_0. In two cases we use a call to an extreme value distribution to compute the expectation.

Most of the cases, however, are heuristic guesses based on observations in the denoising experiment. We found that these values worked well, but we also note that we have not studied these choices over a large range of operator dimension. We advise calibration of the penalty constant for any application, and we note that using an offset like $\lambda_j = \lambda_0 2^j$ works well to test ranges of penalty constants.

X	Y	λ_0	
ℓ_1	ℓ_1	extreme value dist.[2]	
ℓ_1	ℓ_2	extreme value dist.	
ℓ_1	ℓ_∞	Monte Carlo	
ℓ_1	S_1	Monte Carlo	
ℓ_1	S_∞	Monte Carlo	
ℓ_2	ℓ_1	extreme value dist.	
ℓ_2	ℓ_2	$\sqrt{mn} + \sqrt{pq}$	
ℓ_2	ℓ_∞	$(\sqrt{mn} + \sqrt{pq})\sqrt{pq}$	(heuristic)
ℓ_2	S_1	$\sqrt{mn} + \sqrt{pq}$	(heuristic)
ℓ_2	S_∞	$(\sqrt{mn} + \sqrt{pq})\sqrt{\min\{p, q\}}$	
ℓ_∞	ℓ_1	Monte Carlo	
ℓ_∞	ℓ_2	$(\sqrt{mn} + \sqrt{pq})\sqrt{mn}$	(heuristic)
ℓ_∞	ℓ_∞	$(\sqrt{mn} + \sqrt{pq})\sqrt{mnpq}$	(heuristic)
ℓ_∞	S_1	$(\sqrt{mn} + \sqrt{pq})\sqrt{mn}$	(heuristic)
ℓ_∞	S_∞	$(\sqrt{mn} + \sqrt{pq})\sqrt{mn \cdot \min\{p, q\}}$	(heuristic)
S_1	ℓ_1	Monte Carlo	
S_1	ℓ_∞	$(\sqrt{mn} + \sqrt{pq})\sqrt{pq}$	(heuristic)
S_1	S_∞	$(\sqrt{mn} + \sqrt{pq})\sqrt{\min\{p, q\}}$	(heuristic)
S_∞	ℓ_1	Monte Carlo	
S_∞	ℓ_2	$(\sqrt{mn} + \sqrt{pq})\sqrt{\min\{m, n\}}$	(heuristic)
S_∞	ℓ_∞	$(\sqrt{mn} + \sqrt{pq})\sqrt{pq \cdot \min\{m, n\}}$	(heuristic)
S_∞	S_1	$(\sqrt{mn} + \sqrt{pq})\sqrt{\min\{m, n\}}$	(heuristic)
S_∞	S_∞	$(\sqrt{mn} + \sqrt{pq})\sqrt{pq \cdot \min\{m, n\}}$	(heuristic)

Table B.3: **The `penconst_denoise` function.** This table lists the estimate of the penalty constant $\lambda_0 = \mathbb{E}\, N^*_{X,Y}(\mathcal{Z})$ for the $X \otimes Y$ nuclear norm with standard Gaussian noise $\mathcal{Z} \in \mathbb{O}^{m\times n\otimes p\times q}$. In cases where the dual norm is easily computable, we may use "Monte Carlo" to estimate λ_0, and in two cases we appeal to an extreme value distribution.

B.3 Additional figures and tables for the denoising experiment

This section contains the additional figures and tables for the denoising experiment referenced in Chapter 5.

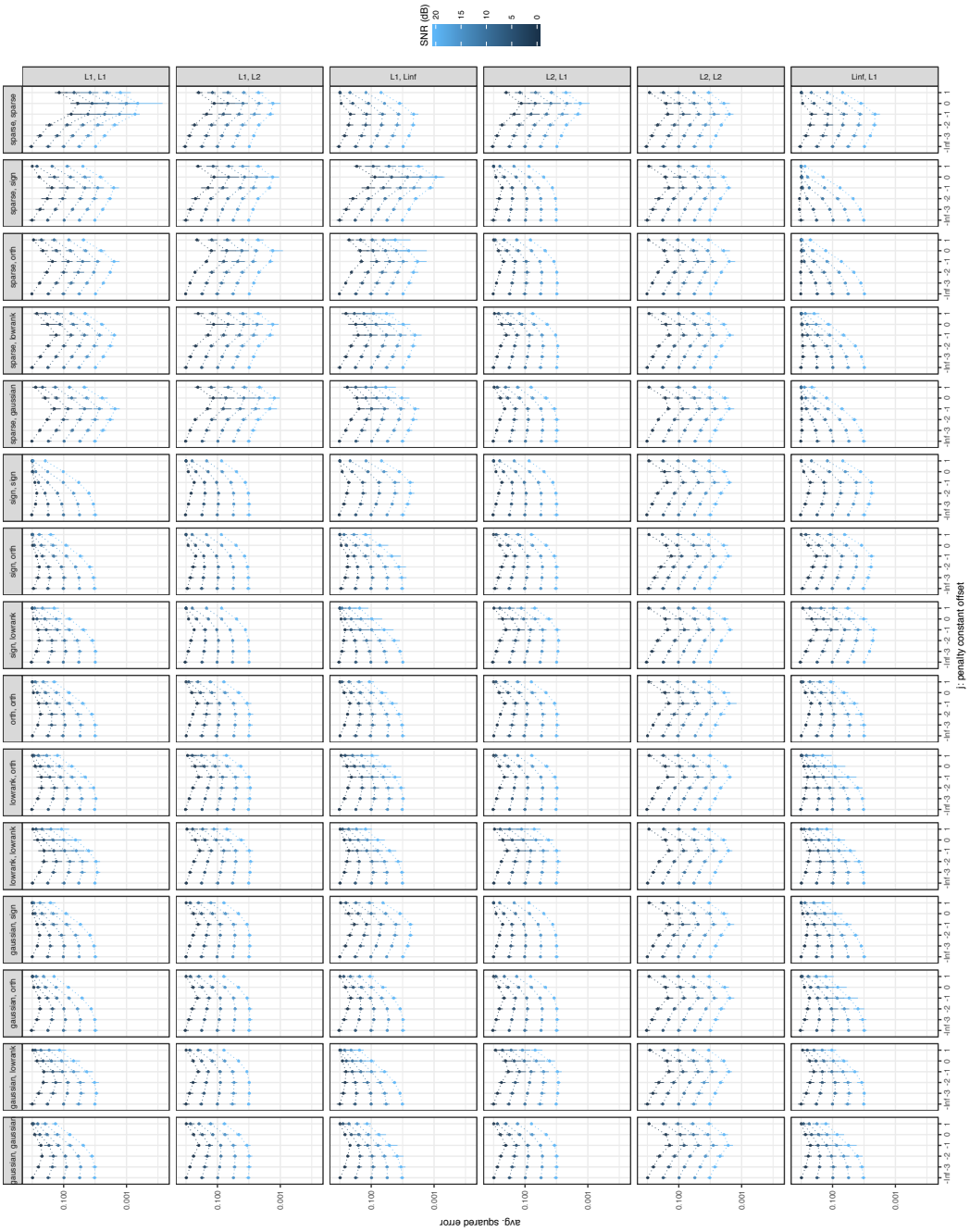


Figure B.1: **Average error vs. penalty constant, convex solver.** Each panel plots the average squared error (5.6) of the denoising procedure over 10 trials versus the penalty constant offset j (5.5) at various SNR (5.7) using the convex solver `matsolve` on rank-1 operators. Lighter hues correspond to higher SNR (less noise). Error bars show the minimum and maximum error over the trials. We facet the figure by factor structure (columns) and nuclear norms (rows).

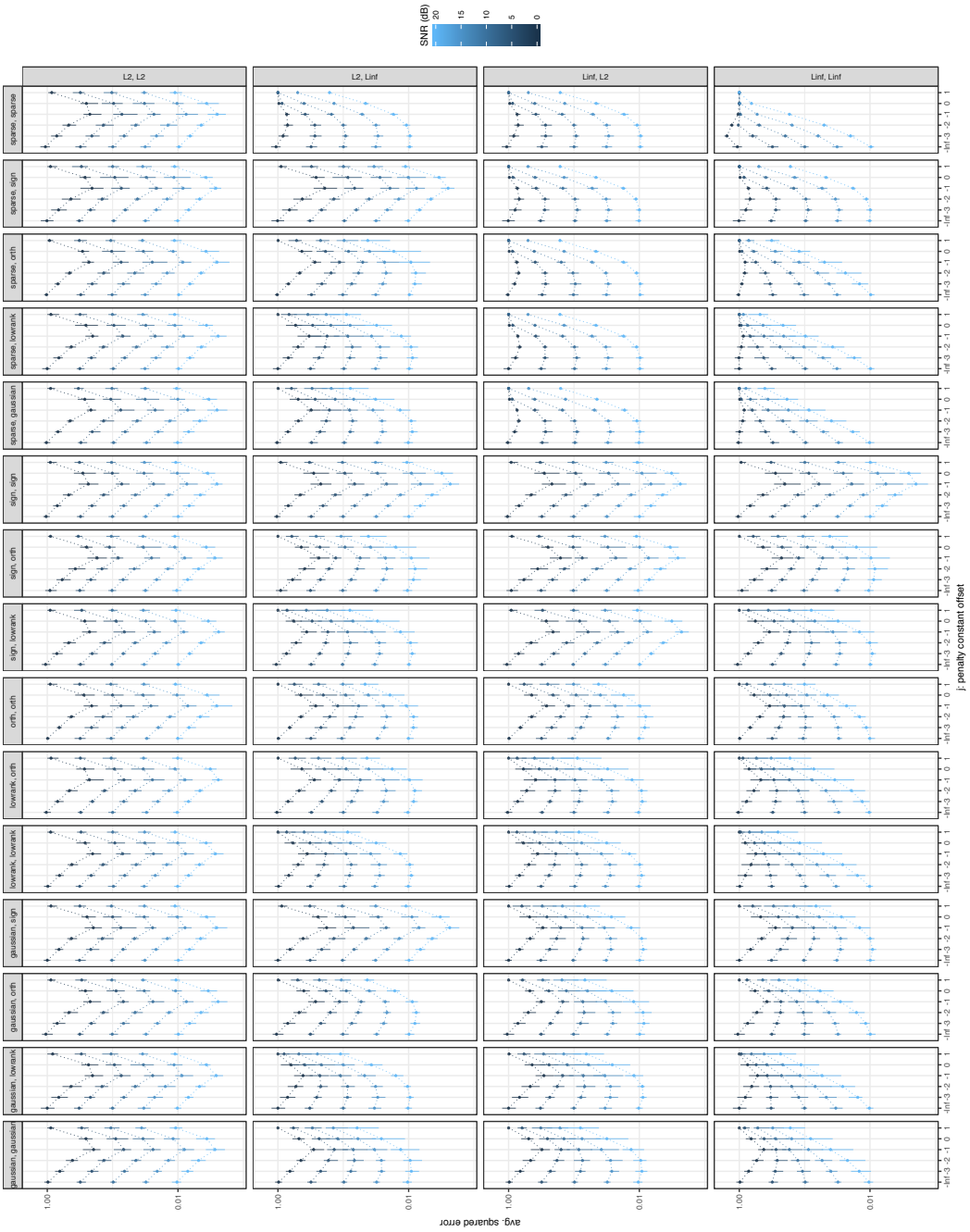


Figure B.2: **Average error vs. penalty constant, SDP solver.** Each panel plots the average squared error (5.6) of the denoising procedure over 10 trials versus the penalty constant offset j (5.5) at various SNR (5.7) using the SDP solver `sdpsolve` on rank-1 operators. Lighter hues correspond to higher SNR (less noise). Error bars show the minimum and maximum error over the trials. We facet the figure by factor structure (columns) and nuclear norms (rows).

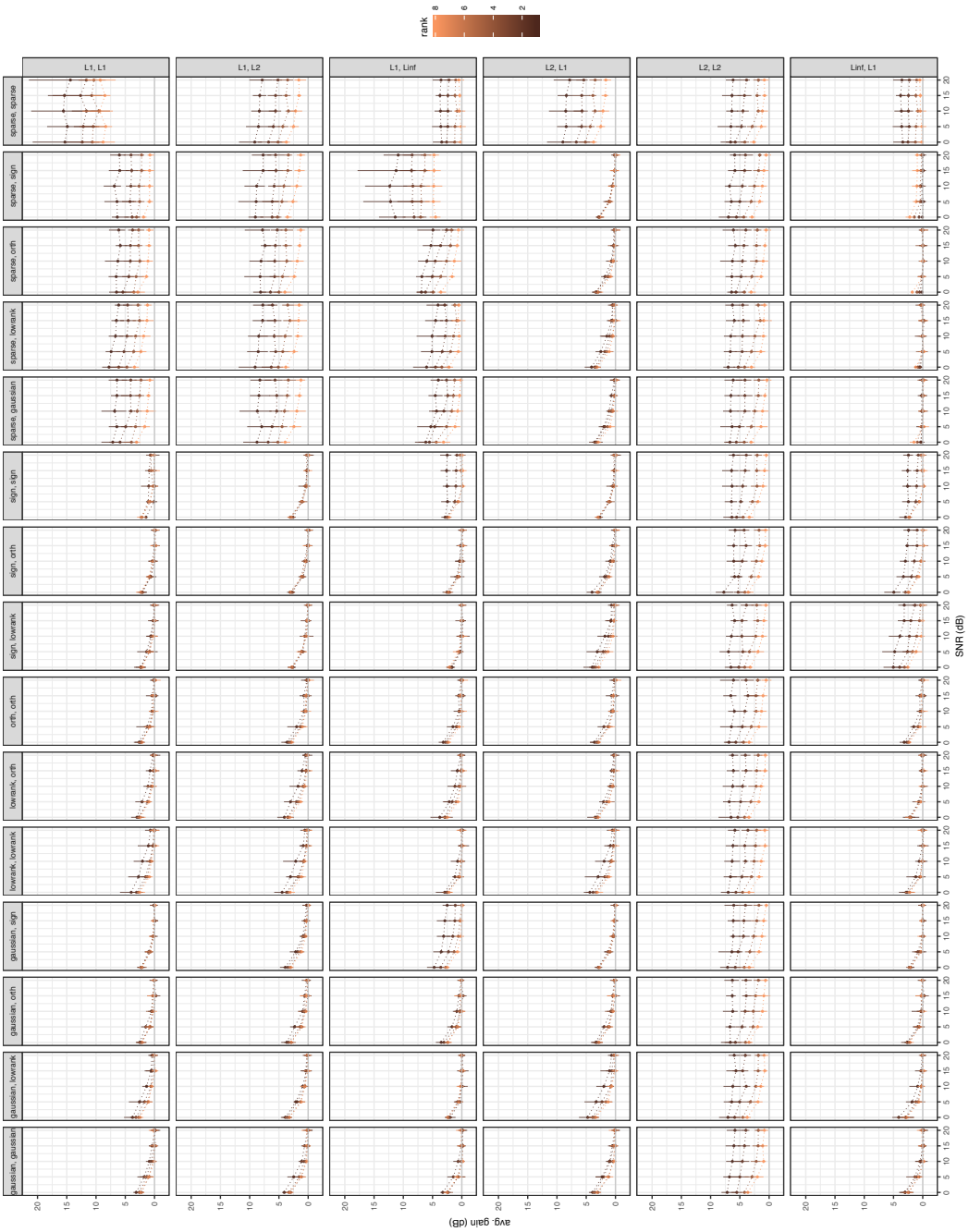


Figure B.3: **Average gain vs. SNR, convex solver.** Each panel plots the average gain in dB (5.9) of the denoising procedure over 10 trials versus the SNR (5.7) using the convex solver `matsolve`. Lighter hues correspond to higher rank operators. Error bars show the minimum and maximum gain over the trials. We facet the figure by factor structure (columns) and nuclear norms (rows).

