Department of Electrical and Computer Engineering

Subsampled Blind Deconvolution via Nuclear Norm Minimization

Alexander Thieken

supervised by

Dr. Kiryung Lee

April 24, 2020

Contents

1 Introduction and Background

Many phenomena can be modeled as systems that preform convolution, including negative effects on data like translation/motion blurs. Blind Deconvolution (BD) is a process used to reverse the negative effects of a system by effectively undoing the convolution. Not only can the signal be recovered, but the impulse response can as well. "Blind" signifies that there is incomplete knowledge of the impulse responses of an LTI system. Solutions exist for preforming BD but they assume data is fully sampled. In this project we start from an existing method [1] for BD then extend to the subsampled case. We show that this new formulation works under similar assumptions. Current results are empirical, but current and future work focuses providing theoretical guarantees for this algorithm.

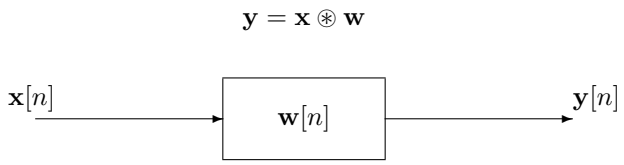


Figure 1: Basic Model LTI System

BD applies to a wide variety of settings that can be modeled with LTI systems. Since both the input signal and the impulse response can be recovered, Blind Deconvolution can be used for simultaneously recover a channels impulse response and the message sent. In addition, we will discuss this method for the 2 dimensional and higher dimensional cases. This allows us to consider several applications in image processing such as image deblurring. Another multidimensional application of interest is Magnetic Resonance Imaging (MRI). In this modality several coils of a an MRI scanner are used to collect samples. Each coil collects the Fourier transform of an image, but a "coil sensitivity" is applied to this image through convolution. Typically these coil sensitivities have to be calibrated before the image acquisition, but with BD the coil sensitivities and the original image can be recovered simultaneously. Without the need to pre-calibrate coil sensitivities and with subsampling, scan times can be reduced.

2 Problem Formulation

This section will provide the mathematical details of the BD problem. We first introduce the mathematical background needed, then jump into the problem formulation for the 1D case. Finally we show how nuclear norm minimization can be used to solve this problem and discuss how look at the problem when using 2 dimensional signals.

2.1 Mathematical Background

2.1.1 Linear Vector Spaces

A Linear Vector Space (LVS) is a generalization of the typically used vector space ($\mathbb{R}^n$ or $\mathbb{C}^n$). Let V be a set and $\mathbb{K}$ be a scalar field (e.g., $\mathbb{R}$ or $\mathbb{C}$). Let $+ : V \times V \to V$ and $\cdot : \mathbb{K} \times V \to V$ denote the elementwise addition and scalar multiplication, respectively. Then V is an LVS if it satisfies the following properties:

1. $\mathbf{a} + \mathbf{b} \in V$ for all $\mathbf{a}, \mathbf{b} \in V$.

2. $c\,\mathbf{a} \in V$ for all $c \in \mathbb{K}$ and $\mathbf{a} \in V$.

More complete lists of properties can be found in many linear algebra textbooks. As we can see, the vector spaces $\mathbb{R}^n$ or $\mathbb{C}^n$ are valid LVS, but now we can describe more general spaces such as the space of matrices

$$V = \{\mathbf{A} \in \mathbb{C}^{m \times n}\}$$

and the space of polynomials of a fixed order

$$V = \{\alpha_0 + \alpha_1 t + \alpha_2 t^t \cdots + \alpha_N t^N \ : \alpha_i \in \mathbb{C}, \forall i = 0, \ldots, N\}.$$

Note that matrices, $A \in \mathbb{C}^{m \times n}$, can be thought of as vectors in a LVS *and* as linear maps from $\mathbb{C}^n$ to $\mathbb{C}^m$. This brings up the the idea of linear maps. In the vectors spaces that many are accustom to ($\mathbb{C}^n$), matrices are the linear maps between vector spaces. But in general LVS's linear maps are called "Linear Operators" which are to LVS's what matrices are to $\mathbb{C}^n$. Similarly, the ideas of norm and inner product can be extended. In the table below, we show each notion in the general LVS V and when $V = \mathbb{C}^n$.

Name	General V	$V = \mathbb{C}^n$
"Norm"	$\|\cdot\| : V \mapsto \mathbb{R}$	$\|\cdot\|_p = \left(\sum_{i=1}^n (x_i)^p\right)^{1/p}$
"Inner Product"	$\langle x, y\rangle$	$\langle x, y\rangle = x^* y$
"Linear Operator"	$\mathcal{A} : V_1 \mapsto V_2$	$\mathcal{A} = A \in \mathbb{C}^{m\times n}$

The idea of a linear operator will be used extensively in the problem formulation and is key to the analysis in the original [1] paper. The other two concepts that are needed to properly describe the problem are that of the Singular Value Decomposition (SVD) and the nuclear norm.

2.1.2 Singular Value Decomposition

The SVD can be thought of as an extension of the eigenvalue decomposition to non-square matrices. It is defined as follows: For each matrix $\mathbf{X} \in \mathbb{C}^{m\times n}$, there exist unitary matrices $\mathbf{U} \in \mathbb{C}^{m\times m}, \mathbf{V} \in \mathbb{C}^{n\times n}$ and a nonnegative diagonal matrix $\mathbf{\Sigma} \in \mathbb{R}^{m\times n}$ such that

$$\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^*.$$

Here $*$ denotes the conjugate transpose. The diagonal entries of $\mathbf{\Sigma}$, denoted by σ_i for $i = 1, \ldots, \min\{m, n\}$, are known as the "singular values" of the matrix $\mathbf{X}$, while $\mathbf{U}$ and $\mathbf{V}$ represent two rotations. Together they describe how $\mathbf{X}$ acts on some input vector by rotating, scaling, then rotating. This is similar how a square matrix's eigen decomposition describes how it rotates, scales, then rotates back. Note that the singular values are ordered as $\sigma_1 \geq \cdots \geq \sigma_d \geq 0$ by convention. We define $d = \min\{m, n\}$ here. The singular values are also connected to the rank of the matrix $\mathbf{X}$. For example, if $\mathbf{X} \in \mathbb{C}^{m\times n},\ m > n$ has rank $r < m$ then only the first r singular values will be non-zero.

2.1.3 Nuclear Norm

Now we define the nuclear norm on the LVS of $m \times n$ matrices (i.e. $V = \{\mathbf{X} \in \mathbb{C}^{m\times n}\}$) as follows:

$$\|\mathbf{X}\|_* = \sum_{i=1}^{d} |\sigma_i|,$$

where σ_i denotes the ith singular value of $\mathbf{X}$.

Minimizing this norm is thought of as the convex relaxation of minimizing the rank of a matrix $\mathbf{X}$. This will be key to describing the problem formulation.

