Distribution Agreement

In presenting this thesis or dissertation as a partial fulfillment of the requirements for an advanced degree from Emory University, I hereby grant to Emory University and its agents the non-exclusive license to archive, make accessible, and display my thesis or dissertation in whole or in part in all forms of media, now or hereafter known, including display on the world wide web. I understand that I may select some access restrictions as part of the online submission of this thesis or dissertation. I retain all ownership rights to the copyright of the thesis or dissertation. I also retain the right to use in future works (such as articles or books) all or part of this thesis or dissertation.

Signature:

_____ _____

Chang Meng Date

Large Scale Inverse Problems: Low-Rank Approximations and Optimization

By

Chang Meng

Doctor of Philosophy

Mathematics

James G. Nagy

Advisor

Lars Ruthotto

Committee Member

Yuanzhe Xi

Committee Member

Accepted:

Kimberly Jacob Arriola, Ph.D., MPH

Dean of the James T. Laney School of Graduate Studies

Date

Large Scale Inverse Problems: Low-Rank Approximations and Optimization

By

Chang Meng

B.S., Emory University, 2016

Advisor: James G. Nagy, Ph.D.

An abstract of

A dissertation submitted to the Faculty of the

James T. Laney School of Graduate Studies of Emory University

in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

in Mathematics

2022

Abstract

Large Scale Inverse Problems: Low-Rank Approximations and Optimization

By Chang Meng

Inverse problems can be found in a variety of scientific applications, and the development of efficient and reliable methods remain an essential and challenging task. In this thesis, we introduce novel low-rank solvers for linear systems that arise from large scale inverse problems, which are usually ill-posed and require the use of regularization to obtain meaningful solutions. The new methods are developed around the concept of regularization: *i*) the low-rank, Kronecker product based forward model approximation method involves the approximation of a truncated singular value decomposition; and *ii*) the low-rank Krylov subspace methods are based on nuclear norm regularization. We explore the performance of these novel low-rank methods in various imaging applications such as image deblurring, inpainting and computer tomography. Besides applications where the forward model is known and fixed, we also consider an extended application, where the forward model is not exactly known and requires calibration. In this context, we are able to not only apply our new low-rank methods, but also propose a new hybrid machine learning and block coordinate descent algorithm that effectively improves solution accuracy.

Large Scale Inverse Problems: Low-Rank Approximations and Optimization

By

Chang Meng

B.S., Emory University, 2016

Advisor: James G. Nagy, Ph.D.

A dissertation submitted to the Faculty of the

James T. Laney School of Graduate Studies of Emory University

in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

in Mathematics

2022

Acknowledgments

Words cannot describe my gratitude for my advisor, Dr. James Nagy, who have supported and inspired me for many years. Without him, I wouldn't have had such a memorable time at Emory. Jim, thank you so much for being my advisor and always being supportive of my choices! I have learned so much from you, and that includes not only knowledge and research, but also how to become a better person. For me, you are both a great professor and a wonderful life mentor.

Also, many thanks to my committee members Lars Ruthotto and Yuanzhe Xi, who have provided so much guidance on my research. I would also like to thank every math teacher and professor I have ever had as a student for showing me the fantastic world of mathematics that shaped me into who I am.

Last but not least, I wouldn't be where I am today without my parents, who have given me unconditional love and support. Mom and dad, thank you for all the sacrifices you have made for me to receive better education, and thank you for being patient with me and letting me take my own road. I am so lucky to be your daughter!

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Inverse problems are ubiquitous in the areas of engineering, science and medical applications, and accurate solutions often require computationally costly or time consuming methods. This thesis focuses on a class of imaging inverse problems that can be modeled as

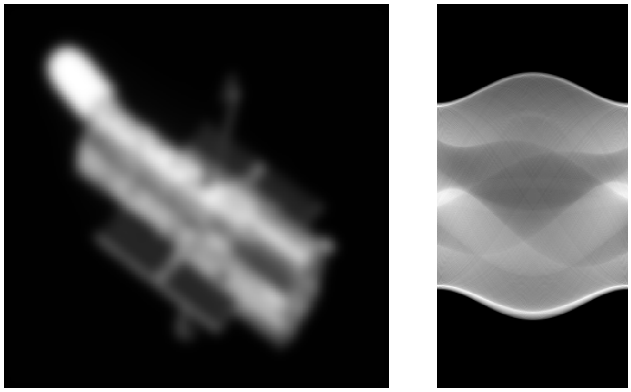
$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}, \tag{1.1}$$

where $\boldsymbol{A} \in \mathbb{R}^{M\times N}$ models the forward process, $\boldsymbol{x} \in \mathbb{R}^N$ is the image of interest, $\boldsymbol{b} \in \mathbb{R}^M$ is the observation that is usually contaminated with unknown noise $\boldsymbol{\eta} \in \mathbb{R}$, and $\boldsymbol{b} = \boldsymbol{b}^{\text{ex}} + \boldsymbol{\eta}$. In particular, the desired solution $\boldsymbol{x}^{\text{ex}}$ is the exact solution to $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}^{\text{ex}}$, where $\boldsymbol{b}^{\text{ex}}$ is the noise-free observation. The vectors $\boldsymbol{x}$ and $\boldsymbol{b}$ are obtained by vectorizing the true image $\boldsymbol{X} \in \mathbb{R}^{n\times n}$, $n = \sqrt{N}$ and the observed or measured

data $\boldsymbol{B}$, i.e.,

$$\boldsymbol{x} = \text{vec}(\boldsymbol{X}) \quad , \quad \boldsymbol{X} = \text{vec}^{-1}(\boldsymbol{x}),$$
$$\boldsymbol{b} = \text{vec}(\boldsymbol{B}) \quad , \quad \boldsymbol{B} = \text{vec}^{-1}(\boldsymbol{b}).$$

In the image deblurring application, the forward model $\boldsymbol{A}$ is constructed using a point spread function (PSF), which can be formulated based on knowledge of the physical process, and can be obtained using a precise mathematical expression or through experimentation [33]. For example, a two dimensional Gaussian function [39, 62] can be applied to construct the PSF for blurring caused by atmospheric turbulence. In this case, the observed data $\boldsymbol{B} = \text{vec}^{-1}(\boldsymbol{b})$ is a blurred image; see Figure 1.1(a) for an example. In imaging applications arising from computed tomography (CT), the matrix $\boldsymbol{A}$ models the Radon Transform [56] that outputs the projection data obtained from a tomographic scan. For CT, the observed data is a so-called "sinogram", which stores the projection data; see Figure 1.1(b) for an example.



(a) Blurred image. (b) CT sinogram.

Figure 1.1: Observed data.

Equivalently, we can rewrite (1.1) as the following least squares problem:

$$\min_{\boldsymbol{x}} \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2^2. \tag{1.2}$$

Large scale linear systems arising from inverse problems are usually very ill-conditioned, meaning that there might not be a unique solution, and a slight change in the data $\boldsymbol{b}$ can lead to a large change in the solution $\boldsymbol{x}$. Hence, regularization is often needed in order to obtain a meaningful solution. Direct regularization methods add a penalty term $\mathcal{R}(\boldsymbol{x})$ to the fit-to-data term $\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2^2$. The regularized optimization problem has the form

$$\min_{\boldsymbol{x}} \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2^2 + \mathcal{R}(\boldsymbol{x}), \tag{1.3}$$

where the regularization term $\mathcal{R}(\boldsymbol{x})$ is defined as a function of $\boldsymbol{x}$ (e.g. a vector norm). By doing so, we would be able to prevent extreme values in $\boldsymbol{x}$. Popular choices for $\mathcal{R}(\boldsymbol{x})$ include the ℓ_2 norm (Tikhonov regularization) and the ℓ_1 norm of $\boldsymbol{x}$. Various types of regularization approaches will be discussed in detail in Chapter 2.

When $\boldsymbol{A}$ is not too large, i.e., if its dimensions M and N are a few thousands or smaller, direct factorization methods such as the SVD factorization (short for Singular Value Decomposition, introduced in Chapter 2) can be efficiently used to solve the least squares problem (1.2), or (1.3) with Tikhonov regularization. However, as the dimension of $\boldsymbol{A}$ increases, the factorizations become increasingly difficult to compute (the complexity for computing the SVD of $\boldsymbol{A} \in \mathbb{R}^{N\times N}$ is $\mathcal{O}(N^3)$), but it is still possible to obtain their approximations. For example, while it is difficult

to compute the full SVD of large matrices, we may try to approximate the leading singular vectors and values to form a Truncated SVD (TSVD) factorization. The TSVD can be thought of as a low-rank approximation to matrix $\boldsymbol{A}$, and by discarding the smallest singular values and corresponding singular vectors, we are basically performing regularization by reducing the noise contribution of $\boldsymbol{\eta}$ in the solution $\boldsymbol{x}$.

An alternative approach for solving the least squares problem (1.2) when $\boldsymbol{A}$ is large is through iterative methods. The simplest example of iterative methods would be the Landweber iteration [46], which is essentially gradient descent applied to the function $f(x) = \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2^2$. Another class of iterative methods is called Krylov subspace methods, which look for solutions that belong to a Krylov subspace generated by $\boldsymbol{A}$ and $\boldsymbol{b}$. Examples of Krylov methods include Conjugate Gradient, LSQR and GMRES (see [64] and the references therein). What has been observed when solving linear systems with iterative methods is that they exhibit a semi-convergence property (will be illustrated in figures in Section 5.2), that is, the relative errors in the solutions decrease in the first iterations, but will increase in the further iterations when noise starts to corrupt the solutions. Hence, iterations need to be stopped at a suitable iteration index. In literature, hybrid methods (see [8, 12, 21]) have been proposed to combine iterative methods with regularization when needed. These hybrid methods can be particularly useful when combined with a non-traditional regularization term such as TV (total variation) regularization [19], or the low-rank enforcing nuclear norm regularization [18].

1.1 Contributions of Work

The main contribution of this thesis is the development of novel low-rank solvers for large scale linear systems. The meaning of "low rank" here is two-fold:

(1) low-rank approximation of forward model $\boldsymbol{A}$, or

(2) low-rank regularization for solution $\boldsymbol{x}$.

With respect to low-rank approximation of $\boldsymbol{A}$, we introduce the newly proposed Kronecker product summation based TSVD approximation method [15, 16]. This new method can be used to efficiently approximate a large number of leading singular values and vectors for large, structured square matrices from ill-posed inverse problems in the application of image deblurring. This novel TSVD approximation method is based on a reordering technique that incorporates information from more terms in the Kronecker summation compared to similar methods (see [41, 52]). Furthermore, theoretical bounds for the approximate TSVD operator and the approximate TSVD filtered solution are also developed.

Regarding low-rank regularization for $\boldsymbol{x}$, we describe new solvers for the computation of low-rank approximate solutions to large-scale linear problems. We are mainly concerned with algorithms that solve the so-called nuclear norm regularized problem, where a suitable nuclear norm penalization on the solution is imposed alongside a fit-to-data term expressed in the 2-norm (i.e., setting $\mathcal{R}(\boldsymbol{x})$ to be equal to the nuclear norm in (1.3)): this has the effect of implicitly enforcing low-rank solutions. By adopting an iteratively reweighted norm approach, the nuclear norm regularized problem is reformulated as a sequence of quadratic problems, which can then be effi-

ciently solved using Krylov methods, giving rise to an inner-outer iteration scheme. This approach differs from the other solvers available in the literature in that:

(a) Kronecker product properties are exploited to define the reweighted 2-norm penalization terms;

(b) efficient preconditioned Krylov methods replace gradient (projection) methods;

(c) the regularization parameter can be efficiently and adaptively set during the iterations.

Furthermore, we reformulate within the framework of flexible Krylov methods both the new inner-outer methods for nuclear norm regularization and some of the existing Krylov methods incorporating low-rank projections. This results in an even more computationally efficient (but heuristic) strategy, that does not rely on an inner-outer iteration scheme. This is the first time Krylov methods have been used to solve a low-rank, nuclear norm regularized problem.

The new low rank Krylov methods are competitive with other state-of-the-art solvers for low-rank problems including image deblurring, computed tomography and inpainting, and deliver reconstructions of increased quality with respect to other classical Krylov methods. In addition to applying the newly developed methods to (1.3), we also consider another least squares problem

$$\min_{\boldsymbol{x},\boldsymbol{p}} \|\boldsymbol{A}(\boldsymbol{p})\boldsymbol{x} - \boldsymbol{b}\|_2^2 + \mathcal{R}(\boldsymbol{x}), \tag{1.4}$$

where $\boldsymbol{p}$ is a set of parameters that $\boldsymbol{A}$ depends on. An example application in which this problem arises is computed tomography that requires geometry parameter

calibration. To solve (1.4), we could consider the framework of block coordinate descent (BCD), in which (1.4) is alternatively minimized with respect to $\boldsymbol{x}$ and $\boldsymbol{p}$ in each step. Since the minimization with respect to $\boldsymbol{x}$ requires a linear solver, our new low rank solvers can be readily used, and they show better convergence properties than hybrid Krylov solvers applied with traditional regularization terms such as ℓ_2.

We also propose a hybrid BCD method for solving (1.4) by incorporating machine learning. That is, we train a machine learning model $\widehat{\Phi}$ that maps $\boldsymbol{b}$ to $\boldsymbol{p}$, and when given an observation $\boldsymbol{b}$, we first use the ML model to make a prediction for $\boldsymbol{p}$, which is then fed into the BCD algorithm as an initial guess. Through this approach, we are able to obtain reconstructions that are of higher quality compared to using BCD or ML alone.

1.2 Outline of Thesis

This thesis is organized as follows. In Chapter 2, we provide an overview of regularization methods, including TSVD, Tikhonov and iterative regularization, as they are the foundation to the development of low-rank based methods, which is the main contribution of this thesis. In Chapter 3, we describe the new Kronecker product summation based approach for approximating the TSVD of $\boldsymbol{A}$. In addition to presenting how the approximation is derived, we also provide theoretical results that demonstrate the efficacy of the TSVD, both for the operator $\boldsymbol{A}$ and for the approximate solution to (1.1) filtered by the TSVD approximation. In Chapter 4, we introduce how to solve the nuclear norm regularized (NNR) using Krylov methods.

To be specific, we derive the iteratively reweighted norm (IRN) approach that solves a sequence of reformulated quadratic problems using Krylov methods, resulting in an inner-outer iteration algorithm. The IRN method is later reformulated within the flexible Krylov method framework that is considered more efficient (but heuristic). In Chapter 5, results of numerical experiments on several image processing problems are presented. In addition to comparing the newly developed low-rank Krylov methods to standard Krylov subspace methods, we also show comparisons with state-of-the-art solvers for low-rank problems, as well as existing Krylov methods incorporating low-rank projections reformulated as flexible Krylov subspace methods. In Chapter 6, we investigate the imaging application of computed tomography that requires parameter calibration, i.e., the forward model $\boldsymbol{A}$ depends on unknown parameters $\boldsymbol{p}$, which need to be calibrated alongside the image reconstruction process. We not only explore the application of the newly proposed low-rank methods in this context, but also investigate a new hybrid machine learning - BCD approach that solves the problem more accurately. We wrap up the thesis in Chapter 7, in which we conclude the findings of this thesis, as well as identify interesting research directions that can be extended from this work.