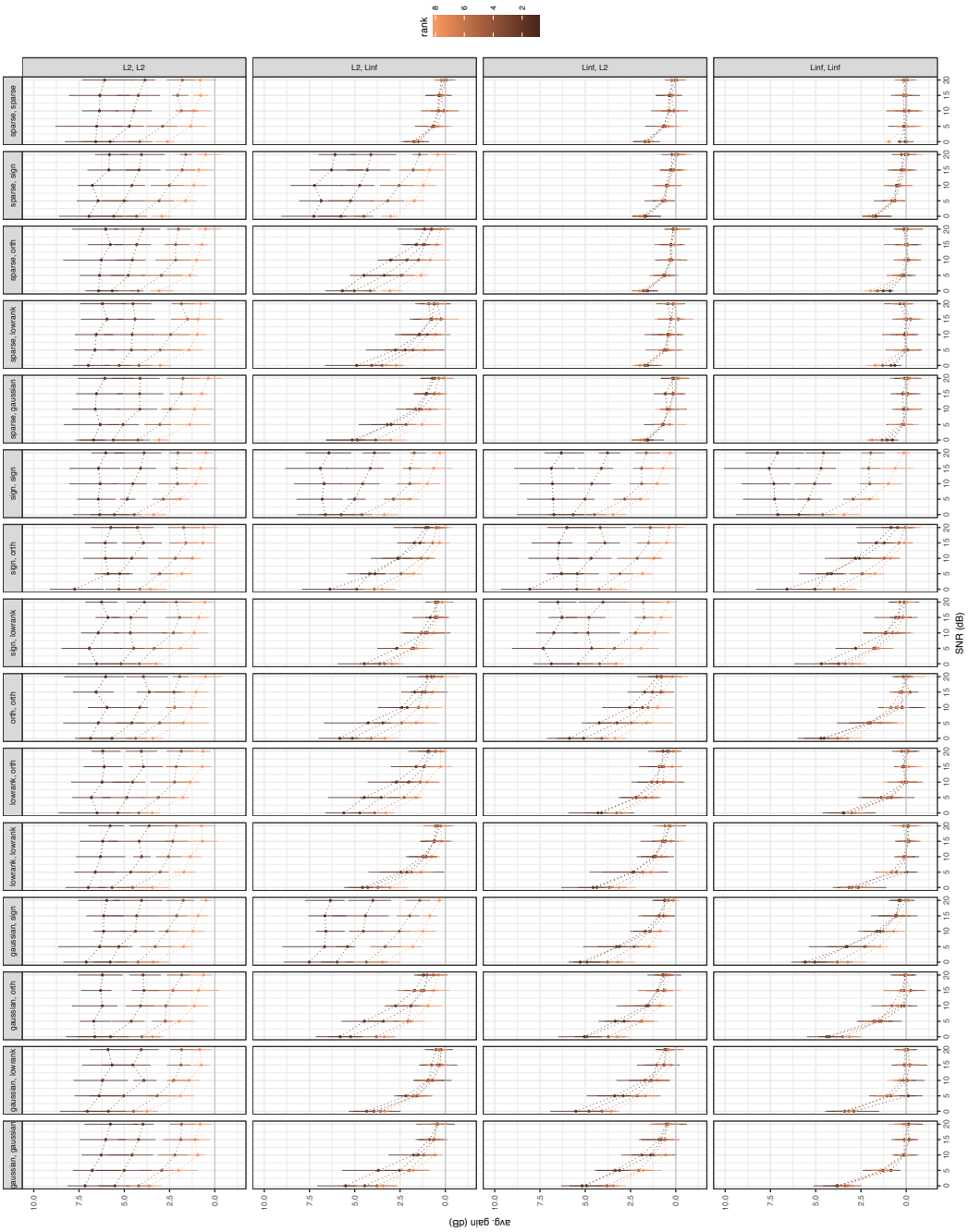


Figure B.4: **Average gain vs. SNR, SDP solver.** Each panel plots the average gain in dB (5.9) of the denoising procedure over 10 trials versus the SNR (5.7) using the SDP solver `sdpsolve`. Lighter hues correspond to higher rank operators. Error bars show the minimum and maximum gain over the trials. We facet the figure by factor structure (columns) and nuclear norms (rows).

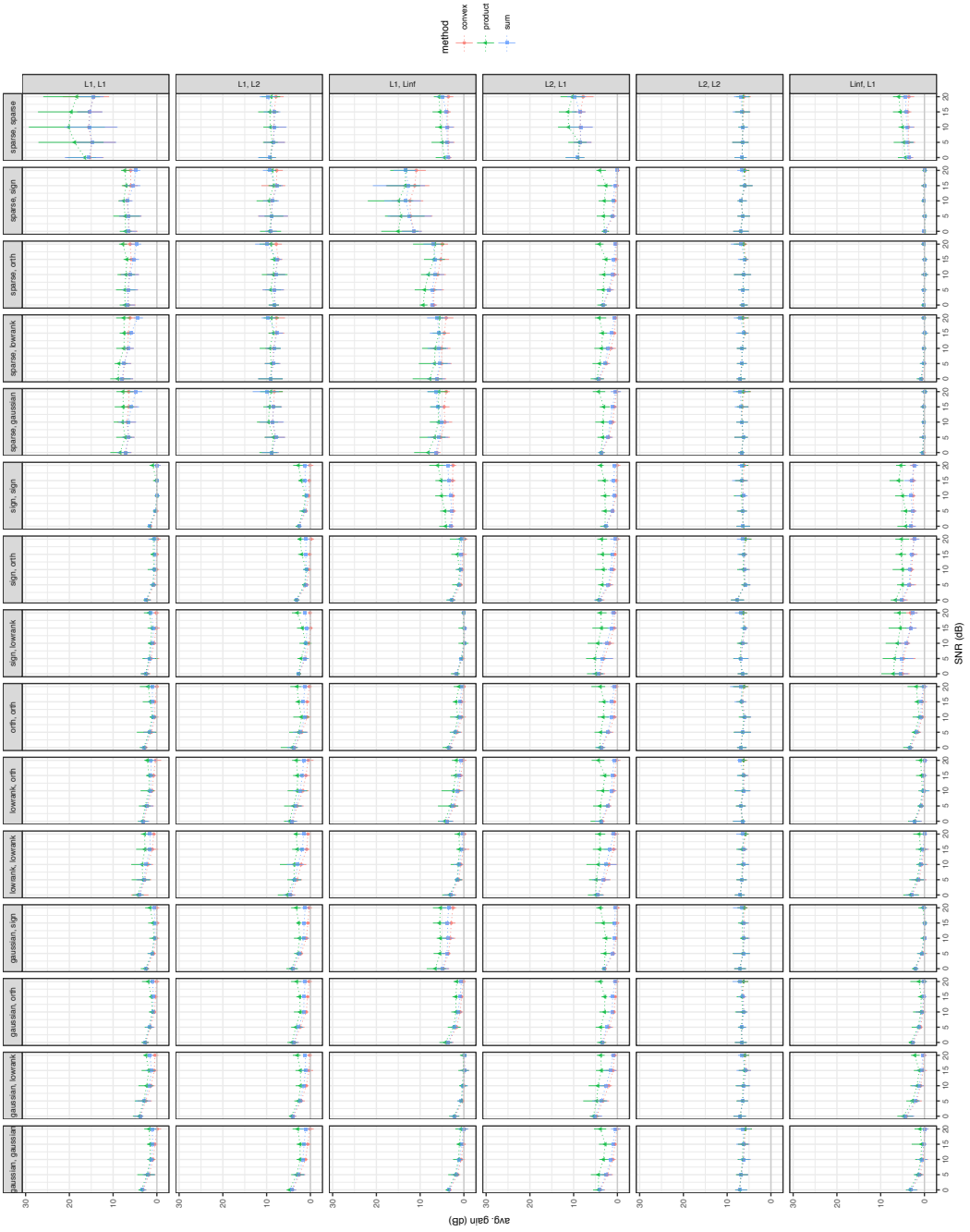


Figure B.5: **Average gain vs. SNR, convex and nonconvex solvers (full).** Each panel plots the average gain in dB (5.9) of the denoising procedure over 10 trials versus the SNR (5.7) with the color/shape indicating the solution method. We test the convex solver `matsolve` and both the sum (4.3) and product (4.2) formulations of the alternating minimization solver `altminsolve` with solver rank 16 and convergence tolerance $\epsilon = 5 \times 10^{-4}$ in (5.10). All operators have rank 1. Error bars show the minimum and maximum gain over the trials. We facet the figure by factor structure (columns) and nuclear norms (rows).

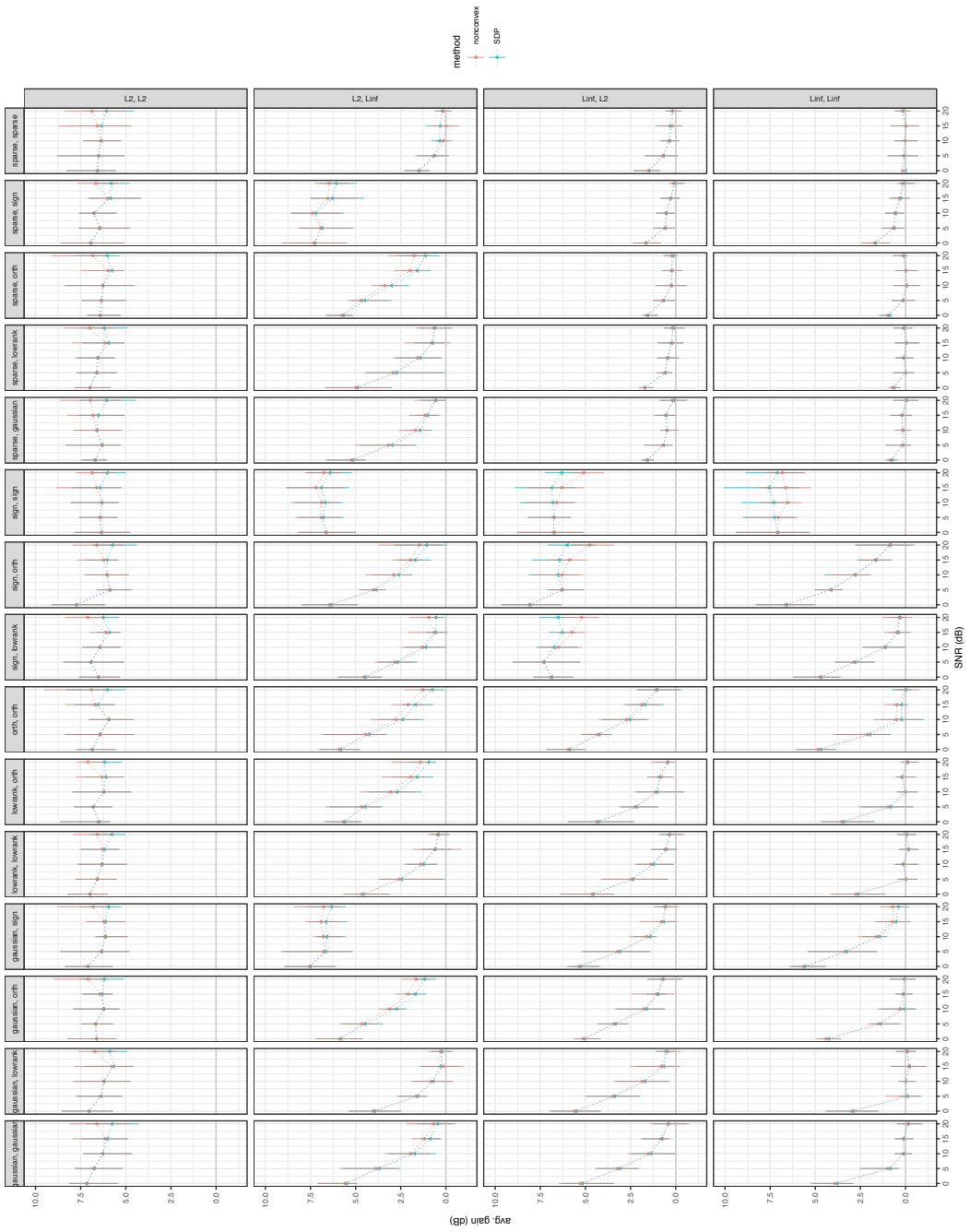


Figure B.6: **Average gain vs. SNR, SDP and nonconvex solvers (full).** Each panel plots the average gain in dB (5.9) of the denoising procedure over 10 trials versus the SNR (5.7) with the color/shape indicating the solution method. We test both the SDP solver `sdpsolve` and the alternating minimization solver `altminsolve` (with solver rank 16 and convergence tolerance $\epsilon = 5 \times 10^{-4}$ in (5.10)) on *relaxed* nuclear norms. All operators have rank 1. Error bars show the minimum and maximum gain over the trials. We facet the figure by factor structure (columns) and relaxed nuclear norms (rows).

Figure B.7a: **Average error vs. penalty constant, alternating minimization solver.** Each panel plots the average squared error (5.6) of the denoising procedure over 10 trials versus the penalty constant offset j (5.5) at various SNR (5.7) using the alternating minimization solver `altminsolve` on rank-1 operators. All problems use solver rank 16 and convergence tolerance $\epsilon = 5 \times 10^{-4}$ in (5.10). Lighter hues correspond to higher SNR (less noise). Error bars show the minimum and maximum error over the trials. We facet the figure by factor structure (columns) and nuclear norms (rows).

Figure B.7b: **Average error vs. penalty constant, alternating minimization solver.** (continued)

Figure B.7c: **Average error vs. penalty constant, alternating minimization solver.** (continued)

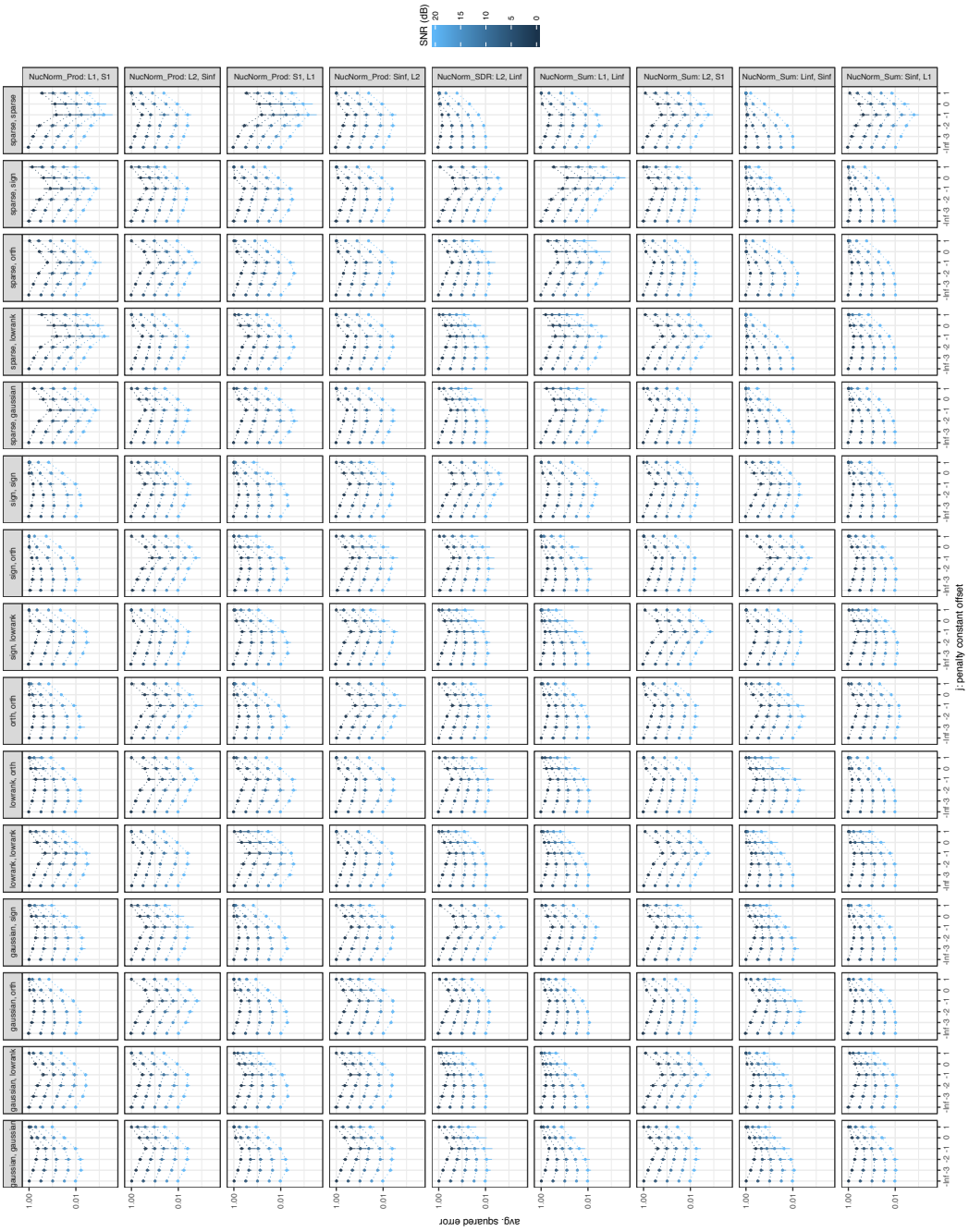


Figure B.7d: **Average error vs. penalty constant, alternating minimization solver.** (continued)