2.1.4 Sampling Operator

The sampling operator, $\mathcal{S}_\Omega$ is important for mathematically describing the BD problem with subsampling. $\mathcal{S}_\Omega(\mathbf{x})$ takes random samples of the input $\mathbf{x}$ with the indices of the samples described by the set Ω. It is important for later analysis that the set of indices, Ω, be uniformly random over all of the indices of the input signal. Mathematically, we treat it as a linear operator, with it's adjoint being zero-padding, i.e. placing all of the samples back in their proper indices and leaving zeros everywhere else. It is very simple to implement in a language like MATLAB because Ω is a set of indices of an array and we simply access those values in the array, and store them in a new array.

2.2 1-D Case

Our goal is to separate two signals $\mathbf{w}, \mathbf{x} \in \mathbb{R}^L$ from their convolution $\mathbf{y} = \mathbf{x} \circledast \mathbf{w}$, $\mathbf{y} \in \mathbb{R}^L$ with only M measurements $\mathbf{z} \in \mathbb{R}^M$. To be exact, we assume the convolution is circular, but it can be extended to linear convolution. First, we assume that we are fully sampling the convolution in the domain of its FT as in [1]. For the unique identification of $\mathbf{x}$ and $\mathbf{w}$ up to the scaling ambiguity, it is necessary to introduce further structural constraints. We assume that $\mathbf{x}$ and $\mathbf{w}$ belong to their corresponding low-dimensional subspaces. This is equivalent to having some prior information about $\mathbf{x}$ and $\mathbf{w}$ before blind decovolution. For example, we may assume that $\mathbf{w}$ corresponds to an FIR filter of a fixed order and $\mathbf{x}$ belongs to a data-adaptive subspace obtained by principal component analysis of known class of signals.

First, we formally describe the subspace models for $\mathbf{x}$ and $\mathbf{w}$. Assume that $\mathbf{x}$ belongs to the columnspace of $\mathbf{C} \in \mathbb{R}^{L\times N}$. Likewise, assume $\mathbf{w}$ is in the columnspaces of $\mathbf{B} \in \mathbb{R}^{L\times K}$. Then there exist $\mathbf{m} \in \mathbb{R}^N$ and $\mathbf{h} \in \mathbb{R}^K$ such that

$$\mathbf{x} = \mathbf{C}\mathbf{m} \quad \text{and} \quad \mathbf{w} = \mathbf{B}\mathbf{h}. \tag{1}$$

So, if we find $\mathbf{m}$ and $\mathbf{h}$, then we also find $\mathbf{x}$ and $\mathbf{w}$ by (1).

Let $\mathcal{F} : \mathbb{C}^L \to \mathbb{C}^L$ denote the Fourier Transform operator. For the brevity, we let $\hat{\mathbf{y}}$ denote $\mathcal{F}(\mathbf{y})$. Likewise, $\hat{\mathbf{B}}$ will denote the matrix obtained by concatenating the FT of columns of $\mathbf{B}$. Throughout this thesis, we will use this shorthand notation for the Fourier transform for vectors and matrices. Then by the convolution theorem for the Fourier Transform (FT), which changes the convolution in the time domain to

point-wise multiplication in the frequency domain, we obtain

$$\hat{\mathbf{y}} = \mathcal{F}(\mathbf{x} \circledast \mathbf{w}) = \mathcal{F}\left(\mathbf{B}\mathbf{h} \circledast \mathbf{C}\mathbf{m}\right) = \sqrt{L}\left(\hat{\mathbf{B}}\mathbf{h} \odot \hat{\mathbf{C}}\mathbf{m}\right), \tag{2}$$

where $\odot$ denotes the point-wise multiplication operator.

Let $\hat{\mathbf{b}}_l$ and $\hat{\mathbf{c}}_l$ denote the l^{th} column of $\hat{\mathbf{B}}^*$ and the l^{th} column of $\sqrt{L}\hat{\mathbf{C}}^{\mathsf{T}}$ respectively. Then the l^{th} entry of $\hat{\mathbf{y}}$ denoted by $\hat{\mathbf{y}}[l]$, is written as

$$\hat{\mathbf{y}}[l] = \mathbf{e}_l^{\mathsf{T}}\hat{\mathbf{B}}\mathbf{h}\,\mathbf{e}_l^{\mathsf{T}}\hat{\mathbf{C}}\mathbf{m} = \hat{\mathbf{b}}_l^*\mathbf{h}\,\hat{\mathbf{c}}_l^{\mathsf{T}}\mathbf{m} = \hat{\mathbf{b}}_l^*\mathbf{h}\,\mathbf{m}^{\mathsf{T}}\hat{\mathbf{c}}_l$$

for $l = 1, \ldots, L$, where $\mathbf{e}_l$ denotes the l^{th} column of the identity matrix of size L. Then $\hat{\mathbf{y}}$ is compactly rewritten as

$$\hat{\mathbf{y}} = \sqrt{L}\ \text{diag}\left(\hat{\mathbf{B}}\mathbf{h}\mathbf{m}^{\mathsf{T}}\hat{\mathbf{C}}^{\mathsf{T}}\right). \tag{3}$$

Here $\text{diag} : \mathbb{C}^{L\times L} \to \mathbb{C}^L$ constructs a column vector from the diagonal entries of its matrix-valued argument.

Since each entry of $\hat{\mathbf{y}}$ is a linear function of $\mathbf{h}\mathbf{m}^{\mathsf{T}}$, we can construct a linear operator $\mathcal{A} : \mathbb{C}^{K\times N} \to \mathbb{C}^L$ that satisfies $\hat{\mathbf{y}} = \mathcal{A}(\mathbf{h}\mathbf{m}^{\mathsf{T}})$ as follows: Let

$$\mathbf{\Phi}_l = \hat{\mathbf{b}}_l\hat{\mathbf{c}}_l^*, \quad \forall\ l = 1, \ldots, L.$$

Then $\mathcal{A}$ is defined by

$$\mathcal{A}(\mathbf{X}) = \begin{bmatrix} \langle \mathbf{\Phi}_1, \mathbf{X}\rangle \\ \vdots \\ \langle \mathbf{\Phi}_L, \mathbf{X}\rangle \end{bmatrix} = \begin{bmatrix} \text{trace}\left(\mathbf{\Phi}_1^*\mathbf{X}\right) \\ \vdots \\ \text{trace}\left(\mathbf{\Phi}_L^*\mathbf{X}\right) \end{bmatrix}, \quad \forall\ \mathbf{X} \in \mathbb{C}^{K\times N}. \tag{4}$$

So far, this is mostly unchanged from [1]. Our modification assumes we take samples in the domain of the convolution. The change of domain of the convolution is in the interest of MRI, where samples are taken in the same domain as the convolution. This defines a new linear operator $\mathcal{Q} : \mathbb{C}^L \to \mathbb{C}^M$ as

$$\mathcal{Q} := \sqrt{\frac{L}{M}}\mathcal{S}_\Omega\mathcal{F}^*.$$

Note that the observations are normalized by the number of samples M, and $M < L$ for subsampling. If we let $\mathbf{X} = \mathbf{h}\mathbf{m}^{\mathsf{T}}$, our new vector of observations $\mathbf{z} \in \mathbb{R}^M$ becomes:

$$z = \mathcal{Q}(\mathcal{A}(\mathbf{X}))$$

These two operators can be grouped into a single linear operator, but it does not change the overall action on the input. Now we have a rigorous way to describe the how to go from the sub-sampled observations to $\mathbf{x}$ and $\mathbf{w}$. Finding $\mathbf{X}$ is analogous to finding $\mathbf{x}$ and $\mathbf{w}$ up to a scaling factor.