Chapter 2

Overview of Regularization Methods

Problems (1.1) and (1.2) are ill-conditioned because for large scale linear systems arising from imaging inverse problems, singular values of $\boldsymbol{A}$ usually decay slowly to 0, making the solution $\boldsymbol{x}$ very sensitive to noise in the observation $\boldsymbol{b}$. In this chapter, we show how the singular value decomposition (SVD) of $\boldsymbol{A}$ is related to the conditioning of the problem, discuss different types of regularization methods, and present techniques of choosing the regularization parameter. We start with a brief introduction of the SVD. Consider the SVD of $\boldsymbol{A} \in \mathbb{R}^{M\times N}$, which takes the form

$$\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T,$$

where $\boldsymbol{U} = [\boldsymbol{u}_1\ \boldsymbol{u}_2\ \cdots\ \boldsymbol{u}_M] \in \mathbb{R}^{M\times M}$ and $\boldsymbol{V} = [\boldsymbol{v}_1\ \boldsymbol{v}_2\ \cdots\ \boldsymbol{v}_N] \in \mathbb{R}^{N\times N}$ are orthogonal matrices, and $\boldsymbol{\Sigma} \in \mathbb{R}^{M\times N}$ is a diagonal matrix so that $\boldsymbol{\Sigma} = \operatorname{diag}(\sigma_1, \sigma_2, \cdots, \sigma_N)$,

where its diagonal entries are monotonically non-increasing: $\sigma_1 \geqslant \sigma_2 \geqslant \cdots \geqslant \sigma_N \geqslant 0$.

Without loss of generality, we assume that $M \geqslant N$, and the solution to (1.1) can be written as

$$\boldsymbol{x} = \boldsymbol{V}\boldsymbol{\Sigma}^{\dagger}\boldsymbol{U}^T\boldsymbol{b} = \sum_{i=1}^{N} \frac{\boldsymbol{u}_i^T\boldsymbol{b}}{\sigma_i}\boldsymbol{v}_i = \sum_{i=1}^{N} \frac{\boldsymbol{u}_i^T\boldsymbol{b}^{\text{ex}}}{\sigma_i}\boldsymbol{v}_i + \sum_{i=1}^{N} \frac{\boldsymbol{u}_i^T\boldsymbol{\eta}}{\sigma_i}\boldsymbol{v}_i,$$

where $\boldsymbol{\Sigma}^{\dagger} \in \mathbb{R}^{N\times M}$ is a diagonal matrix with diagonal entries $(1/\sigma_1, 1/\sigma_2, \cdots, 1/\sigma_N)$. We immediately observe that if σ_i is very small (i.e., for large indices i) compared to $\boldsymbol{\eta}$, then with σ_i on the denominator and $\boldsymbol{\eta}$ on the numerator, the noise contribution is going to be amplified by small σ_i's, and may even dominate over the true solution.

Regularization can be thought of as a way to "filter out" the noise contribution in the solution. Let $\phi_1, \phi_2, \cdots, \phi_N$ be the "filter factors" and define the regularized solution to be

$$\boldsymbol{x}_{\text{reg}} = \sum_{i=1}^{N} \phi_i \frac{\boldsymbol{u}_i^T\boldsymbol{b}}{\sigma_i}\boldsymbol{v}_i. \tag{2.1}$$

The filter factors ϕ_i should be chosen such that $0 \leqslant \phi_i \leqslant 1$ and ϕ_i tend to 0 as i increases so that for large i's, ϕ_i's can cancel out the noise contribution caused by small σ_i's. Two common regularization methods – Tikhonov regularization and Truncated SVD can both be conveniently expressed using filter factors.

2.1 Tikhonov Regularization and TSVD

Tikhonov Regularization In standard Tikhonov regularization, we have that $\mathcal{R}(\boldsymbol{x}) = \lambda\|\boldsymbol{x}\|_2^2$, $\lambda > 0$ in (1.3). Because of its definition, Tikhonov regularization is

also called ℓ_2 regularization. The regularized solution is the unique solution to

$$\min_{\boldsymbol{x}} \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2^2 + \lambda\|\boldsymbol{x}\|_2^2, \tag{2.2}$$

which is equivalent to solving the following least squares problem

$$\boldsymbol{x}_\lambda = \arg\min_{\boldsymbol{x}} \left\| \begin{bmatrix} \boldsymbol{A}\boldsymbol{x} \\ \sqrt{\lambda}\boldsymbol{I} \end{bmatrix} - \begin{bmatrix} \boldsymbol{b} \\ \boldsymbol{0} \end{bmatrix} \right\|_2^2, \tag{2.3}$$

or its corresponding normal equations

$$(\boldsymbol{A}^T\boldsymbol{A} + \lambda\boldsymbol{I})\boldsymbol{x} = \boldsymbol{A}^T\boldsymbol{b}.$$

Note that Tikhonov regularization does not require us to compute or approximate the SVD of $\boldsymbol{A}$. However, the regularization parameter λ needs to be chosen beforehand, and ideally, we should take $\sigma_N \leqslant \lambda \leqslant \sigma_1$. It can be shown that the filter factors for Tikhonov regularization are $\phi_i = \frac{\sigma_i^2}{\sigma_i^2+\lambda}$. By introducing the filter factors, we can express $\boldsymbol{x}_\lambda$ as

$$\boldsymbol{x}_\lambda = \boldsymbol{V}\boldsymbol{\Sigma}_\lambda^\dagger\boldsymbol{U}^T\boldsymbol{b}, \tag{2.4}$$

where $\boldsymbol{\Sigma}_\lambda^\dagger = \mathrm{diag}(\phi_1/\sigma_1, \cdots, \phi_N/\sigma_N) = \mathrm{diag}(\frac{\sigma_1}{\sigma_1^2+\lambda}, \cdots, \frac{\sigma_N}{\sigma_N^2+\lambda})$.

Tikhonov regularization may be generalized to define $\mathcal{R}(\boldsymbol{x}) = \lambda\|\boldsymbol{L}\boldsymbol{x}\|_2^2$. Sometimes, we introduce the additional regularization matrix $\boldsymbol{L}$ because certain properties are desired for the solution $\boldsymbol{x}_\lambda$. For example, we can take $\boldsymbol{L}$ to be a discretization of the derivative operator, and in this way, we are implicitly applying a "smoothness"

constraint to the solution. By applying the regularization matrix $\boldsymbol{L}$, the least squares problem is

$$\boldsymbol{x}_\lambda = \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2^2 + \lambda\|\boldsymbol{L}\boldsymbol{x}\|_2^2, \tag{2.5}$$

and equivalently,

$$\boldsymbol{x}_\lambda = \arg\min_{\boldsymbol{x}} \left\| \begin{bmatrix} \boldsymbol{A}\boldsymbol{x} \\ \sqrt{\lambda}\boldsymbol{L} \end{bmatrix} - \begin{bmatrix} \boldsymbol{b} \\ \boldsymbol{0} \end{bmatrix} \right\|_2^2. \tag{2.6}$$

The associated normal equations are

$$(\boldsymbol{A}^T\boldsymbol{A} + \lambda\boldsymbol{L}^T\boldsymbol{L})\boldsymbol{x} = \boldsymbol{A}^T\boldsymbol{b}.$$

Note that (2.5) can be solved by substitution of variable $\widehat{\boldsymbol{x}} = \boldsymbol{L}\boldsymbol{x}$ and $\boldsymbol{x} = \boldsymbol{L}^{-1}\widehat{\boldsymbol{x}}$ under the assumption that $\boldsymbol{L}$ is invertible. By doing so, the regularized problem becomes

$$\boldsymbol{x}_\lambda = \|\boldsymbol{A}\boldsymbol{L}^{-1}\widehat{\boldsymbol{x}} - \boldsymbol{b}\|_2^2 + \lambda\|\widehat{\boldsymbol{x}}\|_2^2, \tag{2.7}$$

and the generalized Tikhonov regularized problem can be reduced to a standard Tikhonov problem.

Both standard and general Tikhonov regularized solutions can be obtained rather easily by using standard linear system solvers. If $\boldsymbol{A}$ is small (which is rarely the case), we can use direct solvers – a good example would be MATLAB's backslash operator applied to (2.3) and (2.6), which uses QR factorization to solve the rectangular system. This is feasible when $\boldsymbol{A}$ is small, but in most cases that arise in image processing, the matrix $\boldsymbol{A}$ is very large, requiring us to use an iterative solver such as

Krylov subspace methods.

Truncated SVD. TSVD is implemented by setting small singular values of $\boldsymbol{A}$ to 0 directly. Hence, the TSVD solution is given by

$$\boldsymbol{x}_{\mathrm{TSVD}} = \sum_{i=1}^{k} \frac{\boldsymbol{u}_i^T \boldsymbol{b}}{\sigma_i} \boldsymbol{v}_i, \tag{2.8}$$

where $k \leqslant N$ is the truncation rank that needs to be chosen. This solution is equivalent to applying the following filter factors to (2.1):

$$\phi_i = \begin{cases} 1 & \text{for } i \leqslant k, \\ 0 & \text{for } i > k. \end{cases}$$

By defining the filter factors, we can rewrite (2.8) as

$$\boldsymbol{x}_{\mathrm{TSVD}} = \sum_{i=1}^{N} \phi_i \frac{\boldsymbol{u}_i^T \boldsymbol{b}}{\sigma_i} \boldsymbol{v}_i \quad \text{and} \quad \boldsymbol{x}_{\mathrm{TSVD}} = \boldsymbol{V} \boldsymbol{\Sigma}_{\mathrm{TSVD}}^{\dagger} \boldsymbol{U}^T \boldsymbol{b},$$

where $\boldsymbol{\Sigma}_{\mathrm{TSVD}}^{\dagger} = \mathrm{diag}(\phi_1/\sigma_1, \cdots, \phi_N/\sigma_N) = \mathrm{diag}(1/\sigma_1, \cdots, 1/\sigma_k, 0, \cdots, 0)$.

An important question associated with using TSVD is: how should we choose the truncation rank k? Unfortunately, there is no perfect answer to this question. The truncation rank k depends on the individual problem and there is no best way to choose it for all problems. But there are guides that can be used to estimate k. In Section 2.2, we introduce techniques for selecting λ for Tikhonov regularization, and these techniques can be easily extended to TSVD. In practice, it is extremely

hard to compute the full SVD of large matrices $\boldsymbol{A}$. But we can still apply TSVD regularization by approximating the leading (largest) singular values and vectors, which will be further discussed in Chapter 3.

2.2 Choosing the Regularization Parameter for Tikhonov Regularization

Techniques for choosing the regularization parameter for Tikhonov regularization include the L-curve method [34], generalized cross validation and the discrepancy principle [30]. These techniques require evaluating the residual

$$\|\boldsymbol{A}\boldsymbol{x}_\lambda - \boldsymbol{b}\|_2 = \|\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{\Sigma}_\lambda^\dagger \boldsymbol{U}^T\boldsymbol{b} - \boldsymbol{b}\|_2 = \|\boldsymbol{\Sigma}\boldsymbol{\Sigma}_\lambda^\dagger \boldsymbol{b} - \boldsymbol{U}^T\boldsymbol{b}\|_2 = \sum_{i=1}^{N} \frac{\lambda \widehat{b}_i}{\sigma_i^2 + \lambda},$$

where $\widehat{\boldsymbol{b}} = \boldsymbol{U}^T\boldsymbol{b}$. In addition to going over these three methods, we also explain how to find the optimal regularization parameter, which requires us to know $\boldsymbol{x}^{\text{ex}}$ and can be used for analysis purposes, but it is not so realistic in real imaging applications.

The L-curve method. This method is first proposed in [34], and requires solving the Tikhonov regularized problem (2.2) repeatedly, each time with a different regularization parameter λ_k. The range of λ_k should depend on the singular spectrum of $\boldsymbol{A}$, and an example would be taking equally spaced points between 10^{-8} and 10^{-3}. For each solution $\boldsymbol{x}_\lambda$ obtained by solving the Tikhonov regularized problem with λ_k, we plot the residual $\boldsymbol{r}_k = \|\boldsymbol{A}\boldsymbol{x}_\lambda - \boldsymbol{b}\|_2$ against the norm of the solution $\|\boldsymbol{x}_\lambda\|_2$,

which should result in an L-shaped curve. The λ_k corresponding to the "corner" of the L-curve renders a solution that has a small residual and whose norm is small, corresponding to the fit-to-data term and norm penalty term in the least squares problem (2.2) or (2.7). The `lcornor.m` function from Regularization Tools [29] can be used to locate the corner of the L-curve.

Generalized Cross Validation (GCV). As described in [30], the GCV method looks for the regularization parameter λ that minimizes the GCV function

$$G(\lambda) = \frac{N\|\boldsymbol{A}\boldsymbol{x}_\lambda - \boldsymbol{b}\|_2^2}{\Big(\operatorname{Tr}(\boldsymbol{I} - \boldsymbol{A}\boldsymbol{A}_\lambda^\dagger)\Big)^2} = \frac{N\|\boldsymbol{\Sigma}\boldsymbol{\Sigma}_\lambda^\dagger \boldsymbol{b} - \boldsymbol{U}^T\boldsymbol{b}\|_2^2}{\Big(\operatorname{Tr}(\boldsymbol{I} - \boldsymbol{\Sigma}\boldsymbol{\Sigma}_\lambda^\dagger)\Big)^2} = \frac{N\Big(\sum_{i=1}^N \frac{\widehat{b}_i}{\sigma_i^2+\lambda}\Big)^2}{\Big(\sum_{i=1}^N \frac{1}{\sigma_i^2+\lambda}\Big)^2},$$

where $\boldsymbol{A}_\lambda^\dagger = \boldsymbol{V}^T\boldsymbol{\Sigma}_\lambda^\dagger\boldsymbol{U}^T$ as in (2.4), and the trace of $\boldsymbol{I} - \boldsymbol{\Sigma}\boldsymbol{\Sigma}_\lambda^\dagger$ can be easily computed since it is a diagonal matrix. The minimizer λ of $G(\lambda)$ can be found using MATLAB's `fminbnd` function, by inputting an approximate bound for λ, for example, $\lambda_{\min} = 0$ and $\lambda_{\max} =$ an approximation of σ_1.