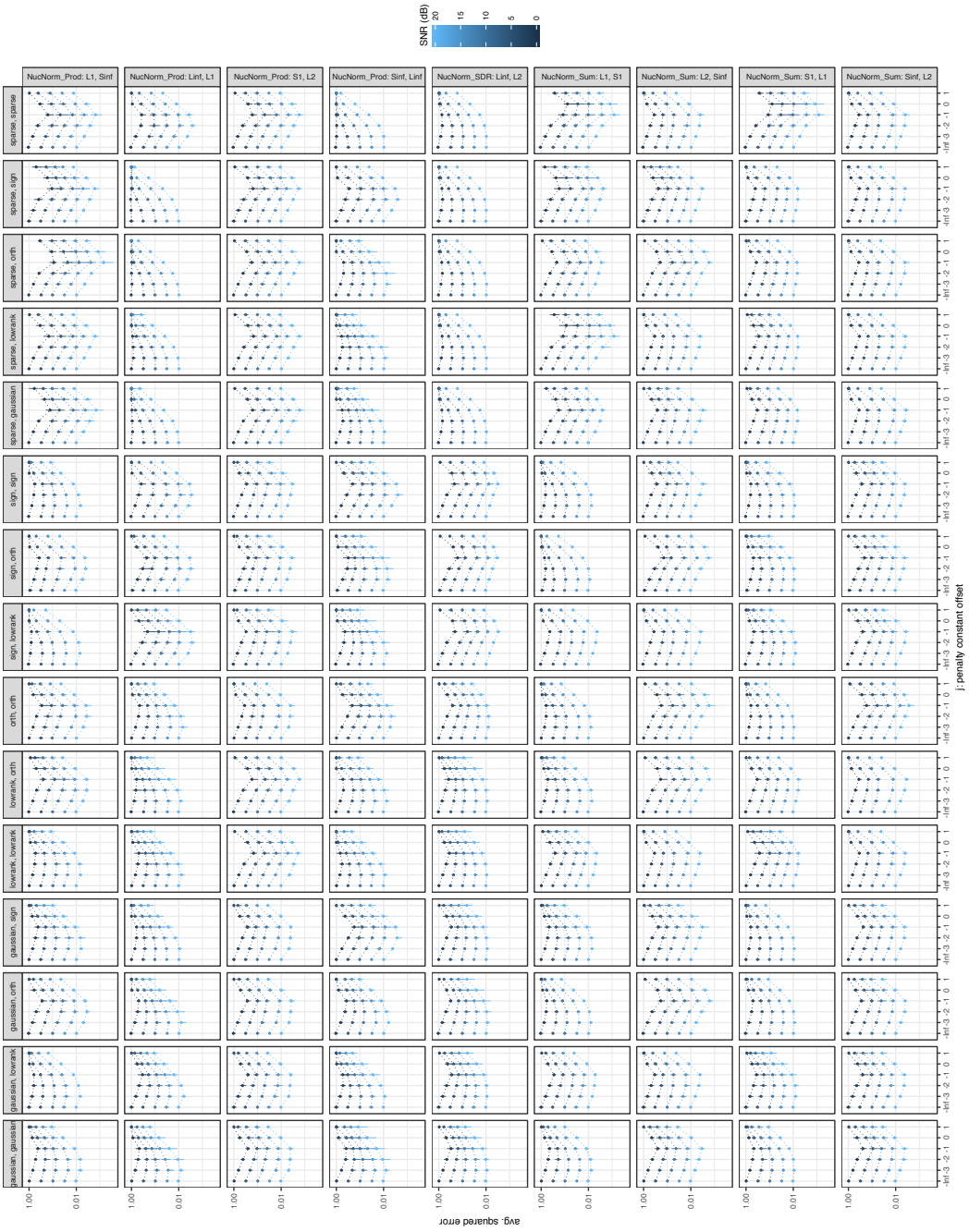


Figure B.7e: **Average error vs. penalty constant, alternating minimization solver.** (continued)

Figure B.7f: **Average error vs. penalty constant, alternating minimization solver.** (continued)

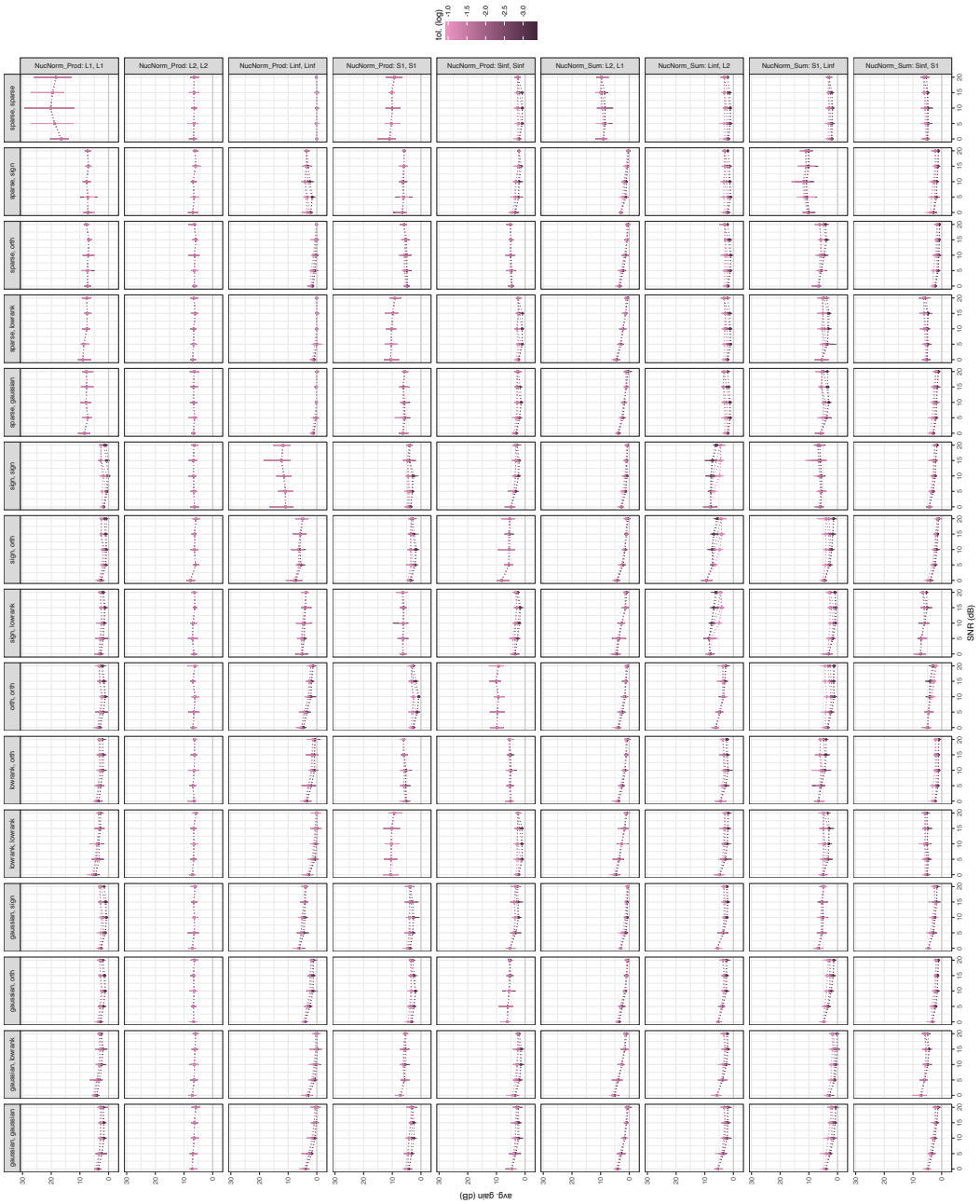


Figure B.8a: **Average gain vs. SNR, alternating minimization solver.** Each panel plots the average gain in dB (5.9) of the denoising procedure over 10 trials versus the SNR (5.7) with the color indicating the relative convergence tolerance (5.10) of the alternating minimization solver on a logarithmic scale. Lighter hues correspond to looser tolerances. Error bars show the minimum and maximum gain over the trials. We facet the figure by factor structure (columns) and nuclear norms (rows).

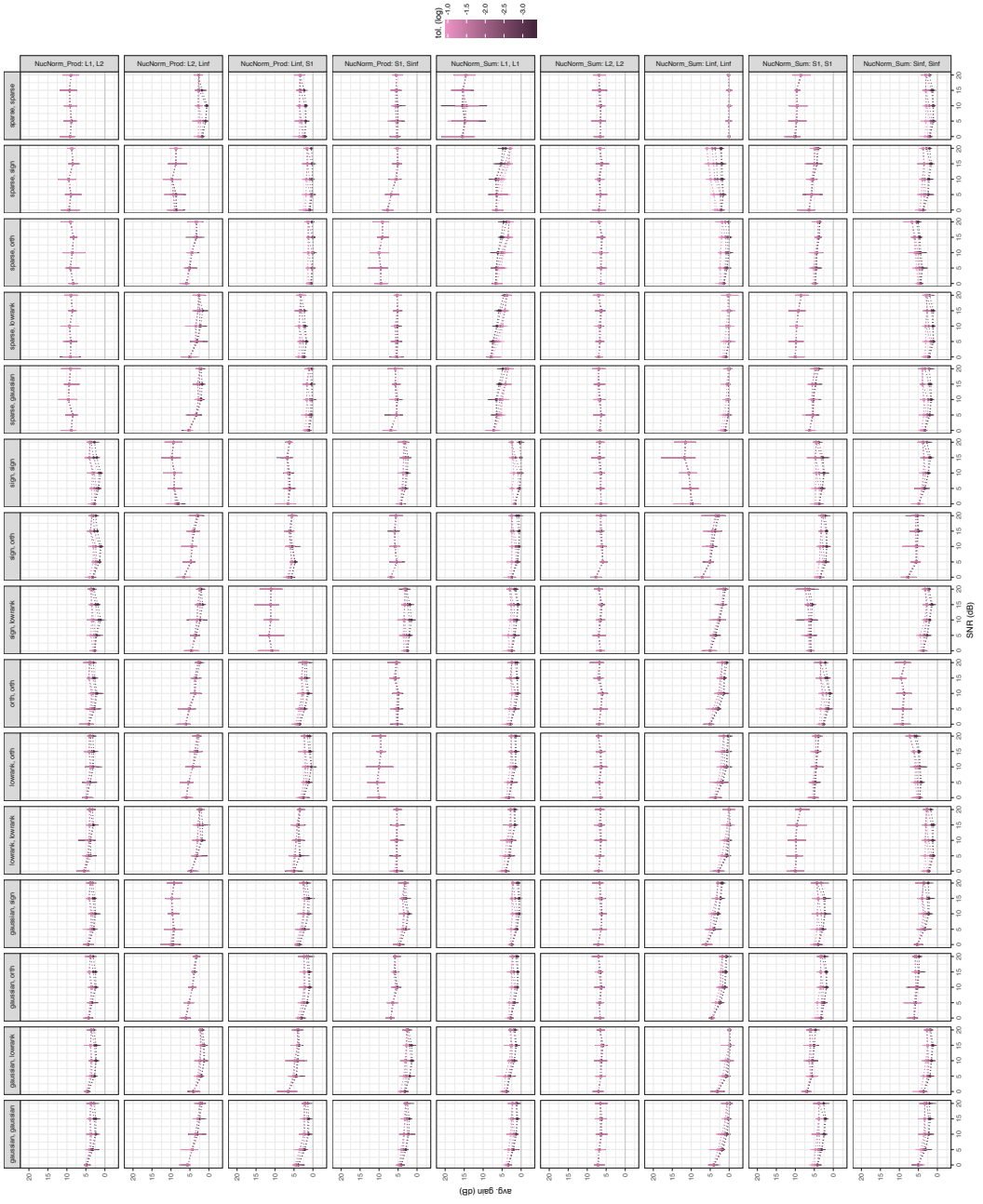


Figure B.8b: **Average gain vs. SNR, alternating minimization solver.** (continued)

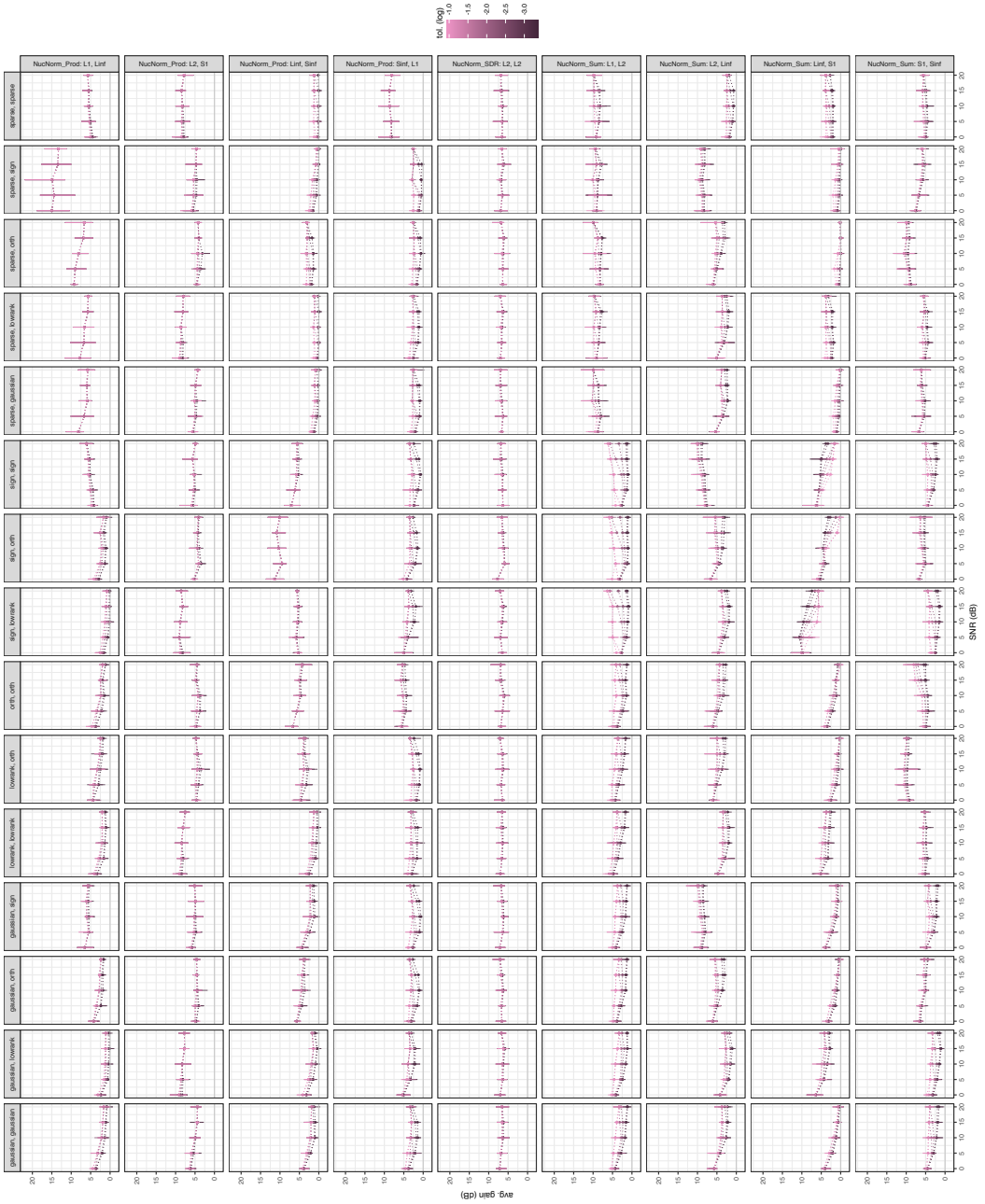


Figure B.8c: **Average gain vs. SNR, alternating minimization solver.** (continued)