2.2.1 Solution

If we can recover $\mathbf{X}$ then we can separate it into $\mathbf{h}$ and $\mathbf{m}$ using the SVD and use the known subspaces $\mathbf{C}$ and $\mathbf{B}$ to get $\mathbf{x}$ and $\mathbf{w}$. Since the matrix X is rank-1, only one singular value will be nonzero, thus there will only be one left singular vector, which is $\mathbf{h}$ and one right singular vector, which is $\mathbf{m}$. Since we cannot easily minimize rank, we can solve the following minimization problem.

$$\begin{aligned} \underset{X}{\text{minimize}} \quad & \|\mathbf{X}\|_* \\ \text{subject to} \quad & \mathbf{z} = \mathcal{Q}(\mathcal{A}(\mathbf{X})) \end{aligned} \tag{5}$$

We know from its definition that $\mathbf{X}$ is rank-1 so we can use the a minimization of the nuclear norm to solve for $\mathbf{X}$. Minimizing $\|\mathbf{X}\|_* = \sum_{i=1}^{r} |\sigma_r|$ will give us a solution that have few non-zero singular values and therefore low rank. This is similar to how minimizing $\|x\|_1 = \sum_{i=1}^{n} |x_i|$ encourages sparse solutions for a vector. The other advantage of the nuclear norm is that it is convex, so a local minimizer is a global minimizer. We expect that $\mathbf{X} = \mathbf{h}\mathbf{m}^T$ is the minimizer to this optimization problem because we can only move on the set where the constraint is true.

2.3 2-D Case

For the 2D case we think of the subspaces, $\mathbf{B}$ and $\mathbf{C}$ as general linear operators instead of just matrices. As before, the Fourier transform of their components, $\hat{\mathbf{b}}_l$ and $\hat{\mathbf{c}}_l$, are what act on $\mathbf{h}\mathbf{m}^T$. We can use many of the same simplifications as the 1-D case if we just think of the vectorization of each element. The subspace

model becomes:

$$\mathbf{x} = \mathcal{C}(\mathbf{m}) \quad \text{and} \quad \mathbf{w} = \mathcal{B}(\mathbf{h}).$$

Here, $\mathbf{y}, \mathbf{x}$ and $\mathbf{w}$ are all $L_1 \times L_2$ rather than just $L \times 1$. Next we explicitly define the linear operators, $\mathcal{C} : \mathbb{R}^N \mapsto \mathbb{C}^{L_1 \times L_2}$ and $\mathcal{B} : \mathbb{R}^K \mapsto \mathbb{C}^{L_1 \times L_2}$. These linear operators tell us how to go from each subspace to the signals $\mathbf{x}$ and $\mathbf{w}$. Before, the subspaces were spanned by the columns of a matrix, but now the subspaces are spanned by matrices, which are "generalized" vectors in a LVS. The basis vectors for $\mathbf{x}$'s subspaces are, $\Psi_i \in \mathbb{C}^{L_1 \times L_2}$ with $i = 1, ..., N$. Likewise, for $\mathbf{w}$'s subspace, the basis vectors are $\Phi_i \in \mathbb{C}^{L_1 \times L_2}$ with $i = 1, ..., K$. So we can see that with the vectors $\mathbf{h}$ and $\mathbf{m}$ we can construct

$$\mathbf{x} = \mathcal{C}(\mathbf{m}) = \sum_{i=1}^{N} \Psi_i \mathbf{m}(i) \quad \text{and} \quad \mathbf{w} = \mathcal{B}(\mathbf{h}) = \sum_{i=1}^{K} \Phi_i \mathbf{h}(i).$$

We can write the same equations as the 1D case with linear operators instead of matrices. After we vectorize we use the same exact equations as the 1D case, with different definitions of $\hat{\mathbf{B}}$ and $\hat{\mathbf{C}}$. The vectorization of $\hat{\mathbf{y}}$, written as $\text{vec}(\hat{\mathbf{y}})$, simply stacks all of its columns into one long $L_1 \cdot L_2 \times 1$ vector. Additionally, the Fourier Transform is now the (normalized) 2D Fourier Transform, $\mathcal{F} : \mathbb{C}^{L_1 \times L_2} \to \mathbb{C}^{L_1 \times L_2}$.

First we use convolution theorem of the FT as in (2),

$$\hat{\mathbf{y}} = \mathcal{F}(\mathcal{B}(\mathbf{h}) \circledast \mathcal{C}(\mathbf{m})) = \sqrt{L_1 L_2} \; (\; \mathcal{F}(\mathcal{B}(\mathbf{h})) \odot \mathcal{F}(\mathcal{C}(\mathbf{m})) \;).$$

Next, we use the vectorization of $\hat{\mathbf{y}}$ then write the linear operators $\mathcal{F}(\mathcal{B}(\cdot))$ as $\hat{\mathbf{B}}$ and $\mathcal{F}(\mathcal{C}(\cdot))$ as $\hat{\mathbf{C}}$. This puts the output in terms of matrix-vector multiplications:

$$\text{vec}(\hat{\mathbf{y}}) = \sqrt{L_1 L_2} \; (\hat{\mathbf{B}}\mathbf{h} \odot \hat{\mathbf{C}}\mathbf{m})$$

Using the same argument as (3) and (4), we see that we take the diagonal entries of the matrix product $\hat{\mathbf{B}}\mathbf{h}\mathbf{m}^\mathsf{T}\hat{\mathbf{C}}^\mathsf{T}$ and that it is a linear operator:

$$\text{vec}(\hat{\mathbf{y}}) = \sqrt{L_1 L_2} \; \text{diag}\left(\hat{\mathbf{B}}\mathbf{h}\mathbf{m}^\mathsf{T}\hat{\mathbf{C}}^\mathsf{T}\right) = \mathcal{A}\left(\mathbf{h}\mathbf{m}^\mathsf{T}\right)$$

$\hat{\mathbf{B}}$ and $\hat{\mathbf{C}}$ are now defined using the vectorization of their components' Fourier Transforms:

$$\hat{\mathbf{C}} = \left[\text{vec}(\hat{\Psi}_1) \; \ldots \; \text{vec}(\hat{\Psi}_K)\right] \in \mathbf{C}^{L_1 L_2 \times K}$$

$$\hat{\mathbf{B}} = \left[\text{vec}(\hat{\Phi}_1) \; \ldots \; \text{vec}(\hat{\Phi}_N)\right] \in \mathbf{C}^{L_1 L_2 \times N}$$