The Discrepancy Principle. To solve the original linear system (1.1) of interest using Tikhonov regularization, we would like our solution $\boldsymbol{x}_\lambda$ to satisfy $\boldsymbol{A}\boldsymbol{x}_\lambda = \boldsymbol{b}^{\text{ex}} + \boldsymbol{\eta}$, so naturally, we have the relation $\boldsymbol{A}\boldsymbol{x}_\lambda - \boldsymbol{b} = \boldsymbol{\eta}$, which is called the discrepancy. The discrepancy principle [30] assigns the regularization parameter λ such that the following equation is satisfied

$$\|\boldsymbol{A}\boldsymbol{x}_\lambda - \boldsymbol{b}\|_2 = \|\boldsymbol{\eta}\|_2 \quad \text{i.e.,} \quad \sum_{i=1}^N \frac{\lambda\widehat{b}_i}{\sigma_i^2 + \lambda} - \|\boldsymbol{\eta}\|_2 = 0. \tag{2.9}$$

This approach requires that the norm, $\|\boldsymbol{\eta}\|_2$, of the Gaussian white noise $\boldsymbol{\eta}$ is available to us. Since $\|\boldsymbol{\eta}\|_2$ cannot be exactly known, in practice the discrepancy principle is implemented as $\|\boldsymbol{A}\boldsymbol{x}_\lambda - \boldsymbol{b}\|_2 = \mu\|\boldsymbol{\eta}\|_2$, where $\mu \gtrapprox 1$ is a user chosen parameter, e.g., $\mu = 1.01$. MATLAB's `fzero` function may be used to solve this equation for the unknown λ.

The Optimal Parameter. Choosing the optimal regularization parameter requires the true solution $\boldsymbol{x}^{\text{ex}}$, which differentiates this last method from the previous three. We introduce this method in order to provide a basis for comparison with other parameter setting techniques in numerical experiments. To find the best regularization parameter, we try to minimize the difference between $\boldsymbol{x}^{\text{ex}}$ and $\boldsymbol{x}_\lambda$:

$$\begin{aligned} \|\boldsymbol{x}^{\text{ex}} - \boldsymbol{x}_\lambda\|_2^2 &= \|\boldsymbol{x}^{\text{ex}} - \boldsymbol{V}\boldsymbol{\Sigma}_\lambda^\dagger \boldsymbol{U}^T\boldsymbol{b}\|_2^2 \\ &= \|\boldsymbol{V}^T\boldsymbol{x}^{\text{ex}} - \boldsymbol{\Sigma}_\lambda^\dagger \boldsymbol{U}^T\boldsymbol{b}\|_2^2 \\ &= \|\widehat{\boldsymbol{x}}^{\text{ex}} - \boldsymbol{\Sigma}_\lambda^\dagger \widehat{\boldsymbol{b}}\|_2^2 \\ &= \sum_{i=1}^{N} \Big(\frac{\sigma_i \widehat{b}_i}{\sigma_i^2 + \lambda} - \widehat{x}_i\Big)^2, \end{aligned} \tag{2.10}$$

where $\widehat{\boldsymbol{b}} = \boldsymbol{U}^T\boldsymbol{b}$ and $\widehat{\boldsymbol{x}}^{\text{ex}} = \boldsymbol{V}\boldsymbol{x}^{\text{ex}}$. We need to look for the λ that minimizes (2.10), and similar to GCV, we can use MATLAB's `fminbnd` function for this purpose.

2.3 Other Regularization Terms

There are many other ways to define the regularization term $\mathcal{R}(\boldsymbol{x})$. Depending on the desired properties in the solution $\boldsymbol{x}$, we can choose $\mathcal{R}(\boldsymbol{x})$ accordingly. All regu-

larization methods serve the purpose of reducing the ill-conditioning in the problem. Popular choices include:

- ℓ_1 regularization: $\mathcal{R}(\boldsymbol{x}) = \|\boldsymbol{x}\|_1$ is used to enforce sparsity in the solution;
- ℓ_p regularization: $\mathcal{R}(\boldsymbol{x}) = \|\boldsymbol{x}\|_p^p$ is a generalization of ℓ_1 and ℓ_2 regularization;
- Nuclear norm regularization: $\mathcal{R}(\boldsymbol{x}) = \|\text{vec}^{-1}(\boldsymbol{x})\|_* = \|\boldsymbol{X}\|_*$ is defined as the sum of singular values of $\boldsymbol{X} = \text{vec}^{-1}(\boldsymbol{x})$, which enforces a low rank constraint implicitly;
- Total variation (TV) regularization: $\mathcal{R}(\boldsymbol{x}) = TV(\boldsymbol{x}) = \sum_{i=1}^{N} \sqrt{[D_h\boldsymbol{x}]_i^2 + [D_v\boldsymbol{x}]_i^2}$, where D_h and D_v denote the finite difference approximations of the horizontal and vertical first derivative operators; it is widely used in image restoration applications, and is very useful in denoising.

While many methods exist for computing the solution to different types of regularized problem, we comment briefly here on one kind of method called "Iteratively Reweighted Norm" (IRN) [25, 61]. The idea is to express different regularization terms $\mathcal{R}(\boldsymbol{x})$ in terms of $\|\boldsymbol{L}(\boldsymbol{x})\boldsymbol{x}\|_2^2$. For example, for ℓ_1 regularization $\boldsymbol{L}(\boldsymbol{x}) = \text{diag}\,(1/\sqrt{\boldsymbol{x}})$, where square root and division are done componentwise.

We immediately notice that the difficulty of this approach lies with the computation of the regularization matrix $\boldsymbol{L}(\boldsymbol{x})$ defined with respect to the solution $\boldsymbol{x}$, which is not available to us. A natural remedy is to use an approximation of $\boldsymbol{x}$ to compute $\boldsymbol{L}(\boldsymbol{x})$, which leads us to Algorithm 1. This algorithm is often referred to as iterative reweighted norm (IRN), which utilizes an inner-outer iteration approach: in the ith outer iteration, the approximation $\boldsymbol{L}_i$ (to be used in place of the true $\boldsymbol{L}(\boldsymbol{x})$)

is computed using the solution $\boldsymbol{x}_{i-1}$ from the previous outer iteration; we then solve the general Tikhonov regularized problem with regularization matrix $\boldsymbol{L}_i$ to obtain the updated solution $\boldsymbol{x}_i$ by using an iterative method (i.e., inner iterations). Note that some elements of $\boldsymbol{x}_i$ may be zero, in which case we may have a division by zero problem depending on the definition of $\boldsymbol{L}(\boldsymbol{x})$ (e.g., $\boldsymbol{L}(\boldsymbol{x}) = \text{diag}\,(1/\sqrt{\boldsymbol{x}})$). This can be avoided by adding a small constant to $\boldsymbol{x}_i$.

Algorithm 1 Iteratively Reweighted Norm

Inputs: $\boldsymbol{A}$, $\boldsymbol{b}$, $\boldsymbol{L}_0 = \boldsymbol{L}(\boldsymbol{x}_0)$, λ
for $i = 1, 2, \cdots$ until a stopping criterion is satisfied **do**
  Solve for $\boldsymbol{x}_i = \arg\min_{\boldsymbol{x}} \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2^2 + \lambda\|\boldsymbol{L}_i\boldsymbol{x}\|_2^2$
  Update $\boldsymbol{L}_i = \boldsymbol{L}(\boldsymbol{x}_{i-1})$
end for
Output: $\boldsymbol{x}_i$'s

Many types of regularization terms can be expressed in a similar way, including ℓ_p regularization for a general p value, TV regularization [19] and nuclear norm regularization [18]. Therefore, the IRN approach can be applied to a wide range of regularized least squares problems in a similar way. In Chapter 4, we elaborate on how to apply IRN to the nuclear norm regularized (NNR) least squares problem which enforces a low rank constraint. In particular, we use Krylov subspace methods (i.e., GMRES and LSQR) as inner solvers for IRN. One disadvantage of the IRN approach is that the inner-outer iteration scheme is not very computationally efficient, since we need to solve many least squares problems in a sequence. So a more efficient alternative – flexible Krylov subspace methods [11, 63] will also be discussed.

2.4 Iterative Regularization

Iterative regularization is different from direct regularization that adds a penalization term $\mathcal{R}(\boldsymbol{x})$ directly. Instead, it exploits the semi-convergence property of iterative methods, and early stopping is performed when semi-convergence is achieved. That is, we apply an iterative method to the linear system $\boldsymbol{Ax} = \boldsymbol{b}$, and terminate iterations when a good solution is obtained, before noise starts to contaminate the solution. Early stopping computes a solution that resembles TSVD regularization [31, 69].

A number of iterative methods can be used for this purpose. In this section, we address Krylov subspace methods, namely, GMRES and LSQR (discussed in Section 2.4.1). Implementations of these two methods can be found in MATLAB's library (functions `gmres.m` and `lsqr.m`). Their hybrid counterparts (discussed in Section 2.4.2), i.e., hybrid GMRES and hybrid LSQR [8, 12, 21] can be considered if we want to combine Krylov subspace methods with Tikhonov regularization. Implementations of the hybrid methods can be found in the MATLAB package IR Tools [17]. Note that in this section, $\boldsymbol{U}$ and $\boldsymbol{V}$ no longer represent the singular vector matrices as in the previous sections of this chapter.

2.4.1 GMRES and LSQR

The Generalized Minimum Residual (GMRES) method is proposed by Saad and Schultz in 1986 [65]. It is an iterative method that aims at solving large scale linear systems with square matrices $\boldsymbol{A} \in \mathbb{R}^{N \times N}$. At the mth iteration of GMRES, the

approximate solution is $\boldsymbol{x}_k \in \mathcal{K}_k(\boldsymbol{A}, \boldsymbol{b})$ under the assumption that the initial guess $\boldsymbol{x}_0 = \boldsymbol{0}$, where $\mathcal{K}_k(\boldsymbol{A}, \boldsymbol{b})$ is the Krylov subspace

$$\mathcal{K}_k(\boldsymbol{A}, \boldsymbol{b}) = \text{span}\{\boldsymbol{b}, \boldsymbol{A}\boldsymbol{b}, \boldsymbol{A}^2\boldsymbol{b}, \cdots, \boldsymbol{A}^k\boldsymbol{b}\}.$$

Let $\boldsymbol{V}_k = [\boldsymbol{v}_1, \cdots, \boldsymbol{v}_k] \in \mathbb{R}^{N\times k}$ be the orthonormal basis for the Krylov subspace $\mathcal{K}_k(\boldsymbol{A}, \boldsymbol{b})$. Specifically, $\boldsymbol{v}_1 = \boldsymbol{b}/\|\boldsymbol{b}\|$. Since the solution $\boldsymbol{x}_k$ is in $\mathcal{K}_k(\boldsymbol{A}, \boldsymbol{b})$, it can be written as $\boldsymbol{V}_k\boldsymbol{y}$ for some coefficients $\boldsymbol{y}$. GMRES first uses the Arnoldi algorithm to generate $\boldsymbol{V}_k \in \mathbb{R}^{N\times(k+1)}$ whose columns are orthonormal, and an upper Hessenberg matrix $\boldsymbol{H}_k \in \mathbb{R}^{(k+1)\times k}$ such that

$$\boldsymbol{A}\boldsymbol{V}_k = \boldsymbol{V}_{k+1}\boldsymbol{H}_k.$$

Then,

$$\begin{aligned} \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2^2 &= \|\boldsymbol{A}\boldsymbol{V}_k\boldsymbol{y} - \boldsymbol{b}\|_2^2 \\ &= \|\boldsymbol{V}_{k+1}\boldsymbol{H}_k\boldsymbol{y} - \|\boldsymbol{b}\|\boldsymbol{v}_1\|_2^2 \\ &= \|\boldsymbol{V}_{k+1}(\boldsymbol{H}_k\boldsymbol{y} - \|\boldsymbol{b}\|\boldsymbol{e}_1)\|_2^2 \\ &= \|\boldsymbol{H}_k\boldsymbol{y} - \|\boldsymbol{b}\|\boldsymbol{e}_1\|_2^2, \end{aligned} \tag{2.11}$$

where $\boldsymbol{e}_1 = (1, 0, \cdots, 0) \in \mathbb{R}^{k+1}$. Hence the GMRES solution $\boldsymbol{x}_k \in \mathcal{K}_k(\boldsymbol{A}, \boldsymbol{b})$ can be easily obtained by solving the projected minimization problem (2.11) for the minimizer $\boldsymbol{y}_k$, which is usually inexpensive to compute. That is, the solution $\boldsymbol{x}_k = \boldsymbol{V}\boldsymbol{y}_k$, where $\boldsymbol{y}_k = \arg\min_{\boldsymbol{y}} \|\boldsymbol{H}_k\boldsymbol{y} - \|\boldsymbol{b}\|\boldsymbol{e}_1\|_2$, which leads to the GMRES algorithm

(Algorithm 2).

Algorithm 2 GMRES

1: Inputs: $\boldsymbol{A}$, $\boldsymbol{b}$
2: Take $\boldsymbol{v}_1 = \boldsymbol{b}/\|\boldsymbol{b}\|_2$
3: **for** $i = 1, 2, \ldots$ until a stopping criterion is satisfied **do**
4: Compute $\boldsymbol{w} = A\boldsymbol{v}_i$
5: Compute $h_{ji} = \boldsymbol{w}^T\boldsymbol{v}_j$ for $j = 1, \ldots, i$ and set $\boldsymbol{w} = \boldsymbol{w} - \sum_{j=1}^{i} h_{ji}\boldsymbol{v}_j$
6: Set $h_{i+1,i} = \|\boldsymbol{w}\|_2$, and if $h_{j+1,j} \neq 0$, take $\boldsymbol{v}_{i+1} = \boldsymbol{w}/h_{i+1,i}$
7: **end for**
8: Compute $\boldsymbol{y}_k = \arg\min_{\boldsymbol{y}} \|\boldsymbol{H}_k\boldsymbol{y} - \|\boldsymbol{b}\|_2\boldsymbol{e}_1\|_2^2$ and take $\boldsymbol{x}_k = \boldsymbol{V}_k\boldsymbol{y}_k$

Another example of Krylov subspace methods is LSQR, which is proposed by Paige and Sanders [54] to solve large and sparse linear systems. Different from GMRES that can only be applied to square matrices $\boldsymbol{A} \in \mathbb{R}^{N\times N}$, LSQR can be used on general rectangular shaped matrices $\boldsymbol{A} \in \mathbb{R}^{M\times N}$, where M is not necessarily equal to N. It is based on Golub-Kahan Bidiagonalization (GKB) [24] and the computed solution $\boldsymbol{x}_k$ belongs to the Krylov subspace $\mathcal{K}_k(\boldsymbol{A}^T\boldsymbol{A}, \boldsymbol{A}^T\boldsymbol{b})$ spanned by columns of $\boldsymbol{V}_k$. The kth iteration of GKB generates $\boldsymbol{V}_k = [\boldsymbol{v}_1, \cdots, \boldsymbol{v}_k] \in \mathbb{R}^{N\times k}$ and $\boldsymbol{U}_{k+1} = [\boldsymbol{u}_1, \cdots, \boldsymbol{u}_{k+1}] \in \mathbb{R}^{M\times(k+1)}$ with orthonormal columns, and lower bidiagonal matrix $\boldsymbol{B}_{k+1} \in \mathbb{R}^{(k+1)\times k}$ such that

$$\boldsymbol{A}\boldsymbol{V}_k = \boldsymbol{U}_{k+1}\boldsymbol{B}_k.$$

Since the solution $\boldsymbol{x} \in \mathcal{R}(\boldsymbol{V}_k)$, by letting $\boldsymbol{x} = \boldsymbol{V}_k\boldsymbol{y}$, it follows that

$$\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2^2 = \|\boldsymbol{B}_k\boldsymbol{y} - \|\boldsymbol{b}\|\boldsymbol{e}_1\|_2^2. \tag{2.12}$$

Steps to achieve (2.12) are omitted because they resemble those for deriving (2.11). Therefore, LSQR computes the solution $\boldsymbol{x}_k = \boldsymbol{V}_k \boldsymbol{y}_k$ for $\boldsymbol{y}_k$ that solves $\min_{\boldsymbol{y}} \|\boldsymbol{B}_k \boldsymbol{y} - \|\boldsymbol{b}\|\boldsymbol{e}_1\|_2$. The LSQR algorithm is presented in Algorithm 3.