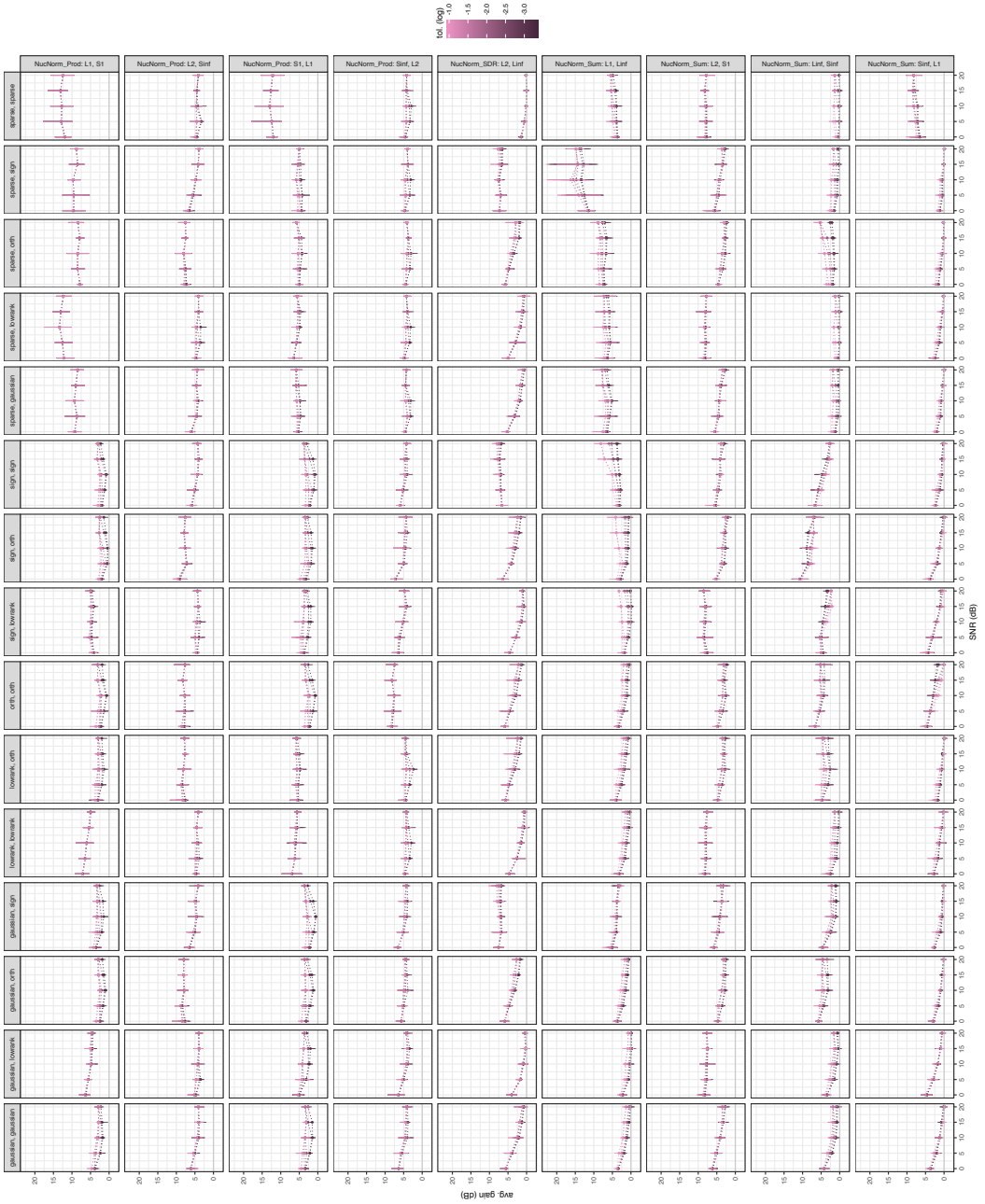


Figure B.8d: **Average gain vs. SNR, alternating minimization solver.** (continued)

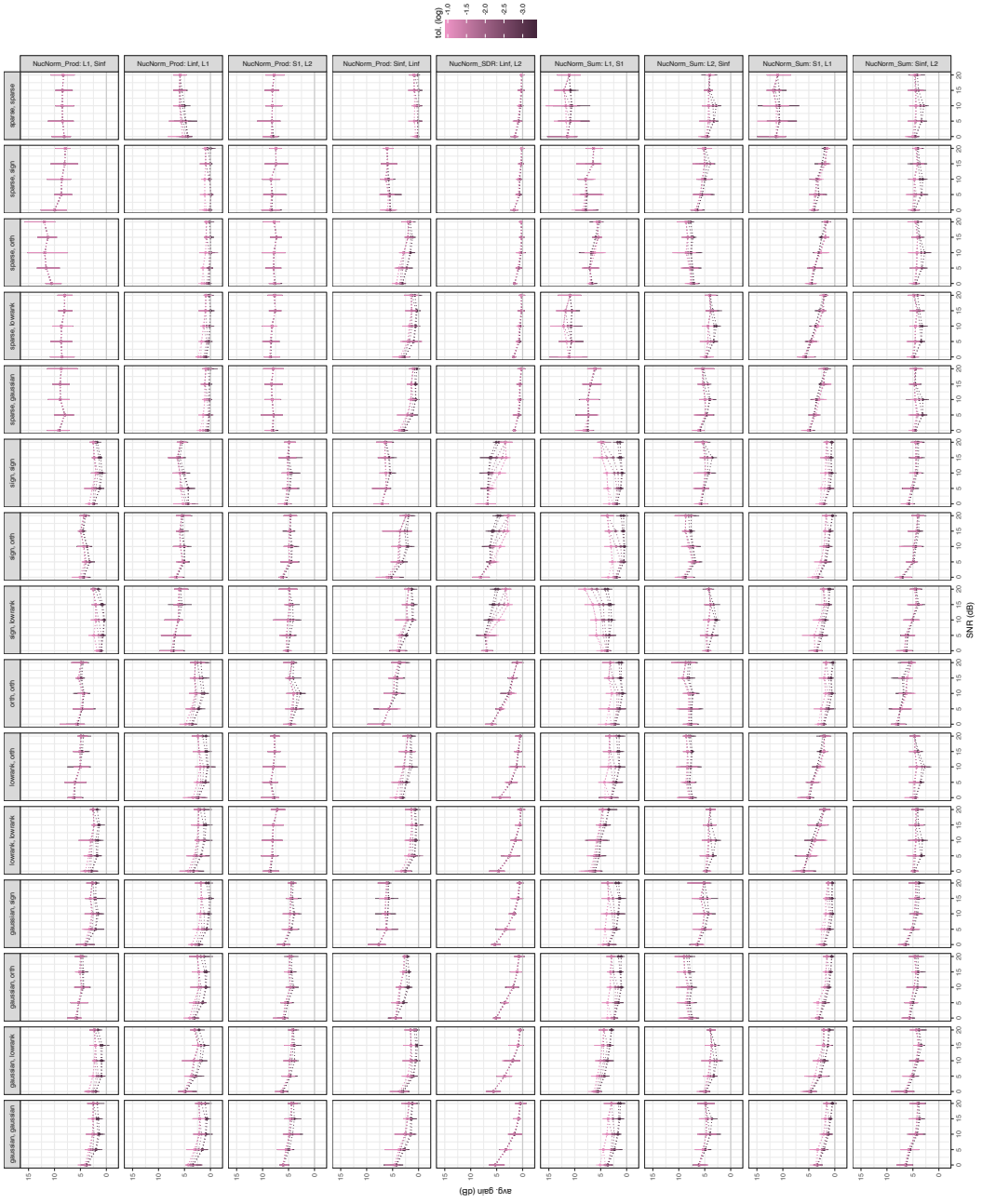


Figure B.8e: **Average gain vs. SNR, alternating minimization solver.** (continued)

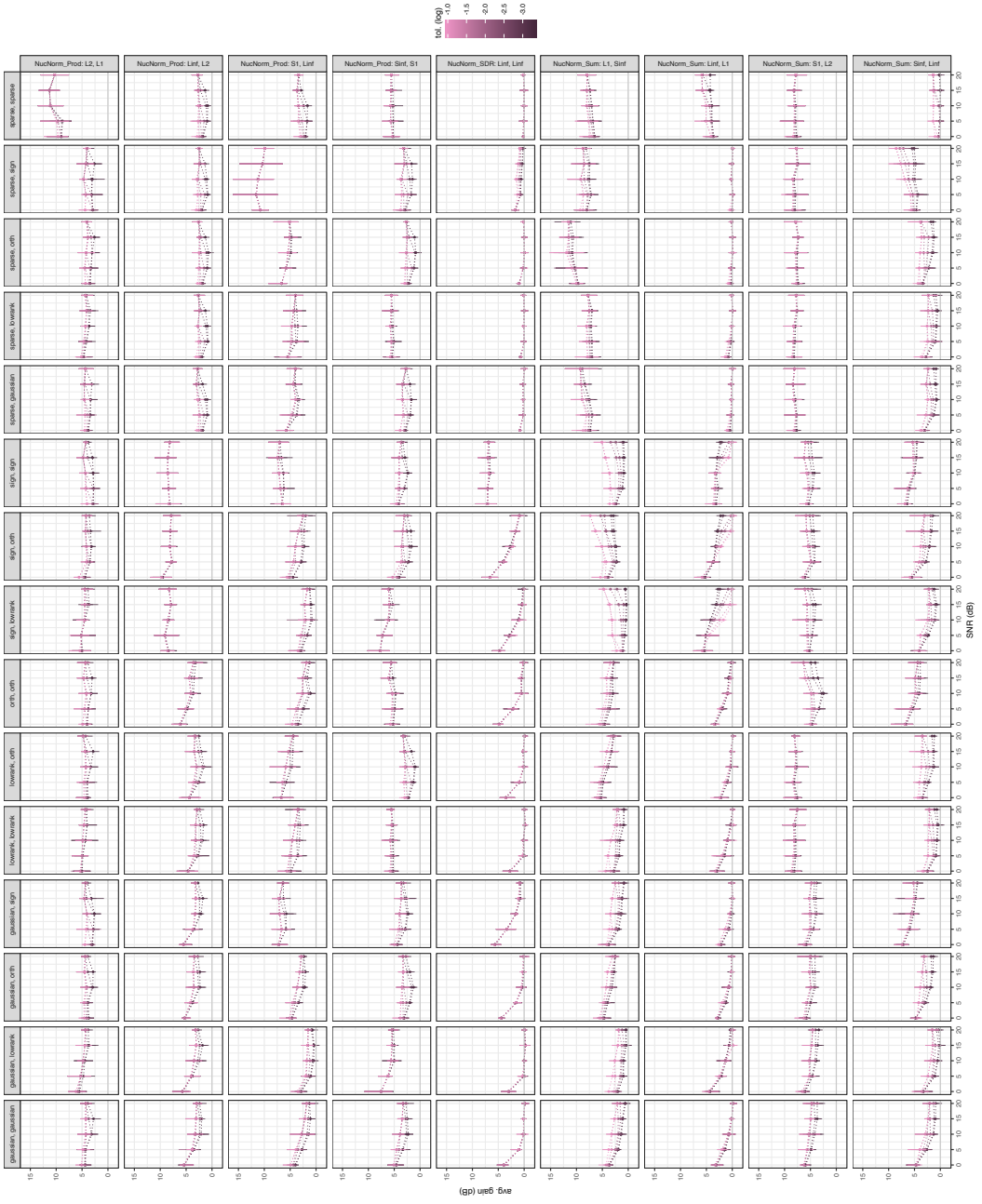


Figure B.8f: **Average gain vs. SNR, alternating minimization solver.** (continued)

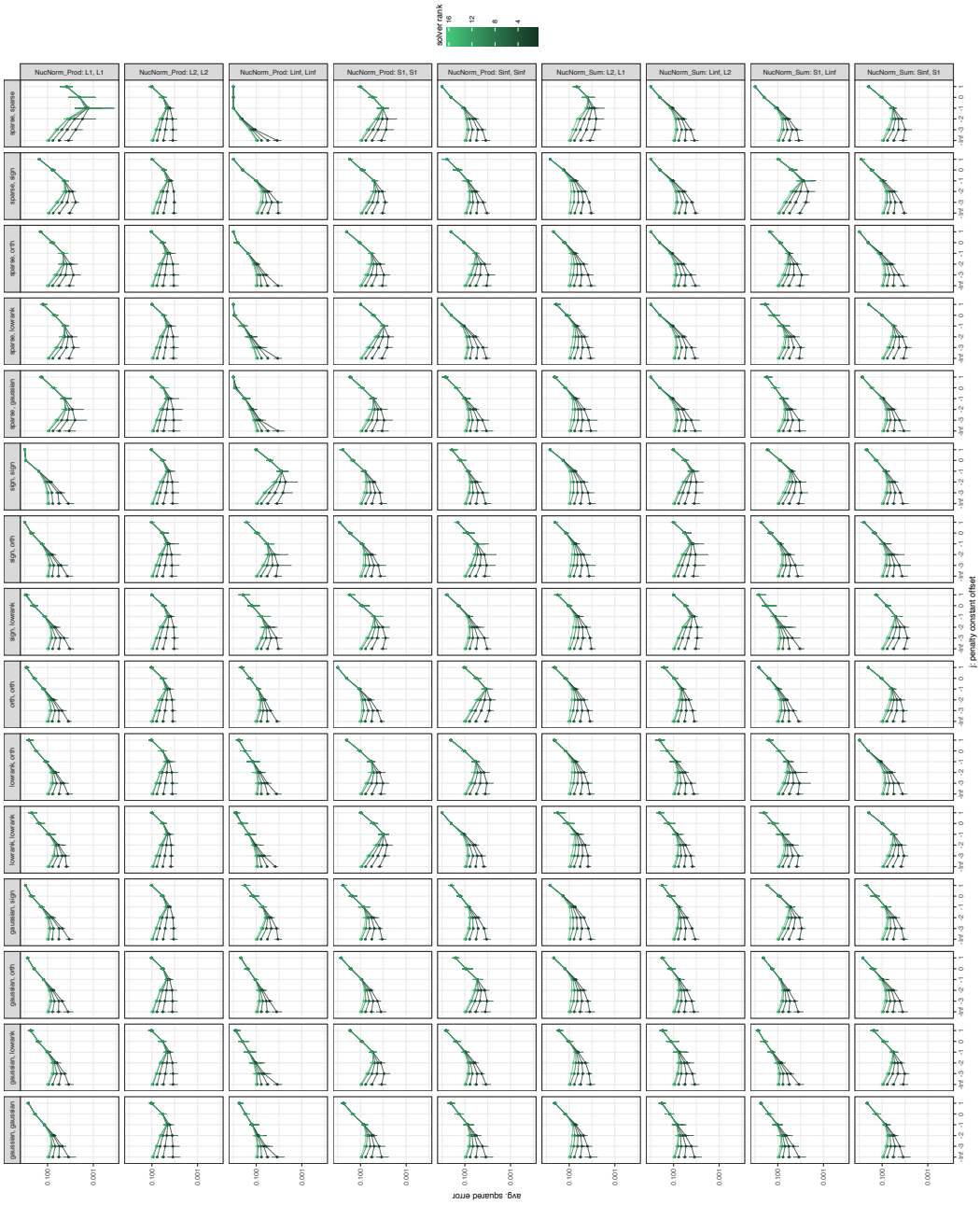


Figure B.9a: **Average error vs. penalty constant, solver rank (full).** Each panel plots the average squared error (5.6) of the denoising procedure over 10 trials versus the penalty constant offset j (5.5) at various solver ranks (r). Lighter hues correspond to higher solver ranks (i.e., more dyads used in the solution). Error bars show the minimum and maximum error over the trials. We facet the figure by factor structure (columns) and nuclear norm (rows). All tests were performed at an SNR (5.7) of 10dB, and all operators have rank 1.