Now everything can be simplified to the 1D case with the same linear operators and subsampling. Here sample in the domain of the convolutions as in MRI.

$$\mathcal{Q} = \sqrt{\frac{L}{M}}\mathcal{S}_\Omega \mathcal{F}^*$$

Again, we define $\mathbf{X} = \mathbf{h}\mathbf{m}^\mathsf{T}$ and the equation for our samples $\mathbf{z}$:

$$\mathbf{z} = \mathcal{Q}(\mathcal{A}(\mathbf{X}))$$

So we see the same nuclear norm minimization will work for the 2D case. If we recover $\mathbf{X}$ then we can extract $\mathbf{h}$ and $\mathbf{m}$, then return to $\mathbf{x}$ and $\mathbf{w}$.

$$\begin{aligned} \underset{X}{\text{minimize}} \quad & \|\mathbf{X}\|_* \\ \text{subject to} \quad & \mathbf{z} = \mathcal{Q}(\mathcal{A}(\mathbf{X})) \end{aligned}$$

Everything relies on the vectorization operation. While it may be surprising that everything returns to 2D images after removing the vectorization, it is just a matter of using the same convention when transforming to a vector and back. So in the 2D case, one would stack each column of a $L_1 \times L_2$ matrix into one long column vector and rearrange when transforming back to a matrix. This will work with higher dimensions too. However, only 2D cases are discussed in this paper.

3 Numerical Results

In this section we go over how this nuclear norm minimization was implemented in MATLAB. Then we discuss the simulations which show the conditions for which this solution works. Finally, we show an example for the 2D case where we wish to reconstruct images that have been convolved with an FIR filter.

3.1 Introduction to TFOCS in MATLAB

TFOCS is library for MATLAB that has solvers built for minimization problems using a standard form. It includes MATLAB functions that can minimize both smooth and non-smooth functions. Since the nuclear

norm is a non-smooth function, the TFOCS function `tfocs_SCD.m` was used. This function is based on a model called *smoothed conic dual* (SCD) and is described further in [3]. The standard form is as follows:

$$\underset{x}{\text{minimize}} \quad f(x) + \frac{1}{2}\mu\|x - x_0\| + h\left(\mathcal{A}(x) + b\right)$$

The first term, f is $\|\mathbf{X}\|_*$ in our case. The second term is used to "smooth" the whole objective so we can use gradient descent based methods. The final term, h is an indicator function, which allows us to move the constraint $\mathcal{A}(x)+b=0$ into the objective. In our case this is the constraint is $\mathcal{Q}(\mathcal{A}(\mathbf{X}))+\mathbf{z}=0$. TFOCS allows the definition of custom linear operators and the chaining of several linear operators together. Below are a few lines of code to show how we use TFOCS to solve our minimization problem, (5):

```
sampOp = linop_compose( linop_subsample({[L,1],[M,1]},Omega), ifftOp ) ;
 z = sampOp(y_hat,1) ;
lin_op = linop_compose(sampOp, @(x,mode)A_op(B,C,L,K,N,x,mode));
X1 = tfocs_SCD(prox_nuclear,{lin_op, -z},prox_l2(1e-3),0.01);
```

Note that we define "`A_op.m`" ourselves using the equations from the problem formulation and $\mathcal{Q}$ is what we call "`sampOp`". The detailed code is attached in Appendix A.

3.2 MATLAB Algorithm Design

The algorithm was written in MATLAB using the TFOCS optimization package. First after the initial equations from the problem formulation were derived, the 1D case was implemented. Each linear operator was written as a function and tested individually with known inputs to ensure that they worked properly. Then, all of the linear operators were chained together using the `linop_compose()` function, which allowed the use of `tfocs_SCD.m` to solve the minimization problem.

Once the 1D case was working, the equations used were further refined to increase the speed of the MATLAB script. This focused on removing loops from the script since MATLAB is line-interpreted. Many

of the equations were rewritten to remove the `diag()` operation because if that operation was used then all of the calculations for the off-diagonal entries were not utilized. Phase maps were generated to analyze performance with different parameters. Finally, the linear operators were rewritten to extend to the 2D case.

3.3 Simulation Results

The results are best summarized by a phase map. These vary over several different values for L,N,K,M to show the probability of success for the nuclear norm minimization in each scenario. In [1] they show that their phase maps are consistent with theory that they derive. Since we have not yet developed the same theory, we show that when using fully sampled data, we reproduce the results from [1] and as we decrease the number of samples, set of parameters over which we are successful decreases by a log factor.

For each simulation we make the same assumptions for the two signals being convolved. We assume that $\mathbf{x}$ is a random vector from subspace C that is a Gaussian matrix. That is to say, C has iid Gaussian entries. $\mathbf{w}$ is also randomly generated, but is from subspace B that contains FIR filters of fixed length K. Each trial was stopped when the step size reached 1×10^{-7}, since the program is convex, this will be close to the absolute minimum. A trial was counted as a success when $\|\mathbf{X}_{\text{true}} - \mathbf{X}_{\text{new}}\| \leq 1 \times 10^{-3}$.

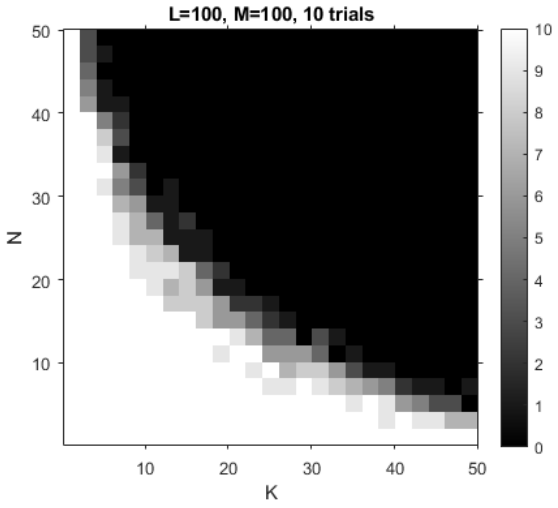
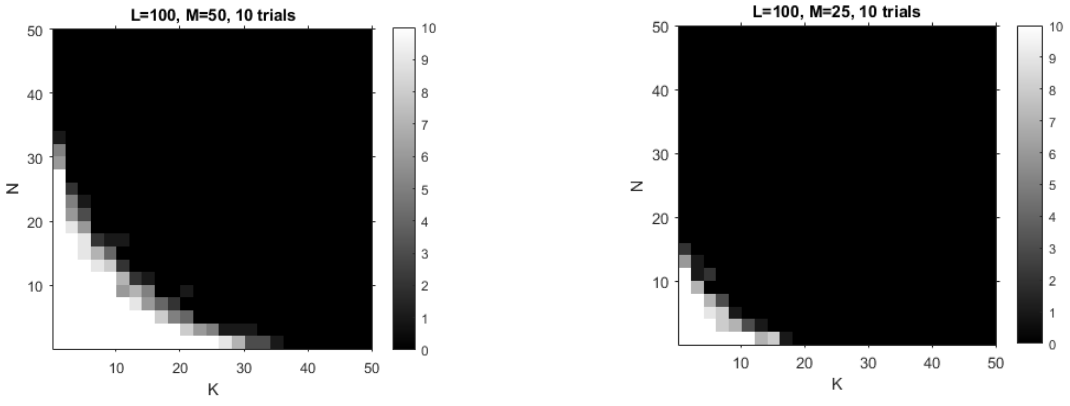