Algorithm 3 LSQR

Inputs: $\boldsymbol{A}$, $\boldsymbol{b}$
Take $\boldsymbol{u}_1 = \boldsymbol{b}/\|\boldsymbol{b}\|_2$
for $i = 1, 2, \ldots,$ until a stopping criterion is satisfied **do**
    Compute $\boldsymbol{w} = \boldsymbol{A}^T \boldsymbol{u}_i$ and $\boldsymbol{w} = \boldsymbol{w} - \beta_{i-1} \boldsymbol{v}_{i-1}$ if $i > 1$
    Set $\alpha_i = \|\boldsymbol{w}\|_2$ and $\boldsymbol{v}_i = \boldsymbol{w}/\alpha_i$
    Compute $\boldsymbol{w} = \boldsymbol{A}^T \boldsymbol{v}_i$, $\boldsymbol{w} = \boldsymbol{w} - \alpha_i \boldsymbol{u}_i$
    Set $\beta_i = \|\boldsymbol{w}\|_2$ and $\boldsymbol{u}_{i+1} = \boldsymbol{w}/\beta_i$
    Let $b_{ii} = \alpha_i$ and $b_{i+1,i} = \beta_i$ and if $\alpha_i = 0$ or $\beta_i = 0$, break
end for

Compute $\boldsymbol{y}_k = \arg\min_{\boldsymbol{y}} \|\boldsymbol{B}_k \boldsymbol{y} - \|\boldsymbol{b}\|_2 \boldsymbol{e}_1\|_2^2$ and take $\boldsymbol{x}_k = \boldsymbol{V}_k \boldsymbol{y}_k$

2.4.2 Hybrid Krylov Methods

Krylov subspace methods such as GMRES are applied to the fit-to-data term $\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2^2$ alone, and iterative regularization is done by early stopping. If we want to apply additional regularization such as Tikhonov, we can adopt hybrid Krylov methods. Hybrid GMRES employs the Arnoldi-Tikhonov scheme [19], and rather than adding the regularization term directly to $\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2^2$, it solves a regularized projected problem, i.e., the regularization term is applied to the projected problem $\|\boldsymbol{H}_k \boldsymbol{y} - \|\boldsymbol{b}\|\boldsymbol{e}_1\|_2^2$. This is legitimate because we have shown that $\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2^2 = \|\boldsymbol{H}_k \boldsymbol{y} - \|\boldsymbol{b}\|\boldsymbol{e}_1\|_2^2$, hence

$$\min_{\boldsymbol{y}} \|\boldsymbol{A}\boldsymbol{V}_k \boldsymbol{y} - \boldsymbol{b}\|_2^2 + \lambda \|\boldsymbol{x}\|_2^2$$

is equivalent to

$$\min_{\boldsymbol{y}} \|\boldsymbol{H}_k\boldsymbol{y} - \|\boldsymbol{b}\|\boldsymbol{e}_1\|_2^2 + \lambda\|\boldsymbol{x}\|_2^2, \tag{2.13}$$

where we have substituted $\boldsymbol{x} = \boldsymbol{V}_k\boldsymbol{y}$. Therefore, we obtain the hybrid GMRES algorithm, which simply replaces the minimization problem in step 8 of Algorithm 2 by (2.13).

Similar to hybrid GMRES, hybrid LSQR can be considered if additional regularization is needed on top of iterative regularization. In the case of hybrid LSQR, direct regularization is applied to the projected problem (2.12) so that in step 10 of Algorithm 3, the following regularized projected minimization problem is solved instead

$$\boldsymbol{y}_k = \arg\min_{\boldsymbol{y}} \|\boldsymbol{B}_k\boldsymbol{y} - \|\boldsymbol{b}\|_2\boldsymbol{e}_1\|_2^2 + \lambda\|\boldsymbol{y}\|_2^2. \tag{2.14}$$

Choosing λ for hybrid Krylov methods To choose a regularization parameter λ for (2.13), one obvious choice would be to use parameter setting techniques described in Section 2.2. However, in the case of discrepancy principle, Gazzola and Novati [20] have proposed a secant approach that can be applied readily to Arnoldi-Tikhonov based methods. The approach exploits the discrepancy principle to look for the λ such that

$$\|\boldsymbol{A}\boldsymbol{x}_\lambda - \boldsymbol{b}\|_2 = \|\boldsymbol{H}_k\boldsymbol{y}_\lambda - \|\boldsymbol{b}\|\boldsymbol{e}_1\|_2 = \|\boldsymbol{\eta}\|_2$$

holds. While we can still solve for λ by running MATLAB's `fzero` function on (2.9), an alternative approach has been proposed in [20], which can efficiently compute a regularization parameter λ_i for each GMRES iteration, and can be employed as a stopping criteria. This discrepancy principle based approach starts with defining the

discrepancy function

$$\psi_i(\lambda) = \|\boldsymbol{A}\boldsymbol{x}_\lambda - \boldsymbol{b}\|_2 = \|\boldsymbol{H}_i\boldsymbol{y}_{i,\lambda} - \|\boldsymbol{b}\|\boldsymbol{e}_1\|_2, \tag{2.15}$$

where $\boldsymbol{y}_{i,\lambda}$ is the solution to (2.13) using regularization parameter λ with $i = k$. Assuming that a good estimate $\epsilon \approx \|\boldsymbol{\eta}\|_2$ is available, we say that the discrepancy principle is satisfied as soon as

$$\psi(\lambda) \leqslant \mu\epsilon, \quad \text{for} \quad \mu \gtrapprox 1.$$

At each iteration i, consider the linear approximation of $\psi(\lambda)$

$$\psi_i(\lambda) \approx \psi_i(0) + \beta_i\lambda, \quad \text{where} \quad \beta_i = \frac{\psi_i(\lambda_{i-1}) - \psi_i(0)}{\lambda_{i-1}}.$$

The expression for β_i is obtained from the equation $\psi_i(\lambda_{i-1}) = \psi_i(0) + \beta_i\lambda_{i-1}$, where $\psi_i(\lambda_{i-1})$ is available from the previous iteration, and $\psi_i(0)$ is the norm of the GMRES residual at iteration i and can be easily computed with the SVD of $\boldsymbol{H}_i$. To select λ_i, we then force $\psi_i(\lambda_i) = \psi_i(0) + \beta_i\lambda_i$, and by substituting β_i, we get

$$\lambda_i = \left|\frac{\mu\epsilon - \psi_i(0)}{\psi_i(\lambda_{i-1}) - \psi_i(0)}\right| \lambda_{i-1}. \tag{2.16}$$

The absolute value is taken to avoid selecting a negative λ_i. Note that the SVD of $\boldsymbol{H}_i$, which was previously used to compute $\psi_i(0)$, can again be used here to solve for $\boldsymbol{y}_{i,\lambda_i}$. Using the discrepancy principle, iterations can be terminated as soon as $\psi_i(\lambda) \leqslant \mu\epsilon$. The updated hybrid GMRES using discrepancy principle as a

regularization technique and stopping criteria is summarized in Algorithm 4.

Algorithm 4 Hybrid GMRES with discrepancy principle

Inputs: $\boldsymbol{A}$, $\boldsymbol{b}$
Take $\boldsymbol{v}_1 = \boldsymbol{b}/\|\boldsymbol{b}\|_2$
for $i = 1, 2, \ldots$ until the discrepancy principle is satisfied **do**
    Compute $\boldsymbol{w} = A\boldsymbol{v}_i$
    Compute $h_{ji} = \boldsymbol{w}^T\boldsymbol{v}_j$ for $j = 1, \ldots, i$ and set $\boldsymbol{w} = \boldsymbol{w} - \sum_{j=1}^{i} h_{ji}\boldsymbol{v}_j$
    Set $h_{i+1,i} = \|\boldsymbol{w}\|_2$, and if $h_{j+1,j} \neq 0$, take $\boldsymbol{v}_{i+1} = \boldsymbol{w}/h_{i+1,i}$
    Select λ_i using (2.16)
    Compute $\boldsymbol{y}_{i,\lambda_i} = \arg\min_{\boldsymbol{y}} \|\boldsymbol{H}_i\boldsymbol{y} - \|\boldsymbol{b}\|_2\boldsymbol{e}_1\|_2^2 + \lambda_i\|\boldsymbol{x}\|_2^2$ and take $\boldsymbol{x}_{i,\lambda} = \boldsymbol{V}_i\boldsymbol{y}_{i,\lambda_i}$
end for

The discrepancy based approach for automatically selecting a regularization parameter for each iteration of hybrid-GMRES can also be generalized to hybrid LSQR, for which the discrepancy function is now defined as

$$\psi_i(\lambda) = \|\boldsymbol{B}_i\boldsymbol{y}_{i,\lambda} - \|\boldsymbol{b}\|_2\boldsymbol{e}_1\|_2,$$

where $\boldsymbol{y}_{i,\lambda}$ is the solution to (2.14) with $i = k$ by using regularization parameter λ. In each iteration i, λ_i is updated using the same formula

$$\lambda_i = \left|\frac{\mu\epsilon - \psi_i(0)}{\psi_i(\lambda_{i-1}) - \psi_i(0)}\right| \lambda_{i-1}. \tag{2.17}$$

The hybrid LSQR method using the discrepancy principle for automatically setting λ and early stopping is shown in Algorithm 5.

Algorithm 5 Hybrid LSQR with discrepancy principle

Inputs: $\boldsymbol{A}$, $\boldsymbol{b}$
Take $\boldsymbol{u}_1 = \boldsymbol{b}/\|\boldsymbol{b}\|_2$
for $i = 1, 2, \ldots,$ until a stopping criterion is satisfied **do**
  Compute $\boldsymbol{w} = \boldsymbol{A}^T\boldsymbol{u}_i$ and $\boldsymbol{w} = \boldsymbol{w} - \beta_{i-1}\boldsymbol{v}_{i-1}$ if $i > 1$
  Set $\alpha_i = \|\boldsymbol{w}\|_2$ and $\boldsymbol{v}_i = \boldsymbol{w}/\alpha_i$
  Compute $\boldsymbol{w} = \boldsymbol{A}^T\boldsymbol{v}_i$, $\boldsymbol{w} = \boldsymbol{w} - \alpha_i\boldsymbol{u}_i$
  Set $\beta_i = \|\boldsymbol{w}\|_2$ and $\boldsymbol{u}_{i+1} = \boldsymbol{w}/\beta_i$
  Let $b_{ii} = \alpha_i$ and $b_{i+1,i} = \beta_i$ and if $\alpha_i = 0$ or $\beta_i = 0$, break
  Select λ_i using (2.17)
  Compute $\boldsymbol{y}_{i,\lambda_i} = \arg\min_{\boldsymbol{y}} \|\boldsymbol{B}_i\boldsymbol{y} - \|\boldsymbol{b}\|_2\boldsymbol{e}_1\|_2^2 + \lambda_i\|\boldsymbol{x}\|_2^2$ and take $\boldsymbol{x}_{i,\lambda} = \boldsymbol{V}_i\boldsymbol{y}_{i,\lambda_i}$
end for

2.4.3 Flexible Krylov Methods

Flexible Krylov methods are a class of linear solvers that can handle iteration-dependent preconditioners: they were originally introduced in [63] for FGMRES, where a preconditioner for GMRES was allowed to change from one iteration to the next, either because at each iteration the preconditioner is implicitly defined by applying an iterative linear solver, or because the preconditioner can be updated with newly-computed information (see [66] for an overview).

In the framework of regularizing linear solvers, flexible Krylov methods were proposed in [11, 19, 22], where the iteration-dependent "preconditioner" was associated to an iteratively reweighted norm approach to Tikhonov-like regularized problems involving penalization terms expressed in some p-norm, $0 < p \leqslant 1$. These "preconditioners" have the effect of enforcing specific regularity into the approximation subspace for the solution, rather than accelerating the convergence of the iterative solvers. Leveraging flexible Krylov subspaces in this setting comes with the upside of avoiding restarts of the iterative solver, which is the approach commonly used when

adopting an iteratively reweighted norm method.

Consider first the case of a square $\boldsymbol{A} \in \mathbb{R}^{N\times N}$. In general, starting with $\boldsymbol{x}_0 = \boldsymbol{0}$, at the kth iteration, FGMRES updates a partial flexible Arnoldi factorization and computes the kth approximate solution as follows:

$$\boldsymbol{A}\boldsymbol{Z}_k = \boldsymbol{V}_{k+1}\boldsymbol{H}_k\,, \quad \boldsymbol{x}_k = \boldsymbol{Z}_k\boldsymbol{y}_k\,, \text{ where } \boldsymbol{y}_k = \arg\min_{\boldsymbol{y}\in\mathbb{R}^k} \|\boldsymbol{H}_k\boldsymbol{y} - \|\boldsymbol{b}\|_2\boldsymbol{e}_1\|_2\,, \tag{2.18}$$

where $\boldsymbol{V}_{k+1} = [\boldsymbol{v}_1,\ \ldots\ , \boldsymbol{v}_{k+1}] \in \mathbb{R}^{N\times(k+1)}$ has orthonormal columns, $\boldsymbol{H}_k \in \mathbb{R}^{(k+1)\times k}$ is upper Hessenberg, and $\boldsymbol{Z}_k = [\boldsymbol{P}_1\boldsymbol{v}_1, \ldots, \boldsymbol{P}_k\boldsymbol{v}_k] \in \mathbb{R}^{N\times k}$ has columns that span the approximation subspace for the solution ($\boldsymbol{P}_i$ is an iteration-dependent preconditioner that is applied to $\boldsymbol{v}_i$). The flexible GMRES method (FGMRES) uses the flexible Arnoldi process [19] to generate iterates of the form $\boldsymbol{x}_k = Z_k\boldsymbol{y}_k$, where the vector $\boldsymbol{y}_k$ is computed as $\boldsymbol{y}_k = \arg\min_{\boldsymbol{y}} \left\| \boldsymbol{H}_k\boldsymbol{y} - \|\boldsymbol{b}\|_2\boldsymbol{e}_1 \right\|_2^2$.