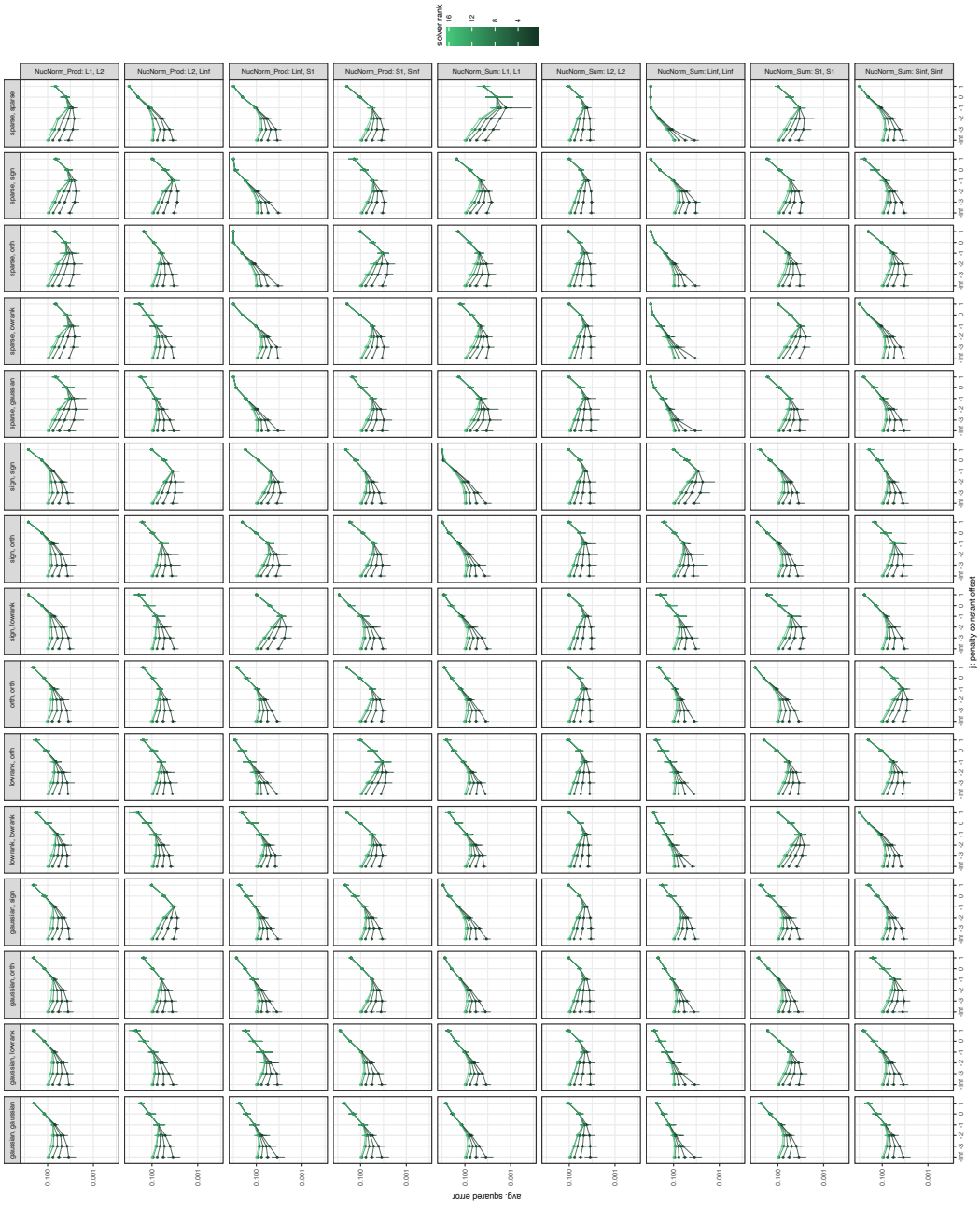


Figure B.9b: **Average error vs. penalty constant, solver rank (full).** (continued)

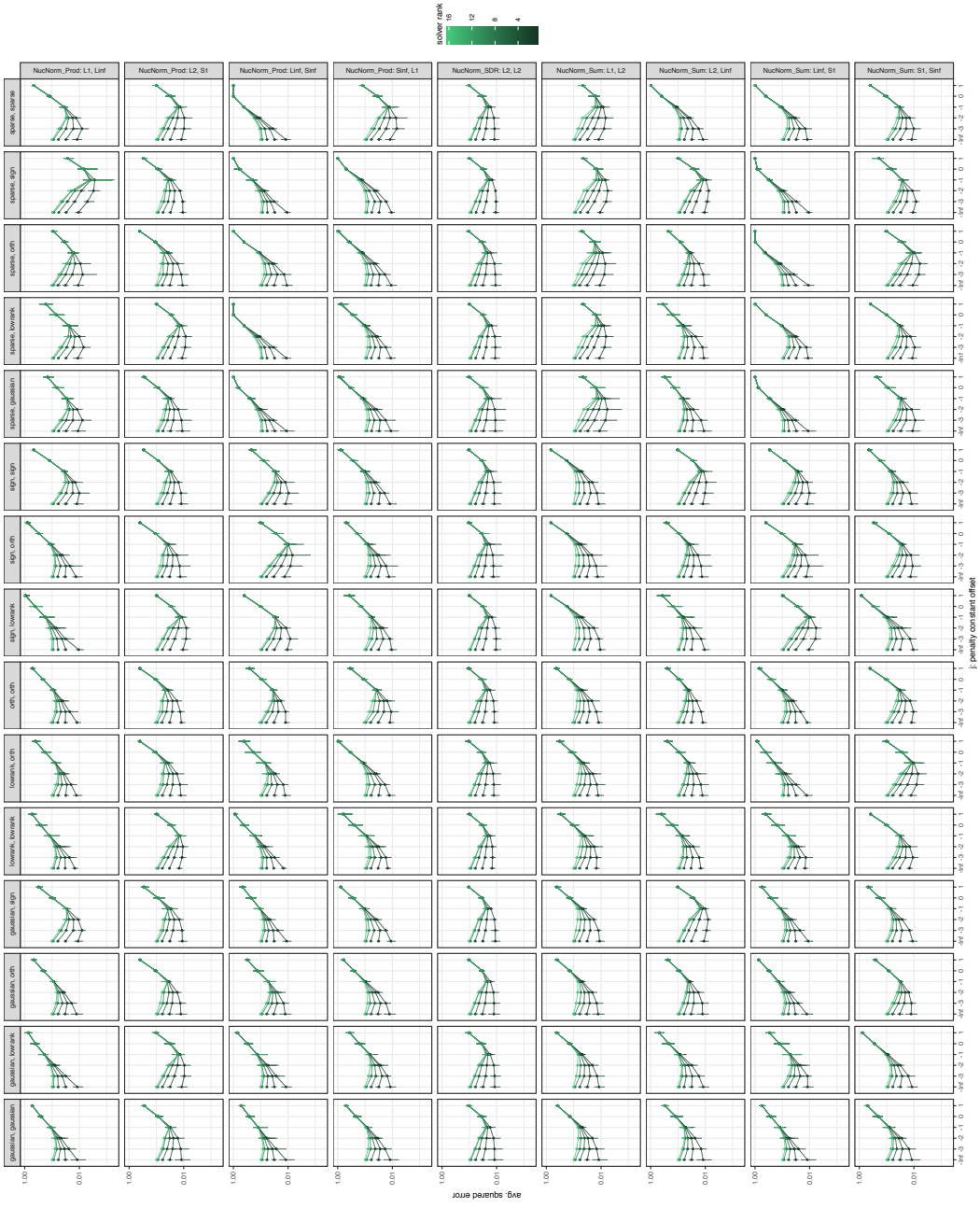


Figure B.9c: **Average error vs. penalty constant, solver rank (full).** (continued)

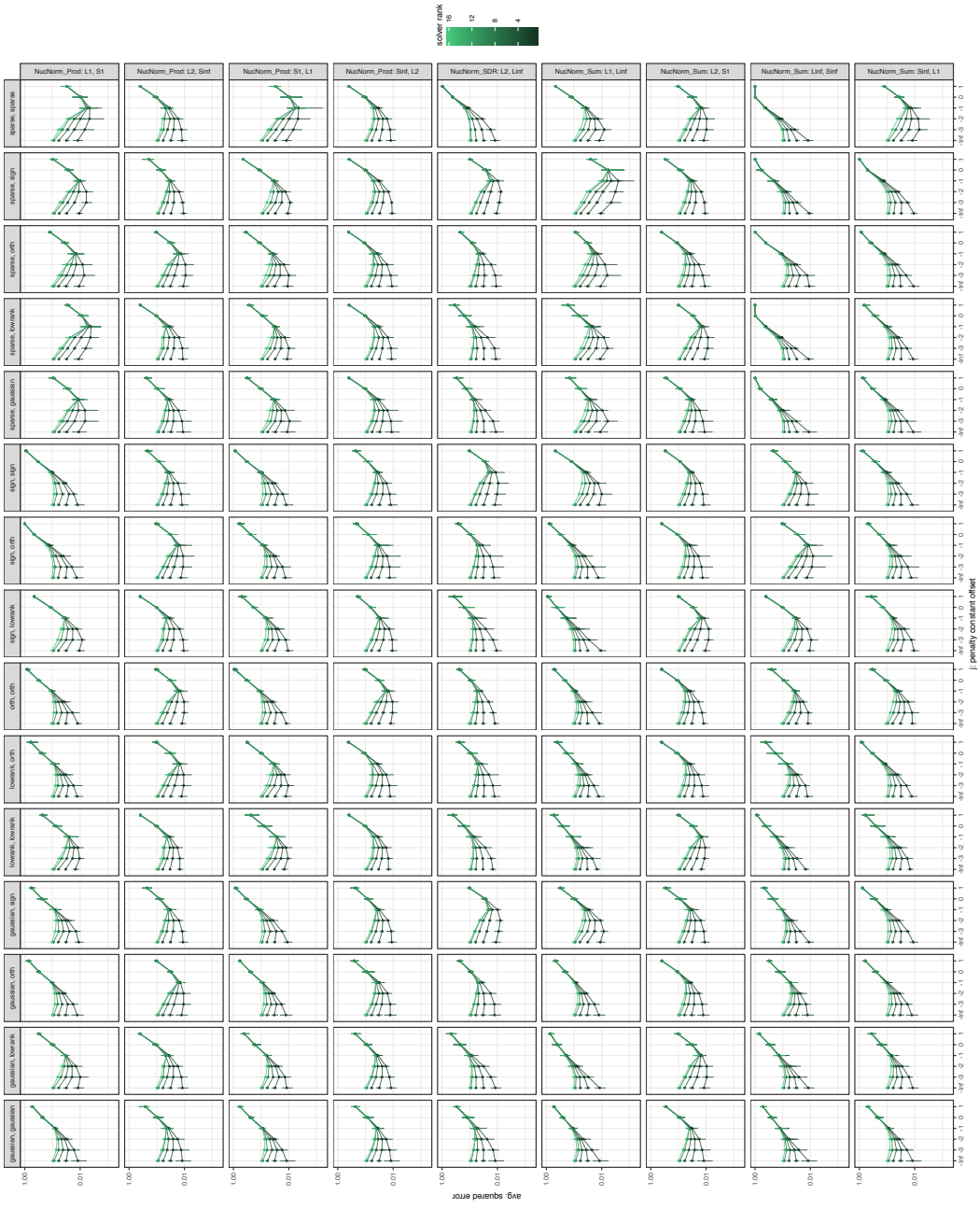


Figure B.9d: **Average error vs. penalty constant, solver rank (full).** (continued)

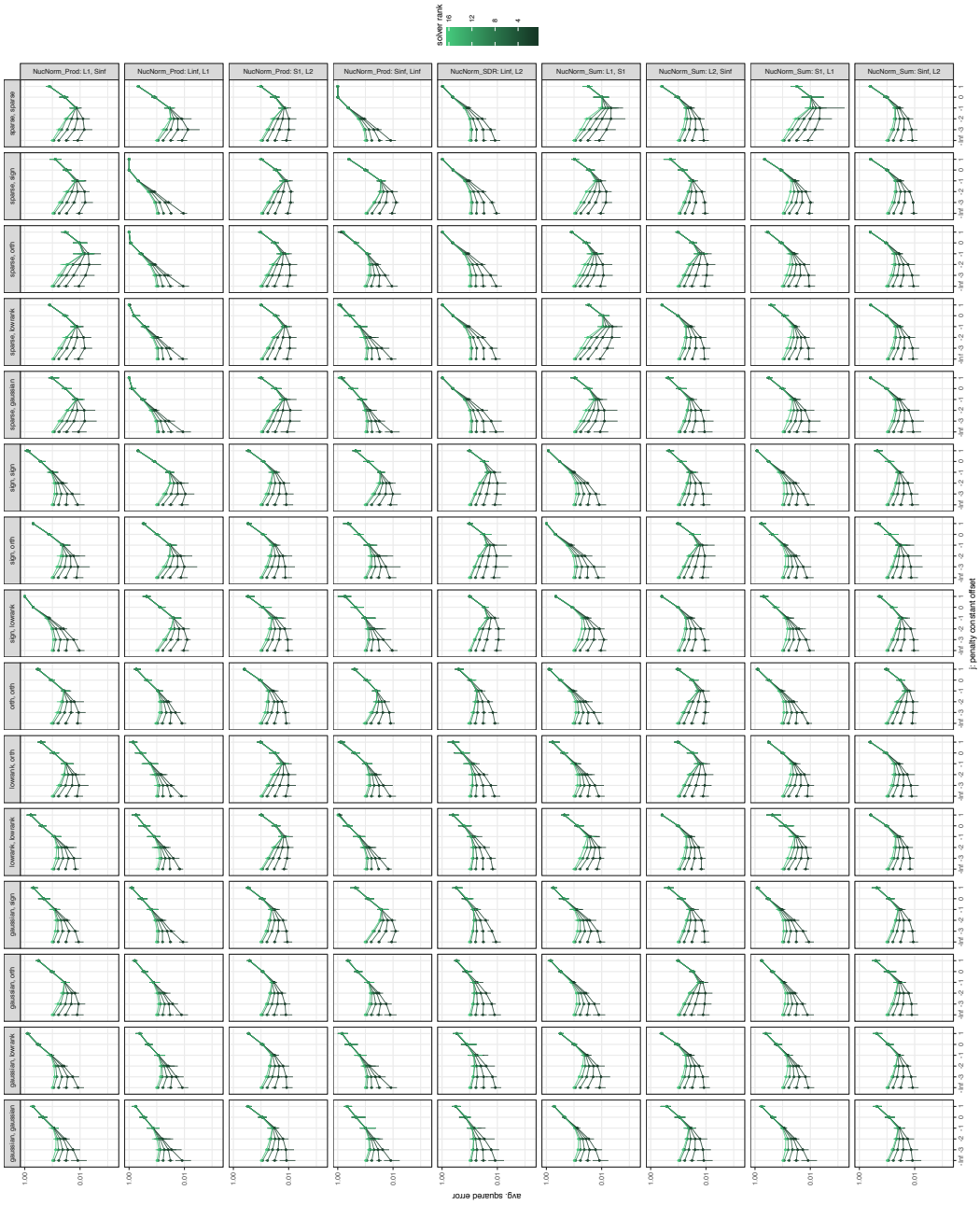


Figure B.9e: **Average error vs. penalty constant, solver rank (full).** (continued)