Figure 2: Phase map with $L = 100$, $M = 100$. Varying over dimensions N and K



(a)

(b)

Figure 3: Phase transition maps with M=50,25. Varying over dimensions N and K

From fig 2 it can be seen that we produce results similar to [1] with fully sampled data (M=100). We see the algorithm works with high probability for $N + K \lesssim L$. If one were to fix the size of the known subspaces, $\mathbf{C}$ and $\mathbf{B}$, then it would be advantageous to increase the total length of the signals $\mathbf{x}$ and $\mathbf{w}$. With our addition of subsampling we observe that the area for which the algorithm works decreases as we take fewer samples. Again, the algorithm works with high probability for $N + K \lesssim M \leq L$. This is expected because we are collecting less information by subsampling, but is important for applications that make use of subsampling. If the dimensions of the known subspace are small enough, we can significantly undersample and still recover $\mathbf{x}$ and $\mathbf{w}$. It is also important to note that the subsampling is random. If one were to use a fixed sampling pattern, the same performance would not be expected.

The distinction between the two subspaces $\mathbf{B}$ and $\mathbf{C}$ is not very important because we can exchange which of the signals is the impulse response and which is the input. This is because convolution is commutative. Showing that this method works with high probability for Gaussian $\mathbf{C}$, implies that it will work for many different choices of $\mathbf{C}$. In a specific application, it is assumed that $\mathbf{C}$ is known, not random. Using $\mathbf{B}$ as an FIR subspace, covers many common cases such channel estimation and multi-coil MRI where the channel and coil-sensitivities are often modeled as FIR.

4 2D Demonstration

A demonstration was created to show a possible application of this algorithm. In this case a set of 1000 Shepp-Logan phantoms were generated with slightly varied parameters. The image sizes were kept to 50 $\times$ 50 pixels in order to keep the processing times low. Principal Component Analysis (PCA) was performed on this set of images and the first 10 Principal Components served as the $\mathbf{\Psi}_i$'s of the "known subspace" $\mathcal{C}$. The other subspace, $\mathcal{B}$ was chosen to be that of 3 $\times$ 3 FIR filters. The weights for one of the images in the set were used along with random FIR filter coefficients as the ground truth $\mathbf{m}$ and $\mathbf{h}$ respectively. In this scenario, we collected subsampled data ($M = \frac{3}{4}L$) from the convolution of an unknown image and an unknown filter. Using nuclear norm minimization, the following results were achieved.

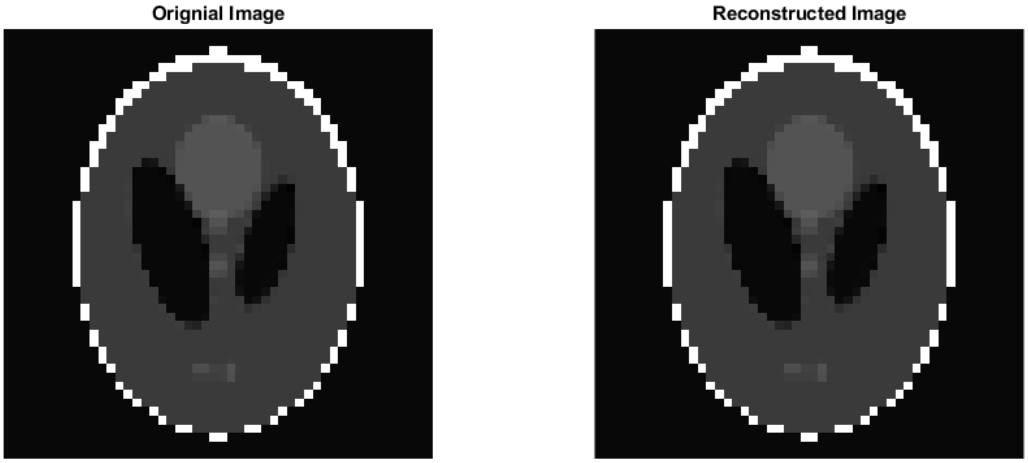


Figure 4: Comparing ground truth and reconstruction of Shepp-Logan phantom

As can be seen the image is reconstructed accurately. The weights, $\mathbf{m}$, where used with PCA components to create this reconstructed image. The filter coefficients, $\mathbf{h}$, were also accurately recovered. Since the solution from the minimization, $\mathbf{X}$, is a low rank matrix, SVD was used to separate the component vectors. The left and right singular vectors were $\mathbf{h}$ and $\mathbf{m}$. They are off by a scaling factor, so the following figures 5 and 6 show the normalized versions of these vectors.

For both $\mathbf{h}$ and $\mathbf{m}$, the scaling factor was negative, so the correlation coefficients between the recovered and ground truth vectors were ≈ -1. This means the vector were along the same direction.