The extension to more general matrices $\boldsymbol{A} \in \mathbb{R}^{M\times N}$, with M not necessarily equal to N, can be naturally devised considering the flexible Golub-Kahan (FGK) process [11]. Taking $\boldsymbol{x}_0 = \boldsymbol{0}$ as the initial guess, the kth FGK iteration updates partial factorizations of the form

$$\boldsymbol{A}\boldsymbol{Z}_k = \boldsymbol{U}_{k+1}\boldsymbol{M}_k \ \text{ and } \ \boldsymbol{A}^T\boldsymbol{U}_{k+1} = \boldsymbol{V}_{k+1}\boldsymbol{T}_{k+1}, \tag{2.19}$$

where the columns of $\boldsymbol{U}_{k+1} \in \mathbb{R}^{M\times(k+1)}$, $\boldsymbol{V}_{k+1} \in \mathbb{R}^{N\times(k+1)}$ are orthonormal, $\boldsymbol{M}_k \in \mathbb{R}^{(k+1)\times k}$ is upper Hessenberg, $\boldsymbol{T}_{k+1} \in \mathbb{R}^{(k+1)\times(k+1)}$ is upper triangular, and $\boldsymbol{Z}_k = [\boldsymbol{P}_1\boldsymbol{v}_1, \ldots, \boldsymbol{P}_k\boldsymbol{v}_k] \in \mathbb{R}^{N\times k}$ has columns that span the approximation subspace for the solution ($\boldsymbol{P}_i$ is an iteration-dependent preconditioner that is applied to $\boldsymbol{v}_i$). The

flexible LSQR method (FLSQR) uses the FGK process (2.19) to generate iterates of the form $\boldsymbol{x}_k = Z_k \boldsymbol{y}_k$, where the vector $\boldsymbol{y}_k$ is computed as $\boldsymbol{y}_k = \arg\min_{\boldsymbol{y}} \left\| \boldsymbol{M}_k \boldsymbol{y} - \|\boldsymbol{b}\|_2 \boldsymbol{e}_1 \right\|_2^2$.

Chapter 3

TSVD approximations

In this Chapter, we introduce methods based on Kronecker product summations for approximating the truncated SVD (TSVD) of $\boldsymbol{A}$. Such approximations can be thought of as a low-rank approximation to the matrix being decomposed, and can be used to approximate the solution to the linear system 1.1. The new approximation approach discussed in this chapter is based on the paper jointly published by Garvey, Nagy and the author of this thesis [16]; see also Garvey's Ph.D. dissertation [15]. As described in Chapter 2, the solution to (1.1) can be computed directly if the full SVD of $\boldsymbol{A}$ is available. However, computing the full SVD is very expensive both time-wise and storage-wise, and is in general infeasible for large matrices. For example, on a personal computer, MATLAB's `svd` function can compute the full SVD of matrices of sizes up to 4096 by 4096 without much difficulty, but as matrices grow larger, a typical personal computer will run out of memory. That is, it is generally infeasible for image restoration problems larger than 64 by 64 pixels. For larger matrices,

if only a few (e.g., 10) leading singular values and vectors are needed, MATLAB's `svds` function, which uses iterative Golub-Kahan Bidiagonalization [24, 47] for the estimation, can be used. However, `svds` is not attractive in our applications since we may need to compute a few thousand singular values and vectors. Therefore, other efficient approximation methods need to be exploited. The focus of this Chapter is computing TSVD approximations for structured square matrices $\boldsymbol{A} \in \mathbb{R}^{N \times N}$ that can be found in the image deblurring application.

3.1 The Kronecker Product

Let $\boldsymbol{G} \in \mathbb{R}^{n \times n}$ and $\boldsymbol{H} \in \mathbb{R}^{n \times n}$ be square matrices with entries g_{ij} and h_{ij}. Then the Kronecker product of $\boldsymbol{G}$ and $\boldsymbol{H}$ is defined as

$$\boldsymbol{G} \otimes \boldsymbol{H} = \begin{bmatrix} g_{11}\boldsymbol{H} & \cdots & g_{1n}\boldsymbol{H} \\ \vdots & & \vdots \\ g_{n1}\boldsymbol{H} & \cdots & g_{nn}\boldsymbol{H} \end{bmatrix} \in \mathbb{R}^{N \times N},$$

where $N = n^2$. Useful properties of the Kronecker product include:

- $(\boldsymbol{A} \otimes \boldsymbol{B})^T = \boldsymbol{A}^T \otimes \boldsymbol{B}^T$,
- $(\boldsymbol{A} \otimes \boldsymbol{B})(\boldsymbol{C} \otimes \boldsymbol{D}) = \boldsymbol{AC} \otimes \boldsymbol{BD}$,
- $(\boldsymbol{B}^T \otimes \boldsymbol{A})\mathrm{vec}(\boldsymbol{X}) = \mathrm{vec}(\boldsymbol{AXB})$,

given that the matrices are of appropriate sizes such that the matrix products $\boldsymbol{AC}$, $\boldsymbol{BD}$ and $\boldsymbol{AXB}$ can be formed.

Let the SVDs of $\boldsymbol{G}$ and $\boldsymbol{H}$ be $\boldsymbol{G} = \boldsymbol{U_G}\boldsymbol{\Sigma_G}\boldsymbol{V_G}^T$ and $\boldsymbol{H} = \boldsymbol{U_H}\boldsymbol{\Sigma_H}\boldsymbol{V_H}^T$ respectively. Suppose that $\boldsymbol{A}$ is the Kronecker product of $\boldsymbol{G}$ and $\boldsymbol{H}$, then we have that

$$\begin{aligned} \boldsymbol{A} = \boldsymbol{G} \otimes \boldsymbol{H} &= (\boldsymbol{U_G}\boldsymbol{\Sigma_G}\boldsymbol{V_G}^T) \otimes (\boldsymbol{U_H}\boldsymbol{\Sigma_H}\boldsymbol{V_H}^T) \\ &= (\boldsymbol{U}_G \otimes \boldsymbol{U_H})(\boldsymbol{\Sigma_G} \otimes \boldsymbol{\Sigma_H})(\boldsymbol{V_G} \otimes \boldsymbol{V_H})^T, \end{aligned}$$

where we have applied properties of the Kronecker product. Note that $(\boldsymbol{U_G} \otimes \boldsymbol{U_H})$ is an orthogonal matrix since

$$\begin{aligned} (\boldsymbol{U_G} \otimes \boldsymbol{U_H})^T(\boldsymbol{U_G} \otimes \boldsymbol{U_H}) &= (\boldsymbol{U_G}^T \otimes \boldsymbol{U_H}^T)(\boldsymbol{U_G} \otimes \boldsymbol{U_H}) \\ &= (\boldsymbol{U_G}^T\boldsymbol{U_G}) \otimes (\boldsymbol{U_H}^T\boldsymbol{U_H}) = \boldsymbol{I} \otimes \boldsymbol{I} = \boldsymbol{I}, \end{aligned}$$

and $(\boldsymbol{U}_G \otimes \boldsymbol{U}_H)(\boldsymbol{U}_G \otimes \boldsymbol{U}_H)^T = \boldsymbol{I}$ can be obtained in the same way. We can also show that $\boldsymbol{V_G} \otimes \boldsymbol{V_H}$ is an orthogonal matrix. Note that $\boldsymbol{\Sigma_G}$ and $\boldsymbol{\Sigma_H}$ are diagonal matrices with monotonically non-increasing diagonal entries. Therefore, $\boldsymbol{\Sigma_G} \otimes \boldsymbol{\Sigma_H}$ is also a diagonal matrix with nonnegative diagonal entries, but they are not necessarily monotonically decreasing. We can apply a permutation matrix $\boldsymbol{P} \in \mathbb{R}^{N \times N}$ to $(\boldsymbol{\Sigma_G} \otimes \boldsymbol{\Sigma_H})$ to reorder the diagonal so that $\boldsymbol{P}^T(\boldsymbol{\Sigma_G} \otimes \boldsymbol{\Sigma_H})\boldsymbol{P}$ has a decreasing diagonal. As a result, we obtain a proper SVD of $\boldsymbol{A}$:

$$\boldsymbol{A} = \underbrace{\big((\boldsymbol{U_G} \otimes \boldsymbol{U_H})\boldsymbol{P}\big)}_{\boldsymbol{U}} \underbrace{\big(\boldsymbol{P}^T(\boldsymbol{\Sigma_G} \otimes \boldsymbol{\Sigma_H})\boldsymbol{P}\big)}_{\boldsymbol{\Sigma}} \underbrace{\big((\boldsymbol{V_G} \otimes \boldsymbol{V_H})\boldsymbol{P}\big)^T}_{\boldsymbol{V}^T}. \tag{3.1}$$

This analysis suggests that if $\boldsymbol{A} \in \mathbb{R}^{N \times N}$ can be written as a Kronecker product of two small matrices $\boldsymbol{G}$, $\boldsymbol{H} \in \mathbb{R}^{n \times n}$, then we can simply compute the SVDs of

$\boldsymbol{G}$ and $\boldsymbol{H}$ to easily obtain the SVD of $\boldsymbol{A}$. In this way, complexity is reduced to $\mathcal{O}(n^3) = \mathcal{O}(N^{3/2})$ from $\mathcal{O}(N^3)$ (if we directly compute the SVD of $\boldsymbol{A}$).

3.2 Kronecker Product Summation

For a general matrix $\boldsymbol{A}$, it is hardly the case that it can be decomposed as $\boldsymbol{G} \otimes \boldsymbol{H}$. However, we may decompose $\boldsymbol{A}$ as a sum of Kronecker products $\boldsymbol{G}_i \otimes \boldsymbol{H}_i$ for $i = 1, 2, \cdots, R$, i.e.,

$$\boldsymbol{A} = \sum_{i=1}^{R} \boldsymbol{G}_i \otimes \boldsymbol{H}_i,$$

where the number of terms R is called the Kronecker rank of $\boldsymbol{A}$. The computational approach to obtain the Kronecker product summation is proposed by Van Loan and Pitsianis [68], and requires taking the SVD of a rearrangement $\widetilde{\boldsymbol{A}} \in \mathbb{R}^{N \times N}$ of matrix $\boldsymbol{A}$. By doing so, the first term of the sum $\boldsymbol{G}_1 \otimes \boldsymbol{H}_1$ is computed such that

$$\boldsymbol{G}_1 \otimes \boldsymbol{H}_1 = \arg\min_{\boldsymbol{G},\boldsymbol{H}} \|\boldsymbol{A} - \boldsymbol{G} \otimes \boldsymbol{H}\|_F^2.$$

So intuitively, the first term $\boldsymbol{G}_1 \otimes \boldsymbol{H}_1$ is the best Kronecker product approximation of $\boldsymbol{A}$. It is the most "significant" term in the summation and contains the most information of $\boldsymbol{A}$ compared to the remaining terms. Therefore, a natural SVD approximation of $\boldsymbol{A}$ is to take the SVD of $\boldsymbol{G}_1 \otimes \boldsymbol{H}_1$, i.e., if $\boldsymbol{G}_1 = \boldsymbol{U}_{\boldsymbol{G}_1} \boldsymbol{\Sigma}_{\boldsymbol{G}_1} \boldsymbol{V}_{\boldsymbol{G}_1}^T$ and

$\boldsymbol{H}_1 = \boldsymbol{U}_{\boldsymbol{H}_1}\boldsymbol{\Sigma}_{\boldsymbol{H}_1}\boldsymbol{V}_{\boldsymbol{H}_1}^T$, and

$$\begin{aligned}\boldsymbol{G}_1 \otimes \boldsymbol{H}_1 &= (\boldsymbol{U}_{\boldsymbol{G}_1}\boldsymbol{\Sigma}_{\boldsymbol{G}_1}\boldsymbol{V}_{\boldsymbol{G}_1}^T) \otimes (\boldsymbol{U}_{\boldsymbol{H}_1}\boldsymbol{\Sigma}_{\boldsymbol{H}_1}\boldsymbol{V}_{\boldsymbol{H}_1}^T) \\ &= (\boldsymbol{U}_{\boldsymbol{G}_1} \otimes \boldsymbol{U}_{\boldsymbol{H}_1})(\boldsymbol{\Sigma}_{\boldsymbol{G}_1} \otimes \boldsymbol{\Sigma}_{\boldsymbol{H}_1})(\boldsymbol{V}_{\boldsymbol{G}_1} \otimes \boldsymbol{V}_{\boldsymbol{H}_1})^T \\ &= \boldsymbol{U}_{\boldsymbol{A}_1}\boldsymbol{\Sigma}_{\boldsymbol{A}_1}\boldsymbol{V}_{\boldsymbol{A}_1}^T,\end{aligned}$$

so the SVD approximation to $\boldsymbol{A}$ is

$$\boldsymbol{A} \approx \boldsymbol{U}_{\boldsymbol{A}_1}\boldsymbol{\Sigma}_{\boldsymbol{A}_1}\boldsymbol{V}_{\boldsymbol{A}_1}^T. \tag{3.2}$$

Note that the singular values in $\boldsymbol{\Sigma}_{\boldsymbol{A}_1}$ are not yet ordered. This approximation uses only the first term in the Kronecker product summation and discards the information in the remaining terms, so naturally, the accuracy of $\boldsymbol{U}_{\boldsymbol{A}_1}\boldsymbol{\Sigma}_{\boldsymbol{A}_1}\boldsymbol{V}_{\boldsymbol{A}_1}^T$ depends greatly on how closely $\boldsymbol{G}_1 \otimes \boldsymbol{H}_1$ approximates $\boldsymbol{A}$, which depends greatly on the structure of $\boldsymbol{A}$.

To use more Kronecker product terms for the SVD approximation, Nagy and Kamm [41, 52] proposed the following

$$\boldsymbol{A} \approx \boldsymbol{U}_{\boldsymbol{A}_1}\grave{\boldsymbol{\Sigma}}\boldsymbol{V}_{\boldsymbol{A}_1}^T, \tag{3.3}$$

where $\boldsymbol{U}_{\boldsymbol{A}_1}$ and $\boldsymbol{V}_{\boldsymbol{A}_1}$ are the same as in (3.2), and $\grave{\boldsymbol{\Sigma}} = \operatorname{diag}(\boldsymbol{U}_{\boldsymbol{A}_1}^T\boldsymbol{A}\boldsymbol{V}_{\boldsymbol{A}_1})$. This approach is better than (3.2) in the sense that it uses more information than just $\boldsymbol{G}_1$ and $\boldsymbol{H}_1$. However, the disadvantage is also very clear – the diagonal entries of $\grave{\boldsymbol{\Sigma}}$ can be negative, which violates the definition of singular values. Therefore, although

both (3.2) and (3.3) are very computationally efficient, we still want to keep looking for better approximation approaches.