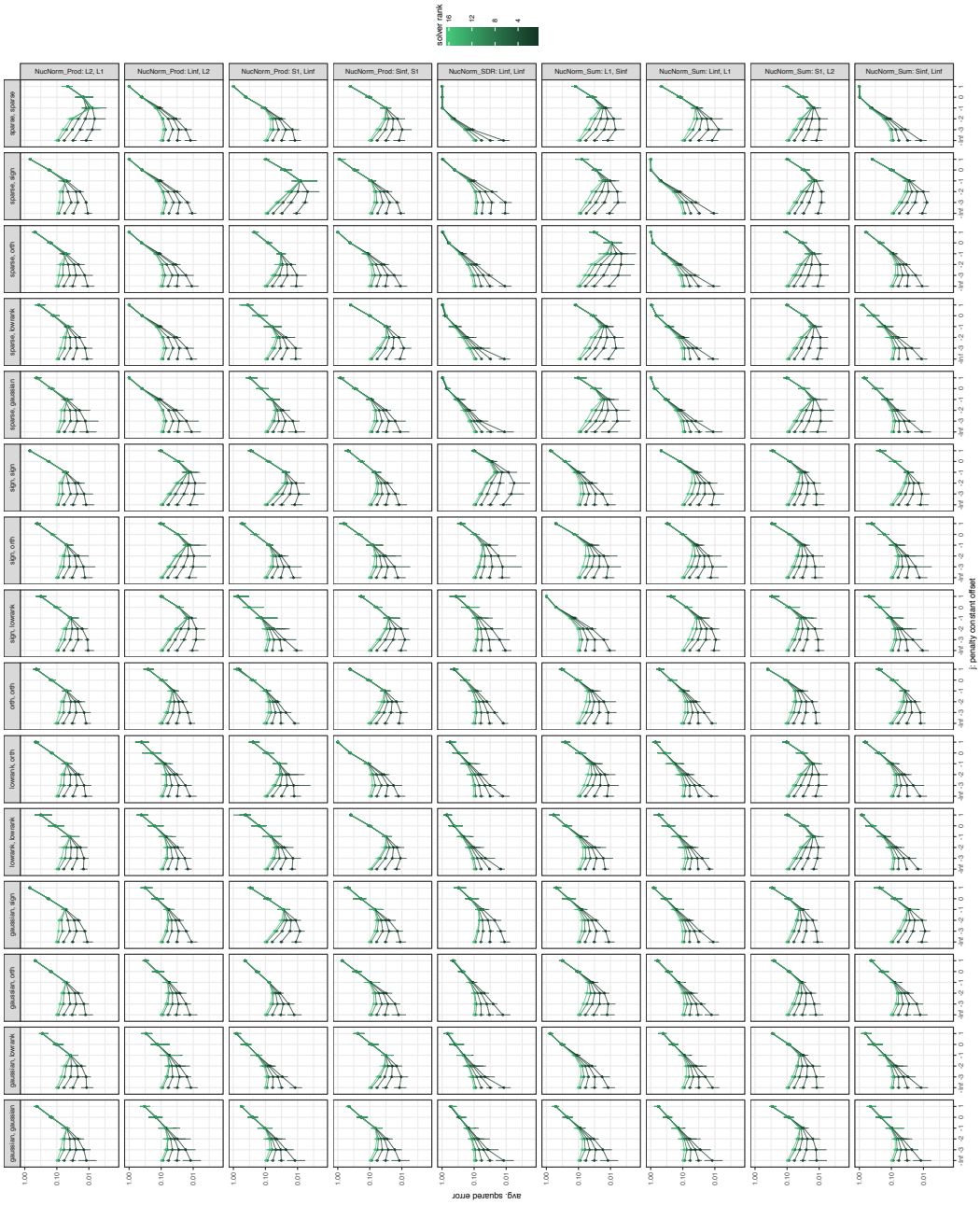


Figure B.9f: **Average error vs. penalty constant, solver rank (full).** (continued)

Operator shape: $4 \times 4 \otimes 4 \times 4$ **Rank:** 2 **SNR:** 15dB **Solver:** `altmin (NucNorm_Sum)` **Solver rank:** 16 **Tolerance:** $\epsilon = 5 \times 10^{-4}$

Factors	ℓ_1, ℓ_1	ℓ_1, ℓ_2	ℓ_1, ℓ_∞	ℓ_1, S_1	ℓ_1, S_∞	ℓ_2, ℓ_1	ℓ_2, ℓ_2	ℓ_2, ℓ_∞	ℓ_2, S_1	ℓ_2, S_∞	ℓ_∞, ℓ_1	ℓ_∞, ℓ_2	ℓ_∞, ℓ_∞	ℓ_∞, S_1	ℓ_∞, S_∞	S_1, ℓ_1	S_1, ℓ_2	S_1, ℓ_∞	S_1, S_1	S_1, S_∞	S_∞, ℓ_1	S_∞, ℓ_2	S_∞, ℓ_∞	S_∞, S_1	S_∞, S_∞
gaus, gaus	0.4	0.8	-0.2	0.6	0.5	0.9	**3.9**	1.2	1.8	2.0	0.1	1.4	-0.2	0.4	0.3	0.5	2.0	0.2	0.9	0.9	0.6	2.1	0.2	0.8	0.9
gaus, lr	0.9	1.1	0.2	2.2	0.6	1.2	4.2	1.2	**5.0**	1.9	0.3	1.6	0.1	1.4	0.3	0.9	2.3	0.3	2.9	0.9	0.8	2.3	0.3	2.6	0.8
gaus, orth	0.1	0.8	0.2	0.3	1.2	0.6	3.7	1.6	1.3	**4.5**	-0.2	1.0	-0.1	-0.0	0.8	0.2	1.7	0.4	0.6	1.9	0.2	1.7	0.5	0.5	2.0
gaus, sign	0.3	1.0	1.7	0.8	0.7	0.8	4.0	**5.7**	1.8	2.3	0.0	1.4	1.3	0.3	0.5	0.5	2.1	2.5	1.0	1.0	0.4	2.0	2.3	0.8	1.0
lr, lr	0.6	0.9	0.1	2.1	0.4	1.0	3.9	1.2	4.8	1.8	0.0	1.3	-0.1	1.1	0.0	1.9	3.9	1.2	**6.5**	1.8	0.3	1.9	-0.0	1.8	0.5
lr, orth	0.6	1.0	0.5	0.5	1.9	0.8	3.7	1.4	1.5	3.8	0.0	1.3	0.0	0.1	0.9	1.8	3.7	2.0	1.9	**6.1**	0.3	1.9	0.1	0.5	1.6
orth, orth	0.0	0.6	0.3	0.4	1.2	0.4	3.3	1.3	1.3	3.5	-0.2	1.0	0.0	-0.2	1.4	0.0	2.0	0.1	0.7	1.1	0.6	3.8	1.7	1.1	**5.3**
sign, lr	0.8	1.5	0.3	2.3	1.0	1.3	4.3	1.1	5.2	2.2	2.4	4.9	0.9	**6.8**	2.0	0.9	2.6	0.3	3.0	1.2	1.0	2.4	0.3	2.8	0.9
sign, orth	0.2	1.3	0.3	1.1	1.1	0.6	3.6	1.7	1.5	3.9	1.3	4.5	2.1	1.8	**6.2**	0.4	1.9	0.7	0.9	2.1	0.5	2.0	0.8	0.7	2.2
sign, sign	1.3	1.2	1.4	1.1	0.8	0.9	3.8	4.9	1.9	2.2	1.2	4.5	**7.6**	2.6	2.5	0.7	2.1	2.7	1.1	1.1	0.6	2.2	2.7	1.0	1.2
sparse, gaus	4.1	**5.5**	3.4	4.9	4.7	0.7	3.9	1.4	1.7	2.1	-0.1	0.6	-0.1	-0.1	-0.1	1.7	4.7	1.6	2.4	2.6	0.3	2.2	0.4	0.8	0.9
sparse, lr	4.5	5.9	3.6	**7.7**	4.6	1.1	4.1	1.4	4.7	1.7	0.0	0.9	0.0	0.6	0.0	2.3	4.5	1.7	6.3	2.3	0.5	2.4	0.4	2.2	0.7
sparse, orth	4.1	5.7	4.6	5.0	**8.1**	0.8	4.0	1.5	1.7	4.7	-0.1	0.9	-0.1	-0.1	0.1	1.6	4.4	2.0	2.3	6.7	0.3	3.1	0.4	1.2	1.8
sparse, sign	4.2	5.7	**9.6**	5.7	4.9	0.6	3.9	5.3	1.8	2.1	-0.1	0.8	0.3	-0.1	-0.1	1.8	4.5	7.4	3.0	2.7	0.5	2.3	2.3	0.8	0.8
sparse, sparse	**12.4**	5.7	2.8	8.4	4.2	5.9	3.9	0.6	5.0	2.0	2.8	0.8	0.2	1.2	0.2	8.3	4.7	1.1	6.2	2.7	4.8	2.2	0.2	2.8	0.9

Operator shape: $4 \times 4 \otimes 4 \times 4$ **Rank:** 4 **SNR:** 15dB **Solver:** `altmin (NucNorm_Sum)` **Solver rank:** 16 **Tolerance:** $\epsilon = 5 \times 10^{-4}$

Factors	ℓ_1, ℓ_1	ℓ_1, ℓ_2	ℓ_1, ℓ_∞	ℓ_1, S_1	ℓ_1, S_∞	ℓ_2, ℓ_1	ℓ_2, ℓ_2	ℓ_2, ℓ_∞	ℓ_2, S_1	ℓ_2, S_∞	ℓ_∞, ℓ_1	ℓ_∞, ℓ_2	ℓ_∞, ℓ_∞	ℓ_∞, S_1	ℓ_∞, S_∞	S_1, ℓ_1	S_1, ℓ_2	S_1, ℓ_∞	S_1, S_1	S_1, S_∞	S_∞, ℓ_1	S_∞, ℓ_2	S_∞, ℓ_∞	S_∞, S_1	S_∞, S_∞
gaus, gaus	-0.1	0.4	-0.1	0.3	0.2	0.2	**1.7**	0.7	0.8	1.1	-0.2	0.7	-0.2	0.0	0.2	0.2	0.9	0.0	0.3	0.4	0.1	1.0	0.2	0.3	0.4
gaus, lr	0.3	0.4	-0.0	0.8	0.1	0.4	1.7	0.5	**1.9**	0.7	-0.0	0.8	-0.0	0.7	-0.0	0.4	1.0	0.1	1.1	0.3	0.3	1.1	-0.0	1.1	0.3
gaus, orth	0.3	0.8	0.4	0.7	0.9	0.7	2.2	1.4	1.1	**2.4**	0.3	1.2	0.3	0.5	0.9	0.6	1.5	0.8	0.7	1.3	0.5	1.5	0.7	0.7	1.2
gaus, sign	0.3	0.6	0.7	0.5	0.5	0.7	2.0	**2.4**	1.1	1.4	0.1	1.1	1.0	0.4	0.5	0.6	1.1	1.0	0.6	0.7	0.5	1.3	1.2	0.6	0.7
lr, lr	0.6	0.6	0.0	1.2	0.4	0.6	2.1	0.8	2.4	1.1	0.0	0.8	0.0	0.6	0.1	1.1	2.4	0.5	**3.6**	0.8	0.3	1.2	0.1	0.8	0.3
lr, orth	0.3	0.7	0.6	0.6	1.1	0.5	2.0	1.4	1.0	2.0	0.1	0.8	0.1	0.1	0.4	0.8	2.1	1.4	1.0	**3.3**	0.1	1.1	0.3	0.4	0.8
orth, orth	0.3	0.7	0.3	0.6	0.8	0.6	2.1	1.3	1.1	2.2	0.2	1.3	0.6	0.4	1.3	0.4	1.1	0.4	0.5	0.9	0.8	2.2	1.2	0.8	**3.4**
sign, lr	0.4	0.6	-0.1	0.8	0.3	0.5	1.8	0.5	2.1	0.8	0.9	2.2	0.2	**3.8**	0.5	0.4	1.0	0.0	1.1	0.3	0.4	1.2	0.0	1.2	0.2
sign, orth	-0.0	0.4	-0.2	0.4	0.3	0.1	1.5	0.8	0.7	1.6	0.2	1.6	0.7	0.4	**3.3**	-0.0	0.7	0.1	0.2	0.5	-0.0	1.0	0.2	0.2	0.8
sign, sign	1.0	0.8	0.4	0.7	0.6	0.6	2.0	2.3	1.1	1.3	0.2	1.9	**4.9**	0.8	1.3	0.6	1.1	1.0	0.6	0.6	0.5	1.4	1.4	0.6	0.7
sparse, gaus	2.7	**3.8**	2.2	2.8	2.7	0.4	1.7	0.7	1.0	1.0	0.0	0.3	0.0	0.0	0.0	0.9	2.3	0.9	1.6	1.2	0.2	1.0	0.1	0.5	0.5
sparse, lr	2.8	3.3	1.4	**5.6**	2.1	0.1	1.4	0.3	1.7	0.4	-0.2	0.1	-0.2	-0.2	-0.2	0.8	1.9	0.2	3.1	0.5	-0.1	0.7	-0.2	0.5	-0.1
sparse, orth	3.1	4.1	3.0	2.8	**6.3**	0.4	2.0	1.2	0.8	2.3	0.0	0.5	0.0	0.0	0.1	0.9	2.6	1.3	1.1	3.6	0.3	1.4	0.5	0.5	1.1
sparse, sign	2.6	3.7	**7.2**	2.9	2.7	0.4	1.7	1.7	0.8	1.1	0.5	0.3	-0.0	-0.0	-0.0	0.8	2.1	3.8	1.4	1.4	0.3	1.0	0.8	0.3	0.5
sparse, sparse	**10.0**	3.9	1.5	6.4	2.6	4.1	1.9	0.3	2.6	1.3	1.5	0.3	0.1	0.3	0.1	6.3	2.4	0.2	3.6	1.3	3.1	1.1	0.1	1.4	0.4

Table B.4: **Denoising gains,** $4 \times 4 \otimes 4 \times 4$**.** These tables show the gains in dB (5.9) for denoising $4 \times 4 \otimes 4 \times 4$ operators of various ranks at all tested combinations of factor structure and nuclear norm. Bold numbers indicate the highest value(s) in each row (i.e., the nuclear norm that empirically denoises the factor structure best). All results use the alternating minimization solver with 16 dyads, a convergence tolerance $\epsilon = 5 \times 10^{-4}$ in (5.10), and SNR (5.7) of 15dB. (Key: gaus = Gaussian, lr = low-rank, orth = orthogonal.)