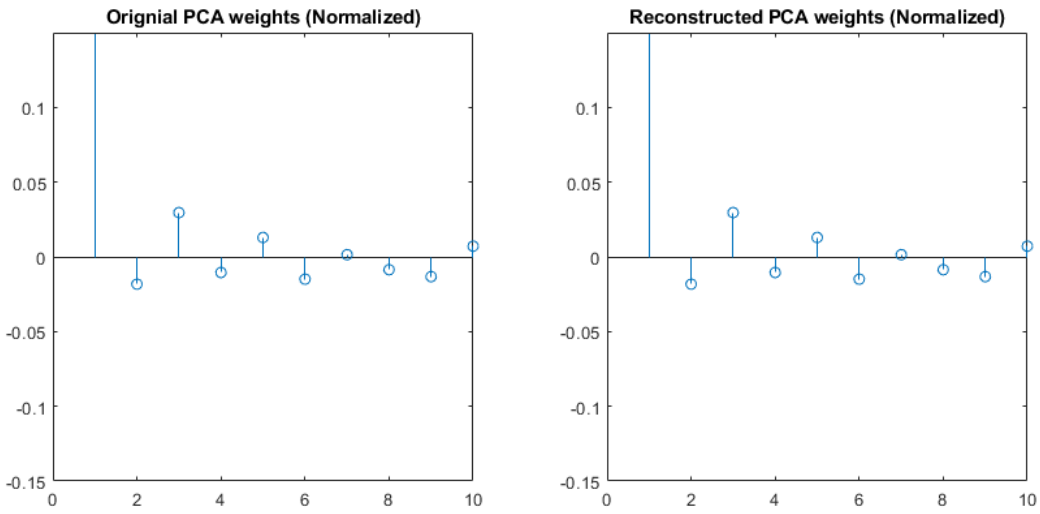


Figure 5: Weights for principal components, $\mathbf{m}$

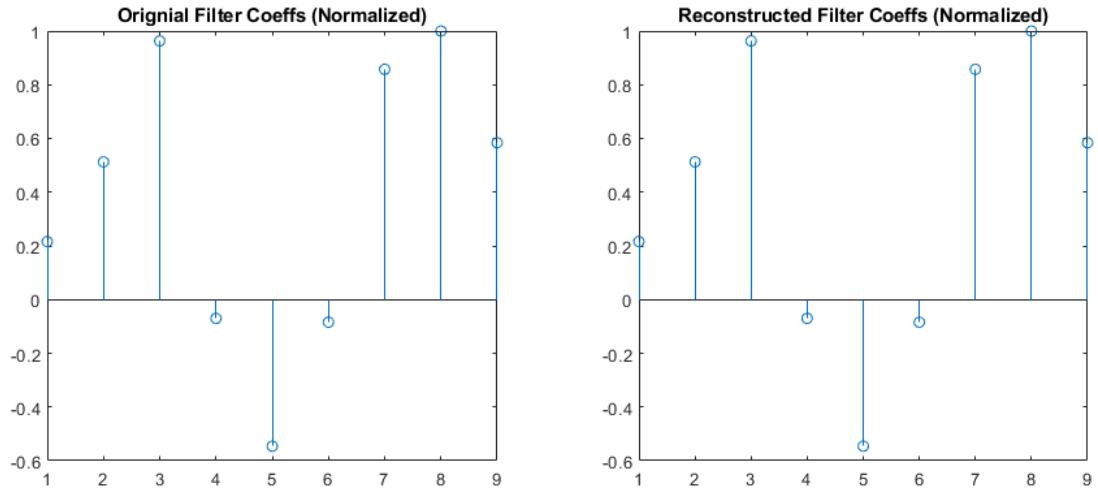


Figure 6: Coefficients for random filter, $\mathbf{h}$

5 Future Work

There are many directions for this project to move forward. The first is to extend the existing theory from [1] to provide similar theoretical guarantees for the subsampled case. It is expected that a similar proof technique can be used. This proof technique has two components, finding the conditions for the "inexact dual certificate", and showing it exists via "golfing scheme". The inexact dual certificate is a sufficient condition that shows that $\mathbf{X}_0 = \mathbf{h}\mathbf{m}^T$ is the unique minimizer of (5). Golfing scheme is an iterative method that proves that an inexact dual certificate exists. Assumptions that will be key are randomized subsampling and the subspace $\mathbf{C}$ being Gaussian. This theory work will fit in with the work by Prof Kiryung Lee's other student, Farhad Mirkazemi, who is working on a review of theory for several Blind Deconvolution methods. Weekly meeting have been held where Prof. Lee's group discuss theoretical background of BD and related topics, these will be continued with this work as part of the discussion.

The second direction for this work will be applying it in an multi-coil MRI setting. The algorithm must work for 2D case and for the multi-coil case. The extension to the multi-coil case has already been completed and implemented in the 1D case. As of the date of this thesis, the multi-coil 2D case is still in the initial stages of development. This will fit in with a model proposed by Dr. Rizwan Ahmad for multi-coil MRI, known as Parallel MRI (pMRI). This model takes advantage of multiple coils collecting data simultaneously as well as expected temporal sparsity from rapidly imaging the same cross section several times. Regular meetings will continue to be held where Prof. Ahmad and Prof. Lee's groups discuss several methods (including this one) for the aforementioned model. Additionally, this will require further coordination with Dr. Ahmad's student, Aaron Pruitt, who manages the computing resources. Many of the simulation results above were done on these computing resources.

Furthermore, if this algorithm were to be tested using MRI data from human subjects, IRB Human Subject Protocol must be followed to collect that data. Additionally, image data generated would have to follow standard formats for medical images such as those outlined by the Digital Imaging and Communications in Medicine (DICOM) standard. For more information on these formats, see [2].

6 Conclusion

In this project, we have shown that we can successfully recover two signals from their convolution with certain assumptions. Previous work focuses mostly on the fully sampled case. Motivated by applications such as MRI where subsampling is important, we extend a known solution for BD to the subsampled case. If we have two known subspaces, rewrite our problem so that we are taking linear measurements of a rank-1 matrix, $\mathbf{X} = \mathbf{h}\mathbf{m}^T$. Next, we can use nuclear norm minimization with our observations as a linear constraint to find that matrix X (as in (5)). For this solution, we have shown the sizes of the two subspaces for which this works through phase maps. In addtion, a example application for a 2D signal was demonstrated. Our main contribution is showing the the extension of BD to the subsampled case is successful numerically and future work will focus on applying this to pMRI and providing theoretical guarantees similar to [1].

7 Acknowledgements

Firstly, I would like to thank my advisor, Dr. Kiryung Lee, who made himself available every time that I needed help on our project, in classes, and dealing with the stress of school. He consistently guided me in the right direction while allowing me to understand and complete this work. Additionally, I would like to thank Dr. Rizwan Ahmad and the other students in Prof. Lee's group, Seonho Kim and Seyedfarhad Mirkazemi. Much of my understanding of this material was the result of group meetings and discussions with them.

References

[1] Ali Ahmed, Benjamin Recht, and Justin Romberg. Blind deconvolution using convex programming. *IEEE Transactions on Information Theory*, 60(3), 2014.

[2] M. Larobina and L. Murino. Medical image file formats. *Springer US*, 27:200–6.

[3] E. J. Candés S. Becker and M. Grant. Templates for convex cone problems with applications to sparse signal recovery. *Math. Prog. Comp.*, (3):165–218, 2011.