3.3 A New TSVD Approximation Method via Kronecker Product Summations

The disadvantages of existing approaches lead to the question of how to use more terms in the Kronecker product summations in order to obtain a more accurate SVD approximation, while strictly adhering to the requirements of the SVD. We now propose an alternative approximation method described in [15, 16]. Starting with the Kronecker sum decomposition $\boldsymbol{A} = \sum_{i=1}^{R} \boldsymbol{G}_i \otimes \boldsymbol{H}_i$, and the SVD of the first term $\boldsymbol{G}_1 \otimes \boldsymbol{H}_1 = \boldsymbol{U}_{\boldsymbol{A}_1} \boldsymbol{\Sigma}_{\boldsymbol{A}_1} \boldsymbol{V}_{\boldsymbol{A}_1}^T$, we have that

$$
\begin{aligned}
\boldsymbol{A} &= \sum_{i=1}^{R} \boldsymbol{G}_i \otimes \boldsymbol{H}_i \\
&= \boldsymbol{U}_{\boldsymbol{A}_1} \boldsymbol{\Sigma}_{\boldsymbol{A}_1} \boldsymbol{V}_{\boldsymbol{A}_1}^T + \sum_{i=2}^{R} \boldsymbol{G}_i \otimes \boldsymbol{H}_i \\
&= \boldsymbol{U}_{\boldsymbol{A}_1} \Bigg(\boldsymbol{\Sigma}_{\boldsymbol{A}_1} + \underbrace{\boldsymbol{U}_{\boldsymbol{A}_1}^T \Big(\sum_{i=2}^{R} \boldsymbol{G}_i \otimes \boldsymbol{H}_i \Big) \boldsymbol{V}_{\boldsymbol{A}_1}}_{\boldsymbol{W}} \Bigg) \boldsymbol{V}_{\boldsymbol{A}_1}^T \\
&= \boldsymbol{U}_{\boldsymbol{A}_1} \Big(\boldsymbol{\Sigma}_{\boldsymbol{A}_1} + \boldsymbol{W} \Big) \boldsymbol{V}_{\boldsymbol{A}_1}^T .
\end{aligned}
$$

In this new method we propose, we focus on computing the k largest singular values and corresponding left and right singular vectors (i.e., to obtain a TSVD

approximation). Hence, we need to apply a reordering matrix $\boldsymbol{P} \in \mathbb{R}^{N\times N}$ to sort the singular values of $\boldsymbol{\Sigma}_{\boldsymbol{A}_1}$ in descending order in the same way as for (3.1). We obtain

$$\begin{aligned} \boldsymbol{A} &= \boldsymbol{U}_{\boldsymbol{A}_1}\boldsymbol{P}\Big(\boldsymbol{P}^T(\boldsymbol{\Sigma}_{\boldsymbol{A}_1}+\boldsymbol{W})\boldsymbol{P}\Big)(\boldsymbol{V}_{\boldsymbol{A}_1}\boldsymbol{P})^T \\ &= \bar{\boldsymbol{U}}_{\boldsymbol{A}_1}\Big(\bar{\boldsymbol{\Sigma}}_{\boldsymbol{A}_1}+\overline{\boldsymbol{W}}\Big)\bar{\boldsymbol{V}}_{\boldsymbol{A}_1}^T, \end{aligned}$$

where $\bar{\boldsymbol{U}}_{\boldsymbol{A}_1} = \boldsymbol{U}_{\boldsymbol{A}_1}\boldsymbol{P}$, $\bar{\boldsymbol{V}}_{\boldsymbol{A}_1} = \boldsymbol{V}_{\boldsymbol{A}_1}\boldsymbol{P}$, $\bar{\boldsymbol{\Sigma}}_{\boldsymbol{A}_1} = \boldsymbol{P}^T\boldsymbol{\Sigma}_{\boldsymbol{A}_1}\boldsymbol{P}$ and $\overline{\boldsymbol{W}} = \boldsymbol{P}^T\boldsymbol{W}\boldsymbol{P}$. Since we are interested in the k largest singular values, we further partition

$$\bar{\boldsymbol{\Sigma}}_{\boldsymbol{A}_1} = \begin{bmatrix} \bar{\boldsymbol{\Sigma}}_{\boldsymbol{A}_1,1:k} & \mathbf{0} \\ \mathbf{0} & \bar{\boldsymbol{\Sigma}}_{\boldsymbol{A}_1,k+1:N} \end{bmatrix}, \tag{3.4}$$

where $\bar{\boldsymbol{\Sigma}}_{\boldsymbol{A}_1,1:k} \in \mathbb{R}^{k\times k}$ and $\bar{\boldsymbol{\Sigma}}_{\boldsymbol{A}_1,k+1:N} \in \mathbb{R}^{N-k\times N-k}$. We also partition $\overline{\boldsymbol{W}} = \begin{bmatrix} \overline{\boldsymbol{W}}_{11} & \overline{\boldsymbol{W}}_{12} \\ \overline{\boldsymbol{W}}_{21} & \overline{\boldsymbol{W}}_{22} \end{bmatrix}$, where the blocks are of the same dimension as the blocks of $\bar{\boldsymbol{\Sigma}}_{\boldsymbol{A}_1}$ in (3.4). Using this notation,

$$
\begin{aligned}
\boldsymbol{A} &= \bar{\boldsymbol{U}}_{\boldsymbol{A}_1}\left(\left[\begin{array}{cc} \bar{\boldsymbol{\Sigma}}_{\boldsymbol{A}_1,1:k} & \boldsymbol{0} \\ \boldsymbol{0} & \bar{\boldsymbol{\Sigma}}_{\boldsymbol{A}_1,k+1:N} \end{array}\right] + \left[\begin{array}{cc} \overline{\boldsymbol{W}}_{11} & \overline{\boldsymbol{W}}_{12} \\ \overline{\boldsymbol{W}}_{21} & \overline{\boldsymbol{W}}_{22} \end{array}\right]\right)\bar{\boldsymbol{V}}_{\boldsymbol{A}_1}^T \\
&= \bar{\boldsymbol{U}}_{\boldsymbol{A}_1}\left(\left[\begin{array}{cc} \bar{\boldsymbol{\Sigma}}_{\boldsymbol{A}_1,1:k} + \overline{\boldsymbol{W}}_{11} & \boldsymbol{0} \\ \boldsymbol{0} & \bar{\boldsymbol{\Sigma}}_{\boldsymbol{A}_1,k+1:N} \end{array}\right] + \left[\begin{array}{cc} \boldsymbol{0} & \overline{\boldsymbol{W}}_{12} \\ \overline{\boldsymbol{W}}_{21} & \overline{\boldsymbol{W}}_{22} \end{array}\right]\right)\bar{\boldsymbol{V}}_{\boldsymbol{A}_1}^T \\
&= \bar{\boldsymbol{U}}_{\boldsymbol{A}_1}\left(\left[\begin{array}{cc} \boldsymbol{U}_T\boldsymbol{\Sigma}_T\boldsymbol{V}_T^T & \boldsymbol{0} \\ \boldsymbol{0} & \bar{\boldsymbol{\Sigma}}_{\boldsymbol{A}_1,k+1:N} \end{array}\right] + \left[\begin{array}{cc} \boldsymbol{0} & \overline{\boldsymbol{W}}_{12} \\ \overline{\boldsymbol{W}}_{21} & \overline{\boldsymbol{W}}_{22} \end{array}\right]\right)\bar{\boldsymbol{V}}_{\boldsymbol{A}_1}^T \\
&= \bar{\boldsymbol{U}}_{\boldsymbol{A}_1}\left[\begin{array}{cc} \boldsymbol{U}_T & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{I} \end{array}\right]\left(\left[\begin{array}{cc} \boldsymbol{\Sigma}_T & \boldsymbol{0} \\ \boldsymbol{0} & \bar{\boldsymbol{\Sigma}}_{\boldsymbol{A}_1,k+1:N} \end{array}\right] + \left[\begin{array}{cc} \boldsymbol{0} & \boldsymbol{U}_T^T\overline{\boldsymbol{W}}_{12} \\ \overline{\boldsymbol{W}}_{21}\boldsymbol{V}_T & \overline{\boldsymbol{W}}_{22} \end{array}\right]\right)\left[\begin{array}{cc} \boldsymbol{V}_T^T & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{I} \end{array}\right]\bar{\boldsymbol{V}}_{\boldsymbol{A}_1}^T \\
&= \breve{\boldsymbol{U}}\left(\left[\begin{array}{cc} \boldsymbol{\Sigma}_T & \boldsymbol{0} \\ \boldsymbol{0} & \bar{\boldsymbol{\Sigma}}_{\boldsymbol{A}_1,k+1:N} \end{array}\right] + \left[\begin{array}{cc} \boldsymbol{0} & \widehat{\boldsymbol{W}}_{12} \\ \widehat{\boldsymbol{W}}_{21} & \overline{\boldsymbol{W}}_{22} \end{array}\right]\right)\breve{\boldsymbol{V}}^T, \qquad (3.5)
\end{aligned}
$$

where we have defined $\widehat{\boldsymbol{W}}_{12} = \boldsymbol{U}_T^T\overline{\boldsymbol{W}}_{12}$, $\widehat{\boldsymbol{W}}_{21} = \overline{\boldsymbol{W}}_{21}\boldsymbol{V}_T$, $\breve{\boldsymbol{U}} = \bar{\boldsymbol{U}}_{\boldsymbol{A}_1}\left[\begin{array}{cc} \boldsymbol{U}_T & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{I} \end{array}\right]$ and $\breve{\boldsymbol{V}} = \bar{\boldsymbol{V}}_{\boldsymbol{A}_1}\left[\begin{array}{cc} \boldsymbol{V}_T & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{I} \end{array}\right]$. Moreover, we have used the full SVD of $\boldsymbol{T} = \bar{\boldsymbol{\Sigma}}_{\boldsymbol{A}_1,1:k} + \overline{\boldsymbol{W}}_{11} = \boldsymbol{U}_T\boldsymbol{\Sigma}_T\boldsymbol{V}_T^T$ in the third step of the derivation. Although computing the full SVD of $\boldsymbol{A}$ is impractical when N is large, it is feasible to compute the full SVD of $\boldsymbol{T}$ if k is moderate. From (3.5), we are able to form a TSVD approximation, $\breve{\boldsymbol{A}}_{\mathrm{TSVD}}$ of $\boldsymbol{A}$

$$
\breve{\boldsymbol{A}}_{\mathrm{TSVD}} \approx \breve{\boldsymbol{U}}_{1:k}\breve{\boldsymbol{\Sigma}}_{1:k}\breve{\boldsymbol{V}}_{1:k}^T, \qquad (3.6)
$$

where $\breve{\boldsymbol{U}}_{1:k} \in \mathbb{R}^{N\times k}$ and $\breve{\boldsymbol{V}}_{1:k} \in \mathbb{R}^{N\times k}$ are the submatrices of $\breve{\boldsymbol{U}}$ and $\breve{\boldsymbol{V}}$ containing their first k columns respectively, and $\breve{\boldsymbol{\Sigma}}_{1:k} = \boldsymbol{\Sigma}_T$ for consistency of notation.

3.3.1 Approximation Quality

In this subsection, we summarize theoretical results that provide bounds for different types of approximation errors, which is the author's main contribution to this joint work. Before we get started, a few more notations need to be introduced. Since we are bounding errors, true quantities need to be referred to. Let the true SVD of $\boldsymbol{A}$ be

$$\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T = \begin{bmatrix} \boldsymbol{U}_{1:k} & \boldsymbol{U}_{k+1:N} \end{bmatrix} \begin{bmatrix} \boldsymbol{\Sigma}_{1:k} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{\Sigma}_{k+1:N} \end{bmatrix} \begin{bmatrix} \boldsymbol{V}_{1:k} & \boldsymbol{V}_{k+1:N} \end{bmatrix}^T,$$

and the true rank-k TSVD approximation of $\boldsymbol{A}$ (i.e., the best rank-k approximation of $\boldsymbol{A}$) is therefore defined as

$$\boldsymbol{A}_{\text{TSVD}} = \boldsymbol{U}_{1:k}\boldsymbol{\Sigma}_{1:k}\boldsymbol{V}_{1:k}^T. \tag{3.7}$$

Sequentially, the true TSVD filtered solution $\boldsymbol{x}_{\text{TSVD}}$ and the approximate TSVD solution $\breve{\boldsymbol{x}}_{\text{TSVD}}$ are

$$\boldsymbol{x}_{\text{TSVD}} = \boldsymbol{V}_{1:k}\boldsymbol{\Sigma}_{1:k}^{-1}\boldsymbol{U}_{1:k}^T\boldsymbol{b} \quad \text{and} \quad \breve{\boldsymbol{x}}_{\text{TSVD}} = \breve{\boldsymbol{V}}_{1:k}\breve{\boldsymbol{\Sigma}}_{1:k}^{-1}\breve{\boldsymbol{U}}_{1:k}^T\boldsymbol{b} \tag{3.8}$$

We start the error analysis with the bound for the singular vector subspaces,

similar to the results presented by Fierro and Bunch [13] for the case of URV and ULV factorizations. The "signal" subspace of $\boldsymbol{A}$ is spanned by $\boldsymbol{U}_{1:k}$ and the "noise" subspace is spanned by $\boldsymbol{V}_{k+1:N}$. The distance between the approximated and true signal and noise subspaces are measured by

$$\|\boldsymbol{U}_{1:k}^T \breve{\boldsymbol{U}}_{k+1:N}\|_2 \quad \text{and} \quad \|\boldsymbol{V}_{1:k}^T \breve{\boldsymbol{V}}_{k+1:N}\|_2,$$

which represent one measure for the quality of the TSVD approximation.

Theorem 3.3.1. *Consider the factorizations of $\boldsymbol{A}$ given in equations (3.7) and (3.6), where σ_i denotes a true singular value and $\breve{\sigma}_i$ denotes an approximate singular value. Then*

$$\|\boldsymbol{U}_{1:k}^T \breve{\boldsymbol{U}}_{k+1:N}\| \leqslant \frac{\sigma_k \|\widehat{\boldsymbol{W}}_{21}\| + \|\widehat{\boldsymbol{W}}_{12}\| \|\boldsymbol{\Sigma}_{K_1,k+1:N} + \overline{\boldsymbol{W}}_{22}\|}{\sigma_k^2 - \|\boldsymbol{\Sigma}_{K_1,k+1:N} + \overline{\boldsymbol{W}}_{22}\|^2}$$

and

$$\|\boldsymbol{V}_{1:k}^T \breve{\boldsymbol{V}}_{k+1:N}\| \leqslant \frac{\sigma_k \|\widehat{\boldsymbol{W}}_{12}\| + \|\widehat{\boldsymbol{W}}_{21}\| \, \|\boldsymbol{\Sigma}_{K_1,k+1:N} + \overline{\boldsymbol{W}}_{22}\|}{\sigma_k^2 - \|\boldsymbol{\Sigma}_{K_1,k+1:N} + \overline{\boldsymbol{W}}_{22}\|^2}$$

We refer reders to [16] for the full proof of Theorem 3.3.1. This Theorem is the foundation for proving error bounds for the pseudoinverse $\boldsymbol{A}_{\text{TSVD}}$ and approximate solution $\boldsymbol{x}_{\text{TSVD}}$ shown below.