Operator shape: $16 \times 16 \otimes 16 \times 16$ **Rank:** 2 **SNR:** 10dB **Solver:** `altmin (NucNorm_Sum)` **Solver rank:** 16 **Tolerance:** $\epsilon = 1 \times 10^{-3}$

Factors	ℓ_1, ℓ_1	ℓ_1, ℓ_2	ℓ_1, ℓ_∞	ℓ_1, S_1	ℓ_1, S_∞	ℓ_2, ℓ_1	ℓ_2, ℓ_2	ℓ_2, ℓ_∞	ℓ_2, S_1	ℓ_2, S_∞	ℓ_∞, ℓ_1	ℓ_∞, ℓ_2	ℓ_∞, ℓ_∞	ℓ_∞, S_1	ℓ_∞, S_∞	S_1, ℓ_1	S_1, ℓ_2	S_1, ℓ_∞	S_1, S_1	S_1, S_∞	S_∞, ℓ_1	S_∞, ℓ_2	S_∞, ℓ_∞	S_∞, S_1	S_∞, S_∞
gaus, gaus	7.0	9.9	7.0	7.9	7.6	9.2	**12.2**	8.1	9.8	**12.2**	7.0	8.0	7.0	7.0	7.1	7.6	11.0	7.1	8.2	9.2	7.4	11.0	7.1	9.0	8.1
gaus, lr	7.0	9.9	7.0	12.0	7.0	9.3	12.3	7.9	15.6	9.6	7.0	8.1	7.0	10.5	7.0	7.9	11.0	7.0	**16.2**	7.4	7.6	10.9	7.0	14.1	7.0
gaus, orth	7.0	9.9	7.0	7.4	10.0	9.1	12.1	9.0	9.6	**16.6**	7.0	8.0	7.0	7.0	8.7	7.6	10.9	8.0	7.8	11.4	7.0	10.8	7.7	8.6	10.4
gaus, sign	7.0	9.9	10.8	7.9	7.6	10.0	12.2	**15.0**	9.8	12.1	7.0	8.0	8.0	7.0	7.0	8.2	11.0	11.8	8.2	9.2	7.9	11.2	9.9	9.0	8.1
lr, lr	6.9	9.7	6.9	11.8	6.9	9.1	12.0	8.0	14.2	9.6	6.9	7.7	6.9	9.8	6.9	11.2	15.7	10.3	**21.0**	16.1	6.9	9.6	6.9	12.1	6.9
lr, orth	7.0	9.7	7.0	7.5	9.8	9.1	12.0	8.3	9.4	14.8	7.0	7.8	7.0	7.0	7.5	11.7	15.4	11.1	12.3	**19.0**	7.0	10.0	7.0	7.0	8.9
orth, orth	7.0	10.0	7.0	7.3	9.9	9.2	12.2	8.5	9.6	**15.1**	7.0	8.5	7.0	7.0	9.1	7.2	11.1	7.1	7.8	10.8	9.4	13.2	8.4	10.1	13.9
sign, lr	7.0	10.9	7.0	13.2	7.0	9.2	12.2	7.8	14.1	9.6	10.4	12.5	7.0	**17.4**	8.0	7.8	11.0	7.0	16.5	7.4	7.6	11.0	7.0	14.2	7.0
sign, orth	7.0	10.9	7.0	8.7	10.3	9.2	12.3	8.3	9.6	15.0	10.4	12.5	7.7	10.4	**15.7**	7.6	11.1	7.3	8.2	10.9	7.3	10.9	7.3	8.7	9.6
sign, sign	7.0	10.8	12.1	8.8	8.1	9.9	12.2	**14.4**	9.8	12.2	12.6	12.5	12.4	10.8	9.5	8.3	10.9	11.7	8.2	9.1	7.8	11.0	10.4	8.9	8.2
sparse, gaus	14.9	20.2	14.1	14.4	**20.5**	9.2	15.3	8.7	10.2	11.9	7.0	7.0	7.0	7.0	7.0	12.4	17.9	11.9	13.7	18.2	7.0	9.7	7.0	8.0	7.7
sparse, lr	14.7	20.2	13.6	**22.4**	19.2	9.2	15.3	7.9	14.3	9.6	7.0	7.0	7.0	7.0	7.0	12.4	18.7	9.7	21.7	16.6	7.0	10.2	7.0	12.8	7.0
sparse, orth	15.1	19.7	13.8	14.4	**20.6**	9.1	15.1	8.5	9.6	14.8	7.0	7.0	7.0	7.0	7.0	12.5	17.7	11.3	12.8	19.1	7.0	10.5	7.0	8.3	9.5
sparse, sign	16.7	19.7	**22.9**	14.3	20.3	9.9	15.1	14.5	9.9	12.1	7.0	7.0	7.0	7.0	7.0	12.4	17.6	21.7	13.3	17.9	7.0	10.2	10.0	8.0	8.2
sparse, sparse	**29.6**	20.8	11.8	24.6	19.5	20.6	15.3	7.0	19.4	9.6	12.0	7.0	7.0	7.7	7.0	22.9	18.8	7.4	22.4	16.2	17.7	10.0	7.0	14.2	7.0

Operator shape: $16 \times 16 \otimes 16 \times 16$ **Rank:** 4 **SNR:** 10dB **Solver:** `altmin (NucNorm_Sum)` **Solver rank:** 16 **Tolerance:** $\epsilon = 1 \times 10^{-3}$

Factors	ℓ_1, ℓ_1	ℓ_1, ℓ_2	ℓ_1, ℓ_∞	ℓ_1, S_1	ℓ_1, S_∞	ℓ_2, ℓ_1	ℓ_2, ℓ_2	ℓ_2, ℓ_∞	ℓ_2, S_1	ℓ_2, S_∞	ℓ_∞, ℓ_1	ℓ_∞, ℓ_2	ℓ_∞, ℓ_∞	ℓ_∞, S_1	ℓ_∞, S_∞	S_1, ℓ_1	S_1, ℓ_2	S_1, ℓ_∞	S_1, S_1	S_1, S_∞	S_∞, ℓ_1	S_∞, ℓ_2	S_∞, ℓ_∞	S_∞, S_1	S_∞, S_∞
gaus, gaus	7.2	9.6	7.2	7.2	7.2	9.3	**10.4**	8.0	9.2	9.5	7.2	7.8	7.2	7.2	7.3	7.2	9.9	7.2	7.2	8.3	7.2	9.3	7.2	8.1	7.8
gaus, lr	7.2	9.7	7.2	11.4	7.2	9.1	10.3	7.4	**13.7**	8.6	7.2	7.6	7.2	9.7	7.2	7.2	9.8	7.2	11.0	7.2	7.2	9.2	7.2	12.3	7.2
gaus, orth	7.2	9.7	7.2	7.2	8.4	9.4	10.4	8.0	9.0	**10.7**	7.2	7.9	7.2	7.2	8.3	7.2	9.8	7.3	7.2	9.7	7.2	9.3	7.4	7.9	8.9
gaus, sign	7.2	9.7	8.9	7.2	7.2	9.9	10.4	**12.9**	9.2	9.3	7.2	7.9	8.3	7.2	7.5	8.5	9.9	10.7	7.2	8.3	7.4	9.3	9.5	8.1	7.9
lr, lr	7.2	9.8	7.2	11.6	7.2	9.2	10.4	7.5	14.0	8.5	7.2	7.2	7.2	8.3	7.2	11.5	12.9	8.9	**19.5**	11.0	7.2	8.4	7.2	10.7	7.2
lr, orth	7.2	9.7	7.2	7.2	8.9	9.3	10.5	8.2	9.0	10.6	7.2	7.5	7.2	7.2	7.4	11.6	13.0	10.3	11.5	**17.8**	7.2	8.1	7.2	7.2	7.7
orth, orth	7.2	9.8	7.2	7.2	8.4	9.2	10.4	8.2	9.0	10.7	7.2	7.8	7.2	7.2	8.4	7.2	9.9	7.2	7.2	9.2	8.0	12.0	7.6	8.8	**13.8**
sign, lr	7.2	10.1	7.2	11.5	7.2	9.2	10.4	7.5	13.7	8.6	9.0	12.7	7.2	**14.8**	7.9	7.2	9.9	7.2	11.3	7.2	7.2	9.3	7.2	11.9	7.2
sign, orth	7.2	10.1	7.2	8.5	8.2	9.3	10.3	8.0	9.0	10.4	8.4	13.1	7.7	9.5	**13.9**	7.2	9.8	7.2	7.2	9.5	7.2	9.3	7.3	8.0	9.1
sign, sign	7.2	10.2	7.6	8.5	7.6	10.0	10.4	13.8	9.2	9.6	7.2	12.9	**15.9**	10.1	9.3	8.3	9.9	10.6	7.2	8.5	7.2	9.4	9.4	8.1	8.0
sparse, gaus	14.3	**15.9**	12.5	13.1	14.3	9.2	10.3	8.0	9.3	9.4	7.2	7.2	7.2	7.2	7.2	11.8	13.7	9.8	12.8	12.1	7.2	8.5	7.2	7.9	7.8
sparse, lr	14.5	16.3	12.2	**19.0**	14.0	9.3	10.4	7.7	14.2	8.6	7.3	7.3	7.3	7.3	7.3	11.8	13.6	8.9	18.7	10.9	7.3	9.2	7.3	11.1	7.3
sparse, orth	14.6	**16.1**	12.5	13.4	15.6	9.3	10.4	8.1	9.1	10.7	7.3	7.3	7.3	7.3	7.3	11.9	14.3	10.5	12.8	16.5	7.3	8.6	7.3	8.2	8.5
sparse, sign	15.1	16.0	**19.8**	13.4	14.1	10.0	10.4	13.3	9.3	9.4	7.2	7.2	7.2	7.2	7.2	12.1	14.3	18.2	12.7	12.0	8.0	8.5	8.8	8.2	7.9
sparse, sparse	**22.5**	17.7	10.1	19.9	16.1	15.9	12.3	7.2	14.3	9.1	9.5	7.2	7.2	7.2	7.2	20.3	14.6	7.2	18.8	11.7	14.9	9.0	7.2	11.9	7.2

Table B.9a: **Denoising gains,** $16 \times 16 \otimes 16 \times 16$**.** These tables show the gains in dB (5.9) for denoising $16 \times 16 \otimes 16 \times 16$ operators of various ranks at all tested combinations of factor structure and nuclear norm. Bold numbers indicate the highest value(s) in each row (i.e., the nuclear norm that empirically denoises the factor structure best). All results use the alternating minimization solver with 16 dyads, a convergence tolerance $\epsilon = 1 \times 10^{-3}$ in (5.10), and SNR (5.7) of 10dB. (Key: gaus = Gaussian, lr = low-rank, orth = orthogonal.)

Operator shape: $16 \times 16 \otimes 16 \times 16$ **Rank:** 8 **SNR:** 10dB **Solver:** `altmin (NucNorm_Sum)` **Solver rank:** 16 **Tolerance:** $\epsilon = 1 \times 10^{-3}$

Factors	ℓ_1, ℓ_1	ℓ_1, ℓ_2	ℓ_1, ℓ_∞	ℓ_1, S_1	ℓ_1, S_∞	ℓ_2, ℓ_1	ℓ_2, ℓ_2	ℓ_2, ℓ_∞	ℓ_2, S_1	ℓ_2, S_∞	ℓ_∞, ℓ_1	ℓ_∞, ℓ_2	ℓ_∞, ℓ_∞	ℓ_∞, S_1	ℓ_∞, S_∞	S_1, ℓ_1	S_1, ℓ_2	S_1, ℓ_∞	S_1, S_1	S_1, S_∞	S_∞, ℓ_1	S_∞, ℓ_2	S_∞, ℓ_∞	S_∞, S_1	S_∞, S_∞
gaus, gaus	7.7	7.7	7.7	7.7	7.7	7.7	**9.2**	7.7	8.8	8.2	7.7	7.7	7.7	7.7	7.7	7.7	9.1	7.7	7.7	7.7	7.7	8.3	7.7	7.7	7.7
gaus, lr	7.7	7.7	7.7	10.3	7.7	7.7	9.1	7.7	**11.5**	7.7	7.7	7.7	7.7	8.7	7.7	7.7	9.0	7.7	10.6	7.7	7.7	8.2	7.7	10.8	7.7
gaus, orth	7.7	7.7	7.7	7.7	7.7	7.7	**9.1**	7.7	8.7	8.5	7.7	7.8	7.7	7.7	7.8	7.7	9.0	7.7	7.7	8.1	7.7	8.2	7.7	7.7	8.1
gaus, sign	7.7	7.7	7.7	7.7	7.7	7.7	9.1	**10.1**	8.7	8.2	7.7	7.8	7.7	7.7	7.7	7.7	9.0	7.9	7.7	7.7	7.7	8.2	7.7	7.7	7.8
lr, lr	7.7	7.7	7.7	10.5	7.7	7.7	9.1	7.7	11.8	7.7	7.7	7.7	7.7	7.7	7.7	10.1	11.3	7.7	**16.4**	9.2	7.7	8.0	7.7	9.0	7.7
lr, orth	7.7	7.7	7.7	7.7	7.7	7.7	9.1	7.7	8.7	8.5	7.7	7.7	7.7	7.7	7.7	9.9	11.4	9.2	9.6	**15.0**	7.7	8.0	7.7	7.7	7.7
orth, orth	7.6	7.6	7.6	7.6	7.6	7.6	9.0	7.6	8.6	8.4	7.6	7.7	7.6	7.6	7.8	7.6	9.0	7.6	7.6	7.7	7.6	9.3	7.6	7.6	**11.6**
sign, lr	7.6	7.6	7.6	9.6	7.6	7.6	9.0	7.6	11.6	7.6	7.6	9.8	7.6	**14.5**	7.6	7.6	8.9	7.6	10.5	7.6	7.6	8.3	7.6	10.9	7.6
sign, orth	7.7	7.7	7.7	7.7	7.7	7.7	9.1	7.7	8.7	8.5	7.7	9.8	7.7	7.7	**12.6**	7.7	9.0	7.7	7.7	7.9	7.7	8.2	7.7	7.7	8.0
sign, sign	7.7	7.7	7.7	7.7	7.7	7.7	9.2	9.2	8.8	8.2	7.7	8.9	**13.9**	8.1	8.2	7.7	9.0	8.1	7.7	7.7	7.7	8.2	8.0	7.7	7.8
sparse, gaus	12.1	**13.4**	10.8	12.4	12.1	7.8	9.1	7.8	8.9	8.2	7.8	7.8	7.8	7.8	7.8	10.5	11.5	9.3	11.0	9.7	7.8	8.1	7.8	7.8	7.9
sparse, lr	12.6	13.1	10.3	**18.1**	11.3	7.7	9.1	7.7	12.0	7.7	7.7	7.7	7.7	7.7	7.7	10.5	11.4	7.7	16.6	9.6	7.7	8.2	7.7	9.2	7.7
sparse, orth	12.2	14.0	10.6	12.4	**14.3**	7.8	9.1	7.8	8.8	8.5	7.8	7.8	7.8	7.8	7.8	10.2	11.5	9.5	10.8	14.2	7.8	8.2	7.8	7.8	8.4
sparse, sign	12.3	14.0	**16.5**	12.7	12.0	7.8	9.1	9.1	8.9	8.1	7.8	7.8	7.8	7.8	7.8	10.1	11.6	15.9	11.1	10.3	7.8	8.1	8.3	7.8	8.0
sparse, sparse	**21.5**	13.2	7.8	19.6	11.7	13.9	9.1	7.8	12.3	8.6	7.8	7.8	7.8	7.8	7.8	18.9	11.5	7.8	17.4	9.5	12.6	8.6	7.8	10.2	7.8

Table B.9b: **Denoising gains,** $16 \times 16 \otimes 16 \times 16$**.** (continued)

Appendix C

Hyperspectral imaging experiments

This appendix provides additional details for the numerical experiments on hyperspectral image denoising in Chapter 6.

C.1 The USGS Digital Spectral Library

We use the USGS Digital Spectral Library [Cla+07] to generate the endmembers for our hyperspectral image test. The raw data contains spectra with some missing data, and so we apply a simple resampling and smoothing procedure to ensure that all members of the library have reflectance data at 224 evenly-spaced wavelengths between 0.4μm and 2.5μm.