Appendices

A MATLAB Code

Function to implement $\mathcal{A}$(in) and it's adjoint $\mathcal{A}^*$(in) for 1D case

```
function [out] = A_op(B,C,L,K,N,in,mode)


    switch mode

        case 0

            %return the input size and output size for forward operation

            out = {[K,N], [L,1]};

        case 1

            %forward operation - need B and C which are the subspaces that

            %the convolved vectors come from.

            %This generates the observations from the outer product that we

            %are trying to find

            out = zeros(L,1);

            B_hat = (1/sqrt(L))*fft(B);

            C_hat = (1/sqrt(L))*fft(C);


            out = sum((sqrt(L)*B_hat*in).*C_hat,2);


        case 2

            % adjoint operator

            out = zeros(K,N);

            B_hat = (1/sqrt(L))*fft(B);

            C_hat = (1/sqrt(L))*fft(C);
```

```
            out = B_hat'*diag(in)*conj(C_hat)*sqrt(L);

    end

end
```

Function to test implimentation for given set of parameters:

```
function [success] = BlindDeconv(L,K,N,M)
%   L is length of of the signals w and x
%   K is the lenght of h
%   N is the length of m
%   M is the number of samples to take in the time domain



    %generate a the signals w=Bh and x=Cm


    B = eye(L,K);
    C = randn(L,N);


    m = randn(N,1);
    h = randn(K,1);


    w = B*h;
    x = C*m;


    X0 = h*m';


    %fully sampled in fourier domain
    y_hat = (1/sqrt(L))*(fft(w).*fft(x));
    y = sqrt(L)*ifft(y_hat);



    %parameters for subsampling
```

```
    Omega = randperm(L,M)';

    %operators for subsampling
    ifftOp = linop_handles([L,L], @(x)ifft(x)*sqrt(L), @(x)fft(x)/sqrt(L) ,'C2C');
    sampOp   = linop_compose( linop_subsample({[L,1],[M,1]},Omega), ifftOp );

    %subsampled data
    z = sampOp(y_hat,1);

    %use nuclear norm minimization to find an approximation of X0
    %use TFOCS for this

    lin_op = linop_compose(sampOp, @(x,mode)A_op(B,C,L,K,N,x,mode));

    opts = tfocs_SCD; opts.printEvery = 0; %suppress the TFOCS outputs for the MonteCarlo
    ↪  Simulation
    opts.tol = 1e-8; %lower step-size tolerance to speed up (default is 1e-8)
    X1 = tfocs_SCD(prox_nuclear,{lin_op, -z},prox_l2(1e-3),0.01,[],[],opts);

    success = norm(X1-X0) < 1e-3;

end
```

Code to run Monte-Carlo Simulation to generate phase maps:

```
clc;
L = 100;
% K = 10;
% N = 10;

N_vals = 2:2:50;
K_vals = 2:2:50;
num_reps = 10;

success = zeros(length(N_vals),length(K_vals));

M = 25; %fix the M value to get a cross-section

for N = N_vals
    for K = K_vals

        tic;
        for i = 1:num_reps
            success(N/2,K/2) = success(N/2,K/2) + BlindDeconv(L,K,N,M);
        end
        time = toc;
        disp(['N = ' num2str(N) ' K = ' num2str(K) ' Time for ' num2str(num_reps) ' reps
        ↪  = ' num2str(time/60) 'min']); %display iteration info
    end
end
```

```
imshow(kron(flip(success,1),ones(10,10)),[])

set(gca, 'visible','on')

yticks(1:50:501)

yticklabels(50:-10:10)

xticks(100:100:600)

xticklabels(10:10:50)

xlabel('K=N')

ylabel('M')

title('Phase map, L=100, 10 trials')
```

Code to generate subspace based on shepp-logan phantoms

```
rng(12345)

clear,clc;

% goal is to generate several different roations, translations, etc of a

% phantom image, then use PCA to reduce to a known subspace that we call C


%size parameters

L1 = 50;

L2 = 50;

n_samp = 1000;


%get parameters for generic shepp-logan then we can alter them slightly

[a_,E_shepp] = phantom('Modified Shepp-Logan',L1);


%generate set of phantoms

data = zeros(L1,L2,n_samp);

for i = 1:n_samp

    %parameters for phantom

    E = E_shepp;

    E(3:end,4:5) = E(3:end,4:5)+(0.01)*randn(8,2); %shift some of the elipses around
    ↪  randomly

    data(:,:,i) = phantom(E,L1); %+(0.01)*randn(L1,L1); %noise is 1% relative to total
    ↪  intensity


end


figure(1)
```

```
for i = 1:10
    imshow(kron(data(:,:,i),ones(10,10)),[])
    title(num2str(i))
    pause(0.5)
end

%% PCA - principal component analysis
data_vec = reshape(data,L1*L2,n_samp);
data_vec = data_vec.';
data_vec_mean = mean(data_vec);
data_vec_centered = data_vec - 0*data_vec_mean;

data_centered = reshape(data_vec_centered.',L1,L2,n_samp);

[coeffs,scores,latent] = pca(data_vec_centered,'Centered',false);

%data_vec_recon = scores*coeffs';
n_comps = 10;
data_vec_recon = scores(:,1:n_comps+1)*coeffs(:,1:n_comps+1).'; %only use n_comps # of
↪  components to reconstruct
data_recon = reshape((data_vec_recon + 0*data_vec_mean).',L1,L2,n_samp);

figure(2)
for i = 1:10%num_samps-1
    imshow(kron(data_recon(:,:,i),ones(10)),[]);
    title(num2str(i))
    pause(0.5)
```

```
end


%% test reconstruction

figure(3)

first_pic_recon = reshape(coeffs(:,1:n_comps)*scores(1,1:n_comps).',L1,L2);

imshow(kron(first_pic_recon,ones(10)),[])




%% save

%save only one known sample

save('subspace2_4_16_20.mat', 'L1','L2','n_comps','scores','coeffs','n_samp')
```

2D operator $mathcal A$

```
function [out] = A_op_2D(L1,L2,K,N,B_hat,C_hat,in,mode)
%A_OP Summary of this function goes here
%   Detailed explanation goes here
    switch mode
        case 0
            %return the input size and output size for forward operation
            out = {[K,N], [L1,L2]};
        case 1
            %forward operation - need B and C which are the subspaces that
            %the convolved vectors come from.
            %This generates the observations from the outer product that we
            %are trying to find
            out = zeros(L1*L2,1);

            out = sum((sqrt(L1*L2)*B_hat*in).*C_hat,2);

            out = reshape(out,L1,L2);

        case 2
            % adjoint operator
            out = zeros(K,N);



            out = B_hat'*diag(reshape(in,L1*L2,1))*conj(C_hat)*sqrt(L1*L2);
    end
end
```

Subspaces **B** and **C** as linear Operators

```
function out = C_op(L1,L2,N,Phis,in,mode)
%C_OP Summary of this function goes here
%   Detailed explanation goes here

    %subspace has guassian matricies as its basis vectors

%     C_is = randn(L1,L2,N);
%     Phis = Phis;

    %will be used to multiply along third direction of basis vectors
    mtx_vec_multplr = dsp.ArrayVectorMultiplier('Dimension',3);

    switch mode
        case 0
            %input and output size for forward operation
            out = {[N,1],[L1,L2]};
        case 1
            %case for forward operation
            %scale each basis vector by input then sum
            out = sum(mtx_vec_multplr(Phis,in),3);

%             out = zeros(L1,L2);
%             for i = 1:N
%                 out = out + C_is(:,:,i).*in(i);
%             end
```

```
        case 2
            %adjoint operation

            out = zeros(N,1);

            for i = 1:N
                out(i) = trace(Phis(:,:,i)'*in);
            end

    end
end

function out = B_op(L1,L2,K,in,mode)
%B_OP Summary of this function goes here
%   Detailed explanation goes here

    if(floor(sqrt(K))~=sqrt(K))
        disp('K does not correspond to square filter');
        out = [];
        return
    end

    %assume the 2D filter is square and causal
    %i.e. only the top left corner is filled

    %square filter is sqrt(K) by sqrt(K)
    filter_size = sqrt(K);
```

```
    switch mode
        case 0
            %input and output size for forward operation
            out = {[K,1],[L1,L2]};
        case 1
            %case for forward operation (synthesis)
            out = zeros(L1,L2);

            out(1:filter_size,1:filter_size) = reshape(in,filter_size,filter_size);

        case 2
            %adjoint operation (analysis)

            out = zeros(K,1);

            out = reshape(in(1:filter_size,1:filter_size),K,1);
end
```