Theorem 3.3.2. *Consider the true TSVD operator $\boldsymbol{A}_{TSVD}$ defined in (3.7) and its approximation $\breve{\boldsymbol{A}}_{TSVD}$ given by (3.6). Let $\breve{\sigma}_i$ be the i^{th} diagonal entry of $\breve{\boldsymbol{\Sigma}}_{1:k}$, and define $\varphi = \left(1 + \sqrt{5}\right)/2$. Then*

$$\frac{\left\| \boldsymbol{A}_{TSVD}^{\dagger} - \breve{\boldsymbol{A}}_{TSVD}^{\dagger} \right\|}{\left\| \boldsymbol{A}_{TSVD}^{\dagger} \right\|} \leqslant \frac{\varphi}{\breve{\sigma}_k} \left(\sigma_1 \|\boldsymbol{V}_{1:k}^T \breve{\boldsymbol{V}}_{k+1:N}\| + \|\widehat{\boldsymbol{W}}_{21}\| \right).$$

Proof. The proof starts with a perturbation result for pseudoinverses presented in [70]: If $\boldsymbol{C}$ is an acute perturbation [49] of $\boldsymbol{D}$, with $\boldsymbol{D} = \boldsymbol{C} + \delta\boldsymbol{C}$, then

$$\|\boldsymbol{C}^\dagger - \boldsymbol{D}^\dagger\| \leqslant \varphi\|\boldsymbol{C}^\dagger\|\|\boldsymbol{D}^\dagger\|\|\delta\boldsymbol{C}\|. \tag{3.9}$$

Since

$$\begin{aligned}\|\boldsymbol{A}_{\mathrm{TSVD}} - \breve{\boldsymbol{A}}_{\mathrm{TSVD}}\| &= \|\boldsymbol{A}\boldsymbol{V}_{1:k}\boldsymbol{V}_{1:k}^T - (\boldsymbol{A}\breve{\boldsymbol{V}}_{1:k}\breve{\boldsymbol{V}}_{1:k}^T - \breve{\boldsymbol{U}}_{k+1:N}\widehat{\boldsymbol{W}}_{21}\breve{\boldsymbol{V}}_{1:k}^T)\| \\ &\leqslant \|\boldsymbol{A}\|\|\boldsymbol{V}_{1:k}\boldsymbol{V}_{1:k}^T - \breve{\boldsymbol{V}}_{1:k}\breve{\boldsymbol{V}}_{1:k}^T\| + \|\widehat{\boldsymbol{W}}_{21}\|,\end{aligned}$$

and it is proved in [24] that $\|\boldsymbol{V}_{1:k}\boldsymbol{V}_{1:k}^T - \breve{\boldsymbol{V}}_{1:k}\breve{\boldsymbol{V}}_{1:k}^T\| = \|\boldsymbol{V}_{1:k}^T\breve{\boldsymbol{V}}_{k+1:N}\|$, we get

$$\|\boldsymbol{A}_{\mathrm{TSVD}} - \breve{\boldsymbol{A}}_{\mathrm{TSVD}}\| \leqslant \|\boldsymbol{A}\|\|\boldsymbol{V}_{1:k}^T\breve{\boldsymbol{V}}_{k+1:N}\| + \|\widehat{\boldsymbol{W}}_{21}\|.$$

$\breve{\boldsymbol{A}}_{\mathrm{TSVD}}$ is an acute perturbation of $\boldsymbol{A}_{\mathrm{TSVD}}$, so by (3.9),

$$\left\|\boldsymbol{A}_{\mathrm{TSVD}}^\dagger - \breve{\boldsymbol{A}}_{\mathrm{TSVD}}^\dagger\right\| \leqslant \varphi\|\boldsymbol{A}_{\mathrm{TSVD}}^\dagger\|\|\breve{\boldsymbol{A}}_{\mathrm{TSVD}}^\dagger\|\left(\|\boldsymbol{A}\|\|\boldsymbol{V}_{1:k}^T\breve{\boldsymbol{V}}_{k+1:N}\| + \|\widehat{\boldsymbol{W}}_{21}\|\right). \tag{3.10}$$

Dividing both sides of the inequality by $\|\boldsymbol{A}_{\mathrm{TSVD}}^\dagger\|$, we obtain

$$\begin{aligned}\frac{\left\|\boldsymbol{A}_{\mathrm{TSVD}}^\dagger - \breve{\boldsymbol{A}}_{\mathrm{TSVD}}^\dagger\right\|}{\left\|\boldsymbol{A}_{\mathrm{TSVD}}^\dagger\right\|} &\leqslant \varphi\|\breve{\boldsymbol{A}}_{\mathrm{TSVD}}^\dagger\|\left(\|\boldsymbol{A}\|\|\boldsymbol{V}_{1:k}^T\breve{\boldsymbol{V}}_{k+1:N}\| + \|\widehat{\boldsymbol{W}}_{21}\|\right) \\ &= \frac{\varphi}{\breve{\sigma}_k}\left(\sigma_1\|\boldsymbol{V}_{1:k}^T\breve{\boldsymbol{V}}_{k+1:N}\| + \|\widehat{\boldsymbol{W}}_{21}\|\right).\end{aligned}$$

□

Theorem 3.3.3. *Consider the true and approximate TSVD filtered solutions* $\boldsymbol{x}_{TSVD}$ *and* $\widehat{\boldsymbol{x}}_{TSVD}$ *as in (3.8). Further, define the residual* $\boldsymbol{r} = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_{TSVD}$*. Then*

$$\frac{\|\boldsymbol{x}_{TSVD} - \breve{\boldsymbol{x}}_{TSVD}\|}{\|\boldsymbol{x}_{TSVD}\|} \leqslant \frac{\varphi\sigma_1}{\sigma_k\widehat{\sigma}_k\sqrt{1-\frac{\|\boldsymbol{r}\|^2}{\|\boldsymbol{b}\|^2}}}\left(\sigma_1\|\boldsymbol{V}_{1:k}^T\breve{\boldsymbol{V}}_{k+1:N}\| + \|\widehat{\boldsymbol{W}}_{21}\|\right). \tag{3.11}$$

Proof. From inequality (3.10), we obtain

$$\begin{aligned}\|\boldsymbol{x}_{\mathrm{TSVD}} - \breve{\boldsymbol{x}}_{\mathrm{TSVD}}\| &\leqslant \left\|\boldsymbol{A}_{\mathrm{TSVD}}^{\dagger} - \breve{\boldsymbol{A}}_{\mathrm{TSVD}}^{\dagger}\right\|\|\boldsymbol{b}\| \\ &\leqslant \varphi\|\boldsymbol{A}_{\mathrm{TSVD}}^{\dagger}\|\|\breve{\boldsymbol{A}}_{\mathrm{TSVD}}^{\dagger}\|\left(\|\boldsymbol{A}\|\|\boldsymbol{A}_{1:k}^T\breve{\boldsymbol{V}}_{k+1:N}\| + \|\widehat{\boldsymbol{W}}_{21}\|\right)\|\boldsymbol{d}\|.\end{aligned}$$

By the triangle inequality and submultiplicativity of induced norms,

$$\|\boldsymbol{x}_{\mathrm{TSVD}}\| \geqslant \frac{\|\boldsymbol{b}\|}{\|\boldsymbol{A}\|}\sqrt{1-\frac{\|\boldsymbol{r}\|^2}{\|\boldsymbol{b}\|^2}},$$

it follows that

$$\begin{aligned}\frac{\|\boldsymbol{x}_{\mathrm{TSVD}} - \breve{\boldsymbol{x}}_{\mathrm{TSVD}}\|}{\|\boldsymbol{x}_{\mathrm{TSVD}}\|} &\leqslant \frac{\varphi}{\sqrt{1-\frac{\|\boldsymbol{r}\|^2}{\|\boldsymbol{b}\|^2}}}\|\boldsymbol{A}_{\mathrm{TSVD}}^{\dagger}\|\|\breve{\boldsymbol{A}}_{\mathrm{TSVD}}^{\dagger}\|\|\boldsymbol{A}\|\left(\|\boldsymbol{A}\|\|\boldsymbol{V}_{1:k}^T\breve{\boldsymbol{V}}_{k+1:N}\| + \|\widehat{\boldsymbol{W}}_{21}\|\right) \\ &= \frac{\varphi\sigma_1}{\sigma_k\widehat{\sigma}_k\sqrt{1-\frac{\|\boldsymbol{r}\|^2}{\|\boldsymbol{b}\|^2}}}\left(\sigma_1\|\boldsymbol{V}_{1:k}^T\breve{\boldsymbol{V}}_{k+1:N}\| + \|\widehat{\boldsymbol{W}}_{21}\|\right).\end{aligned}$$

□

These two theorems tell us that the quality of the approximate pseudoinverse and the approximate TSVD solution is dependent on three conditions:

- The distance between the approximate and true noise subspaces;

- The ratio of the largest singular value σ_1 of $\boldsymbol{A}$ to the kth approximate singular value $\breve{\sigma}_k$;

- The size of $\|\widehat{\boldsymbol{W}}_{21}\|$, which depends on whether $\boldsymbol{G}_1 \otimes \boldsymbol{H}_1$ is a good approximation to $\boldsymbol{A}$ and whether k is a good truncation rank.

This newly proposed TSVD approximation method is supported by both theoretical and numerical results, in which the new method computes more accurate solutions than (3.2) and (3.3). Our new method can also approximate singular values and vectors more efficiently than other well-known schemes such as randomized methods [27] and Golub-Kahan bidiagonalization [24, 47]. In this thesis, we omit details on method implementation and numerical experiments, and refer readers to [15] and [16] for a thorough introduction of this new approach.

Chapter 4

Low Rank Regularization: Derivation of New Methods

In the Chapter 3, we discussed Kronecker product summations based methods for approximating the TSVD of $\boldsymbol{A}$. TSVD is a singular value filtering type of regularization scheme that eliminates the noise contribution caused by small singular values of $\boldsymbol{A}$, but there is not an explicit regularization term $\mathcal{R}(\boldsymbol{x})$ associated with it. In this Chapter, we move on to a different type of regularization – nuclear norm regularization (NNR), for which $\mathcal{R}(\boldsymbol{x}) = \|\text{vec}^{-1}(\boldsymbol{x})\|_* = \|\boldsymbol{X}\|_*$, and describe how to solve the nuclear norm regularized problem using Krylov subspace methods and novel preconditioners. We introduce methods that target on both square matrices $\boldsymbol{A} \in \mathbb{R}^{N\times N}$ and rectangular matrices $\boldsymbol{A} \in \mathbb{R}^{M\times N}$. These new methods are proposed in [18], which is the author's joint work with Gazzola and Nagy. We apply nuclear norm regularization to imaging inverse problems that can be formulated as linear

systems

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}, \tag{4.1}$$

where $\boldsymbol{A}$, $\boldsymbol{x}$ and $\boldsymbol{b}$ are defined in the exact same way as in (1.1), in order to implicitly enforce a low rank constraint by penalizing the sum of singular values of the solution. Indeed, two-dimensional images are often assumed to have low-rank or to be well-approximated by low-rank two-dimensional arrays (see [57] and the references therein).

Numerical linear algebra solvers for the estimation of low-rank solutions to linear systems have been developed in the literature, mainly targeting well-posed linear discrete problems, such as those arising when considering the numerical solution of stochastic PDEs (see [48] and the references therein). In particular, the authors of [48] devise a restarted GMRES-like method (RS-LR-GMRES) that involves low-rank projections of the basis vectors of the solution subspace, as well as a low-rank projection of the current solution at the end of each cycle. Since, in general, the basic operations involved in standard GMRES (such as matrix-vector products and vector sums) increase the ranks of the computed quantities, low-rank projections are needed to assure that the computed solution is low-rank.

In the framework of compressive sensing, the authors of [6] consider a modified version of the conjugate gradient method that incorporates appropriate rank-truncation operations. All the methods mentioned so far employ, often in a heuristic way, Krylov subspace methods together with rank-reduction operations (e.g., projections onto a chosen set of low-rank matrices). Since many Krylov subspace methods are iterative regularization methods for (4.1), this brings us to the question of how

incorporating rank-reduction operations would affect the solution of the discrete inverse problem (4.1), with a particular focus on imaging applications.

4.1 Nuclear Norm Regularization

Low-rank matrix estimation can be naturally formulated as a nonconvex optimization problem having either:

(i) a least-squares data fitting term as objective function and a rank constraint; or

(ii) the rank of $\boldsymbol{X} = \text{vec}^{-1}(\boldsymbol{x})$ as objective function and a constraint on the least-squares data fitting term.

The last instance is commonly referred to as *affine rank minimization problem*, and both formulations are in general NP-hard [57]. In this Chapter, we consider the unconstrained and convex optimization problem

$$\min_{\boldsymbol{x}} \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2^2 + \lambda \|\text{vec}^{-1}(\boldsymbol{x})\|_* \,, \tag{4.2}$$

where $\lambda > 0$ is a regularization parameter and $\|\cdot\|_*$ denotes the nuclear norm of $\text{vec}^{-1}(\boldsymbol{x}) = \boldsymbol{X}$, defined as the sum of the singular values of $\boldsymbol{X}$. If the singular value decomposition (SVD) of $\boldsymbol{X}$ is given by $\boldsymbol{X} = \boldsymbol{U}_{\boldsymbol{X}} \boldsymbol{\Sigma}_{\boldsymbol{X}} \boldsymbol{V}_{\boldsymbol{X}}^T$, where $\boldsymbol{U}_{\boldsymbol{X}}, \boldsymbol{V}_{\boldsymbol{X}} \in \mathbb{R}^{n \times n}$ are orthogonal matrices, and $\boldsymbol{\Sigma}_{\boldsymbol{X}} \in \mathbb{R}^{n \times n}$ is the diagonal matrix whose diagonal entries

are $\sigma_1(\boldsymbol{X}) \geqslant \cdots \geqslant \sigma_n(\boldsymbol{X}) \geqslant 0$, then

$$\|\boldsymbol{X}\|_* = \sum_{i=1}^{n} \sigma_i(\boldsymbol{X})\,.$$

Problem (4.2) is refered to as a *nuclear norm regularized (NNR) problem*. In particular, the nuclear norm is a convex function that has been proven to be the best convex lower approximation of the rank function over the set of matrices $\boldsymbol{X}$ such that $\|\boldsymbol{X}\|_2 \leqslant 1$ (see [57] and the references therein). The nuclear norm has been used in many applications, such as low-rank matrix completion and compressed sensing; see, e.g., [7, 23, 43, 50, 57], where the constrained formulation of problem (4.2) has also been considered (note that, for a proper choice of $\lambda > 0$, constrained and unconstrained formulations are equivalent; see, e.g., [60]). In the framework of compressive sensing, under the assumption that the matrix $\boldsymbol{A}$ satisfies a certain null-space property, recovery guarantees for the affine rank minimization problem are proven in [14, 51]. We also consider the following formulation

$$\min_{\boldsymbol{x}} \|\boldsymbol{A}\boldsymbol{x}-\boldsymbol{b}\|_2^2+\lambda\|\mathrm{vec}^{-1}(\boldsymbol{x})\|_{*,p}\,, \quad \text{where} \quad \|\boldsymbol{X}\|_{*,p} = \sum_{i=1}^{n}(\sigma_i(\boldsymbol{X}))^p, \quad 0 < p \leqslant 1\,. \tag{4.3}$$

Problem (4.3) is refereed to as *NNRp problem*, and it generalizes problem (4.2) (which is obtained taking $p = 1$ in (4.3)). The constrained version of (4.3) is already considered in [51], where the authors empirically show an improved recovery performance of the constrained formulation of problem (4.3) with $p < 1$ with respect to $p = 1$. Note, however, that the choice $p < 1$ in (4.3) results in a nonconvex minimization problem.