For each material, the database includes a column vector $\boldsymbol{x}$ of wavelengths and a corresponding column vector $\boldsymbol{y}$ of reflectance values. We perform the following procedure to each item in the library:

1. For any entry y_i that is `NaN`, set $y_i = 0$.
2. Select the indices of $\boldsymbol{x}$ that are not `NaN`.
3. Create a scipy `UnivariateSpline` with the subsets of $\boldsymbol{x}$ and $\boldsymbol{y}$ corresponding to those indices. Set the parameters $k = 3$ and $s = 10$.
4. Create a new $\boldsymbol{x}$ with 224 linearly-spaced points between 0.4 and 2.5 (inclusive).
5. Create a new $\boldsymbol{y}$ by applying the spline to the resampled $\boldsymbol{x}$.

C.2 Generating the test image

The test HSI is a $75 \times 75 \times 224$ hyperspectral image generated through a linear mixing model according to the procedure of Iordache et al. [IBP11]. The image consists of an equally-spaced 5×5 grid of patches that are each 5×5 pixels large. To fill the patches we:

1. Choose 5 spectra from our spectral library (Section C.1).
2. For each patch in the ith row of the grid, choose a combination (without replacement) from the set of all combinations of i endmembers.
3. Fill in the spectra of the patch as an equal combination of the i chosen endmembers.
4. For $i = 5$, set all patches to be an equal combination of all endmembers.

The following Python code shows how to generate abundance matrices that generate such a test image:

```
cols = [5, 20, 35, 50, 65]
rows = cols

A = np.zeros((75, 75, 5))
blk = np.ones((5,5))
for i in range(5):
    combos = list(itertools.combinations(range(5), i+1))
    if len(combos) == 1:
        combos = combos*5
    combo_ixs = np.random.choice(len(combos), 5, replace=False)
    for j in range(5):
        for k in range(i+1):
            A[rows[i]:rows[i]+5, cols[j]:cols[j]+5, combos[
                combo_ixs[j]][k]] = 1.0/(i+1)
```

C.3 The Spa+Lr method

We implement the Spa+Lr method of Zhao et al. [ZY15] in Python using *scikit-learn* [Ped+11] for dictionary learning and image manipulation routines. For the sake of experimentation, assume that we have access to the true image $\mathcal{A}^\natural$, and use this to generate a dictionary that is then used in the recovery. This

saves time in repeated experimentation and certainly does not diminish the performance of Spa+Lr. Assume that $\mathcal{B} \in \mathbb{O}^{m \times n \otimes p}$ is a noisy HSI, and σ is an estimate of the standard deviation of the noise. We proceed as follows:

1. Reshape $\mathcal{A}^\natural$, $\mathcal{B}$ into matrices $\boldsymbol{A}^\natural$, $\boldsymbol{B} \in \mathbb{M}^{mn \times p}$.
2. Set $\gamma = (30/\sigma)/(10^{2.5})$. (This achieved better performance than $\gamma = 30/\sigma$ from the paper.)
3. Set $\lambda = 100$.
4. Set $\mu = \lambda\sigma \cdot \max\{\sqrt{mn}, \sqrt{p}\}/6.5$.
5. Fix a patch size of (8, 8), and extract overlapping patches from the true image $\boldsymbol{A}^\natural$. De-mean and normalize the patches.
6. Use the `MiniBatchDictionaryLearning` class of *scikit-learn* to learn a dictionary from the patches.
7. Set $\boldsymbol{X} = \boldsymbol{B}$.
8. Perform $k_{\max}$ iterations of the following:
    a) Extract overlapping patches from $\boldsymbol{X}$. De-mean and normalize the patches.
    b) Use the dictionary object to perform sparse coding on the patches.
    c) Reconstruct the patches from the obtained weights, and re-add the mean and normalization.
    d) Reconstruct the matrix $\boldsymbol{X}$ from the patches.
    e) Compute the SVD of $\boldsymbol{X} = \boldsymbol{W}\boldsymbol{S}\boldsymbol{V}^\mathsf{t}$, and soft-threshold the diagonal of $\boldsymbol{S}$. That is, set $s_{ii} = \operatorname{sgn}(s_{ii}) \cdot \max\{|s_{ii}| - (\mu/\lambda), 0\}$.
    f) Set $\boldsymbol{U} = \boldsymbol{W}\boldsymbol{S}\boldsymbol{V}^\mathsf{t}$.
    g) Set $\boldsymbol{X} = (\gamma\boldsymbol{B} + \rho\boldsymbol{X} + \lambda\boldsymbol{U})/(\gamma + \rho + \lambda)$, where $\rho = mnp$. (This is a simplification of the true averaging procedure.)

C.4 The numerical experiment

This section describes the procedure for the HSI experiment in Chapter 6.

C.4.1 Nuclear norm solver

To denoise with nuclear norms, we use the following procedure:

1. Load the $75 \times 75 \times 224$ test HSI (Section C.2).
2. Compute a noise level σ to reach the desired signal-to-noise ratio (SNR).
3. Generate a $75 \times 75 \times 224$ random operator with independent $\textsc{normal}(0, \sigma)$ entries.
4. Corrupt the test image with the noise array.
5. Compute the penalty constant λ_0 using the procedure in Section B.2 and apply offset j to obtain $\lambda = \lambda_0 \cdot 2^j$.
6. Generate a `Problem` object and call the solver with desired options.
7. Compute the relative error and RSDR of the output.

C.4.2 Truncated dyadic SVD

To denoise with the truncated dyadic SVD, we:

1. Load the $75 \times 75 \times 224$ test HSI (Section C.2).
2. Compute a noise level σ to reach the desired signal-to-noise ratio (SNR).
3. Generate a $75 \times 75 \times 224$ random operator with independent $\textsc{normal}(0, \sigma)$ entries.
4. Corrupt the test image with the noise array.
5. Compute the dyadic SVD (3.7) of the noisy observations and truncate to the top r dyads.
6. Compute the relative error and RSDR of the output.

C.4.3 Spa+Lr

To denoise with Spa+Lr (Section C.3), we:

1. Load the $75 \times 75 \times 224$ test HSI (Section C.2).
2. Compute a noise level σ to reach the desired signal-to-noise ratio (SNR).
3. Generate a $75 \times 75 \times 224$ random operator with independent $\textsc{normal}(0, \sigma)$ entries.
4. Corrupt the test image with the noise array.
5. Run the Spa+Lr procedure.
6. Compute the relative error and RSDR of the output.

C.4.4 Parameter choices

For all experiments, fix SNR at 10dB.

Nuclear norm solver. Set the relative convergence tolerance $\epsilon = 10^{-3}$ and the maximal number of iterations to 10. Use the non-quadratic option (`noquad`) on `altminsolve` and `NucNorm_Prod` for all the nuclear norm objects. For solver ranks 5 and 10, test each of the following pairs of nuclear norm and offsets j:

- $\ell_1 \otimes \ell_2$ with $j = -10, \ldots, -6$,
- $\ell_1 \otimes \mathrm{TV}$ with $j = -13, \ldots, -8$,
- $S_1 \otimes \ell_2$ with $j = -9, \ldots, -6$,
- $S_1 \otimes \mathrm{TV}$ with $j = -9, \ldots, -5$,
- $\mathrm{TV} \otimes \ell_2$ with $j = -13, \ldots, -9$,
- $\mathrm{TV} \otimes \mathrm{TV}$ with $j = -13, \ldots, -9$.

Repeat for 10 trials, and for each nuclear norm return the best average gain[1] over all tested offsets j.

[1] Recall that gain = RSDR − SNR. See Section 5.4.2.

Dyad SVD. Solve the denoising problem with the dyadic SVD for $r = 5, 10$. Repeat 10 times and compute the average gain.

Spa+Lr Run the Spa+Lr procedure 10 times and compute the average gain.

Appendix D

Self-calibration experiments

This appendix details the procedures for the self-calibration experiments described in Chapter 7.

D.1 Single snapshot

This section describes the single snapshot experiments in Section 7.4.1.

D.1.1 Procedure

Repeat the following for desired parameter choices:

1. Generate a parameter vector $\boldsymbol{x} \in \mathbb{R}^m$ with independent standard normal entries.
2. Generate a sparse vector $\boldsymbol{y} \in \mathbb{R}^{256}$.
   a) Choose s indices uniformly at random.
   b) Fill those s entries of $\boldsymbol{y}$ with independent standard normal variates.
3. Construct the matrix $\boldsymbol{S} \in \mathbb{M}^{128\times m}$ from the first m columns of the normalized DCT matrix, and generate a matrix $\boldsymbol{T} \in \mathbb{M}^{128\times 256}$ with standard normal entries. (Performed internally by the `SelfCalibMeasurement` object.)
4. Generate a noise vector $\boldsymbol{z} \in \mathbb{R}^{128}$ with standard normal entries.

5. Compute σ for a target SNR using the procedure from the denoising experiments (Section B.1.3).
6. Generate measurements $\boldsymbol{b} \in \mathbb{R}^{128}$ by computing $\boldsymbol{b} = \operatorname{diag}(\boldsymbol{S}\boldsymbol{x})\boldsymbol{T}\boldsymbol{y} + \sigma\boldsymbol{z}$.
7. Compute the penalty constant λ_0 using the procedure in Section B.2 and apply offset j to obtain $\lambda = \lambda_0 \cdot 2^j$.
8. Generate a `Problem` object and call the solver with desired options.
9. Project the output $\widehat{\mathcal{A}}$ to a rank-1 operator, and compute relative error/RSDR.

Note that the noiseless case takes $\sigma = 0$, and the penalty constant λ is not used.

D.1.2 Parameter choices

For all experiments we vary m and s independently through $1, 2, \ldots, 15$.

Convex solver. Use the convex solver `matsolve` with the ℓ_1 norm and the $\ell_2 \otimes \ell_1$ nuclear norm at SNR 0dB, 5dB, 10dB, 20dB, and ∞dB (noiseless). Repeat for 10 trials.

Alternating minimization solver. Use the alternating minimization solver `altminsolve` with convergence tolerance $\epsilon = 10^{-3}$ and a maximum of 10 outer iterations. Fix SNR at 15dB for all experiments. Test all combinations of the following parameters:

- **Regularizers:** ℓ_1 norm and $\ell_2 \otimes \ell_1$ nuclear norm.
- **Penalty constant offsets:** $j = -2, -1, 0, 1$
- **Solver ranks:** 1, 2, 4, 16

Repeat for 10 trials.

D.2 Multiple snapshot

This section describes the multiple snapshot experiments in Section 7.4.2.

D.2.1 Procedure

Repeat the following for desired parameter choices:

1. Generate a parameter vector $\boldsymbol{x} \in \mathbb{R}^m$ with independent standard normal entries.
2. Generate signals $\boldsymbol{Y} \in \mathbb{M}^{256\times q}$ with one of the following procedures:
   - **Independent snapshots:** For each column of $\boldsymbol{Y}$, choose s indices uniformly at random and fill those s entries of $\boldsymbol{y}$ with independent standard normal variates.
   - **Simultaneous sparsity:** Choose s indices out of $1, \ldots, 256$ uniformly at random. For each column of $\boldsymbol{Y}$, fill those s indices with independent standard normal variates.
   - **Identical snapshots:** Create an s-sparse vector with independent standard normal entires, and fill all columns of $\boldsymbol{Y}$ identically with this vector.
3. Construct the matrix $\boldsymbol{S} \in \mathbb{M}^{128\times m}$ from the first m columns of the normalized DCT matrix, and generate a matrix $\boldsymbol{T} \in \mathbb{M}^{128\times 256}$ with standard normal entries. (Performed internally by the `SelfCalibMeasurement` object.)
4. Generate a noise matrix $\boldsymbol{Z} \in \mathbb{M}^{128\times q}$ with standard normal entries.
5. Compute σ for a target SNR using the procedure from the denoising experiments (Section B.1.3).
6. Generate measurements $\boldsymbol{B} \in \mathbb{M}^{128\times q}$ by computing $\boldsymbol{B} = \operatorname{diag}(\boldsymbol{S}\boldsymbol{x})\boldsymbol{T}\boldsymbol{y} + \sigma\boldsymbol{Z}$.
7. Compute the penalty constant λ_0 using the procedure in Section B.2 and apply offset j to obtain $\lambda = \lambda_0 \cdot 2^j$.
8. Generate a `Problem` object and call the solver with desired options.
9. Project the output $\widehat{\mathcal{A}}$ to a rank-1 operator, and compute relative error/RSDR.

D.2.2 Parameter choices

For all experiments we use the alternating minimization solver and vary m and s independently through 1, 2, ..., 15. Fix the SNR at 15dB, the convergence tolerance at $\epsilon = 10^{-3}$, the maximum number of outer iterations at 10, and the solver rank at 4.

Initial experiment. Set the number of snapshots to $q = 8$. Test all combinations of the following parameters:

- **Regularizers:** $\ell_2 \otimes \ell_1$, $\ell_2 \otimes (\ell_1 \otimes \ell_2)$, and $\ell_2 \otimes (\ell_1 \otimes \ell_\infty)$ nuclear norms
- **Penalty constant offsets:** $j = -4, -3, -2, -1, 0$
- **Signal model:** Independent snapshots, simultaneously sparsity, identical snapshots.

Repeat for 10 trials.

Varying the number of snapshots. Consider only the following pairs of signal model and nuclear norm:

- Independent snapshots with the $\ell_2 \otimes \ell_1$ nuclear norm,
- Simultaneously sparse snapshots with the $\ell_2 \otimes (\ell_1 \otimes \ell_2)$ nuclear norm, and
- Identical snapshots with the $\ell_2 \otimes (\ell_1 \otimes \ell_\infty)$ nuclear norm.

For each of these pairs test all combinations of the following parameters:

- **Number of snapshots:** $q = 2, 4$
- **Penalty constant offsets:** $j = -4, -3, -2, -1, 0$

Repeat for 10 trials.

D.3 Two-dimensional signal

This section describes the experiment with two-dimensional signals in Section 7.4.3.

D.3.1 Procedure

Repeat the following for desired parameter choices:

1. Generate a parameter vector $\boldsymbol{x} \in \mathbb{R}^m$ with independent standard normal entries.
2. Generate a random rank-r matrix $\boldsymbol{Y} \in \mathbb{M}^{64\times 64}$.
3. Construct the matrix $\boldsymbol{S} \in \mathbb{M}^{2048\times m}$ from the first m columns of the normalized DCT matrix, and generate a matrix $\boldsymbol{T} \in \mathbb{M}^{2048\times 4096}$ with standard normal entries. (Performed internally by the `SelfCalibMeasurement` object.)
4. Generate a noise vector $\boldsymbol{z} \in \mathbb{R}^{2048}$ with standard normal entries.
5. Compute σ for a target SNR using the procedure from the denoising experiments (Section B.1.3).
6. Generate measurements $\boldsymbol{b} \in \mathbb{R}^{2048}$ by computing $\boldsymbol{b} = \operatorname{diag}(\boldsymbol{S}\boldsymbol{x})\boldsymbol{T}\operatorname{vec}(\boldsymbol{Y}) + \sigma\boldsymbol{z}$.
7. Compute the penalty constant λ_0 using the procedure in Section B.2 and apply offset j to obtain $\lambda = \lambda_0 \cdot 2^j$.
8. Generate a `Problem` object and call the solver with desired options.
9. Project the output $\widehat{\mathcal{A}}$ to a rank-1 operator, and compute relative error/RSDR.

D.3.2 Parameter choices

We use the alternating minimization solver with the $\ell_2 \otimes S_1$ nuclear norm, 1 dyad, convergence tolerance $\epsilon = 10^{-3}$, and a maximum of 10 outer iterations. Fix the SNR at 15dB. We vary m and r independently through 1, 2, ..., 5 and test each penalty constant offset $j = -2, -1, 0, 1$. Repeat for 10 trials.