functions to calculate $\hat{\mathbf{C}}$ and $\hat{\mathbf{B}}$

```
function B_hat = compute_B_hat(L1,L2,K)
%COMPUTE_B_HAT Summary of this function goes here
%   Detailed explanation goes here

    %assume the 2D filter is square and causal
    %i.e. only the top left corner is filled

    if(floor(sqrt(K))~=sqrt(K))
        disp('K does not correspond to square filter');
        B_hat = [];
        return
    end

    %square filter is sqrt(K) by sqrt(K)
    filter_size = sqrt(K);

    B_hat = zeros(L1*L2,K);

    for i = 1:K
        %single point in FIR filter
        temp = zeros(L1,L2);
        temp2 = zeros(K,1);
        temp2(i) = 1;
        temp(1:filter_size,1:filter_size) = reshape(temp2,filter_size,filter_size);
%         if(i>filter_size)
%             temp(mod(filter_size,i),mod(i,filter_size)) = 1;
```

```
%         elseif(i<filter_size)
%             temp(1,mod(i,filter_size)) = 1;
%         elseif(mod(i,filter_size) == 0)
%             temp(i/filter_size,(i/filter_size)*3) = 1;
%         end



        B_hat(:,i) = reshape((1/sqrt(L1*L2))*fft2(temp), L1*L2, 1);
    end

end

function C_hat = compute_C_hat(L1,L2,N,Phis)
%COMPUTE_C_HAT Summary of this function goes here
%   Detailed explanation goes here

    C_hat = zeros(L1*L2,N);

    for i = 1:N
        C_hat(:,i) = reshape((1/sqrt(L1*L2))*fft2(Phis(:,:,i)),L1*L2,1);
    end

end
```

Main implementation for 2D demo

```
rng(12345)

clear,clc;


%generate a the signals w=Bh and x=Cm

L1 = 50;

L2 = 50;

K = 9;

N = 10;


%% testing different linear operators

%test B_op operator

h = 1:K;

synth_B = B_op(L1,L2,K,h,1);

analy_B = B_op(L1,L2,K,synth_B,2); %should be same as h b/c orthogonal basis




%import basis for linear operator C, this was generated in other script

load('subspace2_4_16_20.mat','scores','coeffs')

Phis = reshape(coeffs(:,1:N),L1,L2,N); %use principle components as subspace


%test C_op operator

m = 1:N;

synth_C = C_op(L1,L2,N,Phis,m,1);

analy_C = C_op(L1,L2,N,Phis,synth_C,2); %not an orthogonal basis, so not same as m

trace(synth_C'*synth_C) == m*analy_C
```

```
%precomupte the bases for subspaces B and C
B_hat = compute_B_hat(L1,L2,K);
C_hat = compute_C_hat(L1,L2,N,Phis);



%test
h = randn(K,1);
m = randn(N,1);


w = B_op(L1,L2,K,h,1);
x = C_op(L1,L2,N,Phis,m,1);


X0 = h*m';


%fully sampled in fourier domain
y_hat = (1/sqrt(L1*L2))*(fft2(w).*fft2(x));
y = sqrt(L1*L2)*ifft2(y_hat);


%test A_op
y_hat_test = A_op_2D(L1,L2,K,N,B_hat,C_hat,X0,1);


norm(y_hat-y_hat_test)
(reshape(y_hat,L1*L2,1)'*reshape(y_hat_test,L1*L2,1))/(norm(reshape(y_hat_test,L1*L2,1))*norm(reshape(y


out1 = trace(y_hat'*(A_op_2D(L1,L2,K,N,B_hat,C_hat,X0,1)));
out2 = trace(A_op_2D(L1,L2,K,N,B_hat,C_hat,y_hat,2)'*X0);
```

```
norm(out1-out2).^2/numel(out1) %MSE pixel by pixel



%% do minimization

% create ground truth

h = randn(K,1); %random FIR filter coefficients

m = scores(1,1:N).'; %use first sample from generated data and first N principal
↪  components

X0 = h*m';



w = B_op(L1,L2,K,h,1);

x = C_op(L1,L2,N,Phis,m,1);



%fully sampled in fourier domain

y_hat = (1/sqrt(L1*L2))*(fft2(w).*fft2(x));

y = sqrt(L1*L2)*ifft2(y_hat);


figure(1);

imshow(kron(y,ones(10)),[]);

pause(1);


%parameters for subsampling

M = round(0.75*L1*L2);

Omega = randperm(L1*L2,M)';
```

```
%operators for subsampling
ifftOp = linop_handles({[L1,L2],[L1,L2]}, @(x)ifft2(x)*sqrt(L1*L2),
↪  @(x)fft2(x)/sqrt(L1*L2) ,'C2C');
sampOp   = linop_compose( linop_subsample({[L1*L2,1],[M,1]},Omega),
↪  linop_reshape([L1,L2],[L1*L2,1]), ifftOp );


%inal lin_op
lin_op = linop_compose(sampOp, @(x,mode)A_op_2D(L1,L2,K,N,B_hat,C_hat,x,mode));


%subsampled data
z = sampOp(y_hat,1);



opts = tfocs_SCD;
opts.maxIts = 10000;
opts.tol = 1e-9;
opts.errFcn = @(f,z,x)error_measure(X0,f,z,x); % measure error between interations


X1 = tfocs_SCD(prox_nuclear,{lin_op, -z},prox_l2(1e-3),0.01,[],[],opts);


norm(X1-X0).^2 /numel(X0)


trace(X1'*X0)/(norm(X0)*norm(X1))


immse(X0,X1)
```

```
%% visualize our solution for h and m
% note that we will always be off by some unknown scaling factor


%svd to separate (almost) rank-1 matrix, remember X0 = h*m.'
[U, S, V] = svd(X1);
hnew = U(:,1); %left singular vector with largest singular value is h
mnew = V(:,1); %right singular vector with largest singular value is m


xnew = C_op(L1,L2,N,Phis,real(mnew),1); %reconstruct image with our solution



figure(2);
subplot(1,2,2)
imshow(kron(-1*xnew,ones(10)),[]);
title('Reconstruction')
subplot(1,2,1)
imshow(kron(x,ones(10)),[]);
title('Orignial')
pause(1);
```