4.2 Derivation of New Methods

In the following, we will quite often interchange $\boldsymbol{x}$ and $\boldsymbol{X}$ and, with a slight abuse of notations, we will denote the action of a linear operator on $\boldsymbol{x}$ or $\boldsymbol{X}$ by $\mathcal{A}(\boldsymbol{X}) = \mathcal{A}\boldsymbol{X} = A\boldsymbol{x}$, and the action of the adjoint operator by $\mathcal{A}^*(\boldsymbol{Y}) = \mathcal{A}^*\boldsymbol{Y} = A^T\mathrm{vec}(\boldsymbol{Y})$. Define the smooth Schatten-p function as

$$\mathcal{S}_p^\gamma(\boldsymbol{X}) = \mathrm{Tr}((\boldsymbol{X}^T\boldsymbol{X} + \gamma\boldsymbol{I})^{p/2}), \quad \text{with } \gamma > 0\,.$$

Note that $\mathcal{S}_p^\gamma(\boldsymbol{X})$ is differentiable for $p > 0$ and convex for $p \geqslant 1$. In particular, for $p = 1$ and $\gamma = 0$ (i.e., no smoothing),

$$\mathcal{S}_1^0(\boldsymbol{X}) = \mathrm{Tr}((\boldsymbol{X}^T\boldsymbol{X})^{1/2}) = \|\boldsymbol{X}\|_*\,.$$

We start by considering the following smooth approximation to (4.3):

$$\min_{\boldsymbol{X}\in\mathbb{R}^{n\times n}} \|\mathcal{A}(\boldsymbol{X}) - \boldsymbol{B}\|_F^2 + \lambda\mathcal{S}_p^\gamma(\boldsymbol{X})\,. \tag{4.4}$$

The following derivations are valid for $p > 0$ (and we keep them generic, being aware that $p = 1$ approximates (4.2)). The optimality conditions associated to (4.4) read

$$\begin{aligned} 0 &= \nabla_{\boldsymbol{X}}\left(\|\mathcal{A}(\boldsymbol{X}) - \boldsymbol{B}\|_F^2 + \lambda\mathcal{S}_p^\gamma(\boldsymbol{X})\right) \\ &= 2\mathcal{A}^*(\mathcal{A}(\boldsymbol{X}) - \boldsymbol{B}) + \lambda\, p(\boldsymbol{X}\boldsymbol{X}^T + \gamma\boldsymbol{I})^{p/2-1}\boldsymbol{X}\,, \end{aligned} \tag{4.5}$$

where we have used that

$$\nabla_{\boldsymbol{X}} \operatorname{Tr}((\boldsymbol{X}^T\boldsymbol{X}+\gamma\boldsymbol{I})^{p/2}) = p\boldsymbol{X}(\boldsymbol{X}^T\boldsymbol{X}+\gamma\boldsymbol{I})^{p/2-1} = p(\boldsymbol{X}\boldsymbol{X}^T+\gamma\boldsymbol{I})^{p/2-1}\boldsymbol{X}\,.$$

Equivalently, the nonlinear system of equations (4.5) with respect to $\boldsymbol{X}$ can be expressed as

$$\begin{aligned}\boldsymbol{X} &= \left(\mathcal{A}^*\mathcal{A}+\widehat{\lambda}(\boldsymbol{X}\boldsymbol{X}^T+\gamma\boldsymbol{I})^{p/2-1}\right)^{-1}\mathcal{A}^*\boldsymbol{B}\\ &= \left(\mathcal{A}^*\mathcal{A}+\widehat{\lambda}((\boldsymbol{X}\boldsymbol{X}^T+\gamma\boldsymbol{I})^{p/4-1/2})^T(\boldsymbol{X}\boldsymbol{X}^T+\gamma\boldsymbol{I})^{p/4-1/2}\right)^{-1}\mathcal{A}^*\boldsymbol{B}\,,\end{aligned}$$

with $\widehat{\lambda} = \lambda\, p/2$, which is naturally associated to the following fixed-point iteration scheme

$$\boldsymbol{X}_{k+1} = \left(\mathcal{A}^*\mathcal{A}+\widehat{\lambda}((\boldsymbol{X}_k\boldsymbol{X}_k^T+\gamma\boldsymbol{I})^{p/4-1/2})^T(\boldsymbol{X}_k\boldsymbol{X}_k^T+\gamma\boldsymbol{I})^{p/4-1/2}\right)^{-1}\mathcal{A}^*\boldsymbol{B}\,, \qquad (4.6)$$

which leads to the solution of (4.4). Equivalently,

$$\boldsymbol{X}_{k+1} = \arg\min_{\boldsymbol{X}} \left\| \begin{bmatrix} \mathcal{A} \\ \sqrt{\widehat{\lambda}}(\boldsymbol{X}_k\boldsymbol{X}_k^T+\gamma\boldsymbol{I})^{p/4-1/2} \end{bmatrix} \boldsymbol{X} - \begin{bmatrix} \boldsymbol{B} \\ \boldsymbol{0} \end{bmatrix} \right\|_F^2,$$

i.e., (4.6) are the normal equations associated to the penalized least squares problem written above or, equivalently,

$$\boldsymbol{X}_{k+1} = \arg\min_{\boldsymbol{X}} \|\mathcal{A}\boldsymbol{X}-\boldsymbol{B}\|_F^2 + \widehat{\lambda}\left\|(\boldsymbol{X}_k\boldsymbol{X}_k^T+\gamma\boldsymbol{I})^{p/4-1/2}\boldsymbol{X}\right\|_F^2\,. \qquad (4.7)$$

We now reformulate problem (4.7) in vectorial form.

Let $\boldsymbol{U}_{\boldsymbol{X}_k}\boldsymbol{\Sigma}_{\boldsymbol{X}_k}\boldsymbol{V}_{\boldsymbol{X}_k}^T = \boldsymbol{X}_k$ be the SVD of $\boldsymbol{X}_k$; thanks to the invariance of the Frobenius norm under orthogonal transformations, the regularization term in the above problem can be rewritten as

$$\begin{aligned}\left\|(\boldsymbol{X}_k\boldsymbol{X}_k^T+\gamma\boldsymbol{I})^{p/4-1/2}\boldsymbol{X}\right\|_F^2 &= \left\|\boldsymbol{U}_{\boldsymbol{X}_k}(\boldsymbol{\Sigma}_{\boldsymbol{X}_k}^2+\gamma\boldsymbol{I})^{p/4-1/2}\boldsymbol{U}_{\boldsymbol{X}_k}^T\boldsymbol{X}\right\|_F^2 \\ &= \left\|(\boldsymbol{\Sigma}_{\boldsymbol{X}_k}^2+\gamma\boldsymbol{I})^{p/4-1/2}\boldsymbol{U}_{\boldsymbol{X}_k}^T\boldsymbol{X}\boldsymbol{V}_{\boldsymbol{X}_k}\right\|_F^2.\end{aligned}$$

Using well-known Kronecker product properties

$$\begin{aligned}\left\|(\boldsymbol{\Sigma}_{\boldsymbol{X}_k}^2+\gamma\boldsymbol{I})^{p/4-1/2}\boldsymbol{U}_{\boldsymbol{X}_k}^T\boldsymbol{X}\boldsymbol{V}_{\boldsymbol{X}_k}\right\|_F^2 &= \left\|\mathrm{vec}\left((\boldsymbol{\Sigma}_{\boldsymbol{X}_k}^2+\gamma\boldsymbol{I})^{p/4-1/2}\boldsymbol{U}_{\boldsymbol{X}_k}^T\boldsymbol{X}\boldsymbol{V}_{\boldsymbol{X}_k}\right)\right\|_2^2 \\ &= \left\|\left(\boldsymbol{V}_{\boldsymbol{X}_k}^T\otimes\left((\boldsymbol{\Sigma}_{\boldsymbol{X}_k}^2+\gamma\boldsymbol{I})^{p/4-1/2}\boldsymbol{U}_{\boldsymbol{X}_k}^T\right)\right)\boldsymbol{x}\right\|_2^2 \\ &= \left\|\left(\boldsymbol{I}\otimes(\boldsymbol{\Sigma}_{\boldsymbol{X}_k}^2+\gamma\boldsymbol{I})^{p/4-1/2}\right)\left(\boldsymbol{V}_{\boldsymbol{X}_k}^T\otimes\boldsymbol{U}_{\boldsymbol{X}_k}^T\right)\boldsymbol{x}\right\|_2^2.\end{aligned}$$

Problem (4.7) is therefore equivalent to

$$\boldsymbol{x}_{k+1} = \arg\min_{\boldsymbol{x}}\|\boldsymbol{A}\boldsymbol{x}-\boldsymbol{b}\|_2^2 + \widehat{\lambda}\|\underbrace{\left(\boldsymbol{I}\otimes(\boldsymbol{\Sigma}_{\boldsymbol{X}_k}^2+\gamma\boldsymbol{I})^{p/4-1/2}\right)}_{=:(\boldsymbol{W}_p^\gamma)_k}\overbrace{\left(\boldsymbol{V}_{\boldsymbol{X}_k}^T\otimes\boldsymbol{U}_{\boldsymbol{X}_k}^T\right)}^{=:\boldsymbol{S}_k}\boldsymbol{x}\|_2^2. \quad (4.8)$$

In the above formulation, $(\boldsymbol{W}_p^\gamma)_k$ is a diagonal weighting matrix and $\boldsymbol{S}_k$ is an orthogonal matrix; both $(\boldsymbol{W}_p^\gamma)_k$ and $\boldsymbol{S}_k$ depend on the current approximation $\boldsymbol{x}_k$ of the solution $\boldsymbol{x}$. Intuitively, the matrix $\boldsymbol{S}_k$ maps $\boldsymbol{x}$ into the "singular value domain" of $\boldsymbol{X}_k$ (and acts as an iteration-dependent sparsity transform), and the matrix $(\boldsymbol{W}_p^\gamma)_k$ assigns suitable weights that allow to approximate a p-norm of the singular values.

Therefore, the penalization term in (4.8) can be interpreted as a reweighted vectorial 2-norm, with respect to a transformation of the solution $\boldsymbol{x}$. For this reason, the proposed approach is called "IRN-NNRp" and is summarized in Algorithm 6.

Algorithm 6 IRN-NNRp

Inputs: $\boldsymbol{A}$, $\boldsymbol{b}$, $(\boldsymbol{W}_p^\gamma)_0 = \boldsymbol{I}$, $\boldsymbol{S}_0 = \boldsymbol{I}$
for $k = 0, 1, \ldots$ until a stopping criterion is satisfied **do**
    Solve problem (4.8)
    "Decrease" γ
    Update $(\boldsymbol{W}_p^\gamma)_{k+1}$ and $\boldsymbol{S}_{k+1}$
end for

4.3 Solution of problem (4.8) via Krylov methods

In this section, we derive new strategies for the efficient solution of the sequence of sub-problems (4.8) appearing in Algorithm 6. First, we rewrite problem (4.8) using an appropriate change of variable as

$$\widehat{\boldsymbol{x}}_{k+1} = \arg\min_{\widehat{\boldsymbol{x}}} \|\boldsymbol{A}\boldsymbol{S}_k^T(\boldsymbol{W}_p^\gamma)_k^{-1}\widehat{\boldsymbol{x}} - \boldsymbol{b}\|_2^2 + \widehat{\lambda}\|\widehat{\boldsymbol{x}}\|_2^2, \text{ with } \widehat{\boldsymbol{x}} = (\boldsymbol{W}_p^\gamma)_k\boldsymbol{S}_k\boldsymbol{x}\,. \tag{4.9}$$

Note that

$$\boldsymbol{S}_k^T = \boldsymbol{S}_k^{-1} = \boldsymbol{V}_{\boldsymbol{X}_k} \otimes \boldsymbol{U}_{\boldsymbol{X}_k} \quad \text{and} \quad (\boldsymbol{W}_p^\gamma)_k^{-1} = \boldsymbol{I} \otimes (\boldsymbol{\Sigma}_{\boldsymbol{X}_k}^2 + \gamma\boldsymbol{I})^{1/2-p/4}\,, \tag{4.10}$$

so that the above transformations (inversion of an orthogonal and a diagonal matrix) are numerically affordable by exploiting properties of Kronecker products. The Tikhonov-regularized problem (4.9) in standard form is equivalent to the Tikhonov-

regularized problem (4.8) in general form. Many Krylov subspace methods based on the Golub-Kahan Bidiagonalization (GKB) or Arnoldi algorithms can be employed to approximate the solution of (4.9), e.g., LSQR and GMRES, as explained in Section 2.4. Moreover, if the regularization parameter $\widehat{\lambda}$ is not known a priori, many efficient strategies to set its value adaptively within the sequence of projected problems can be used (i.e., in the framework of hybrid methods; see Section 2.4.2 and [21, 44]). The matrices $\boldsymbol{S}_k$ and $(\boldsymbol{W}_p^{\gamma})_k^{-1}$ can be formally thought of as preconditioners for the original problem (4.1), whose purpose is to enforce additional regularization into the solution subspace, rather than speeding-up the convergence of linear solvers applied to (4.1).

4.3.1 Methods Based on the GKB Algorithm

The mth step of the GKB algorithm applied to the matrix $\boldsymbol{A}\boldsymbol{S}_k^T(\boldsymbol{W}_p^{\gamma})_k^{-1}$ with starting vector $\boldsymbol{b}$ (i.e., taking $\boldsymbol{x}_0 = \boldsymbol{0}$) can be expressed by the following partial matrix factorizations

$$(\boldsymbol{A}\boldsymbol{S}_k^T(\boldsymbol{W}_p^{\gamma})_k^{-1})\boldsymbol{V}_m = \boldsymbol{U}_{m+1}\boldsymbol{B}_m \quad \text{and} \quad ((\boldsymbol{W}_p^{\gamma})_k^{-1}\boldsymbol{S}_k\boldsymbol{A}^T)\boldsymbol{U}_{m+1} = \boldsymbol{V}_{m+1}\boldsymbol{B}_{m+1}^T, \quad (4.11)$$

where $\boldsymbol{U}_j \in \mathbb{R}^{M\times j}$ and $\boldsymbol{V}_j \in \mathbb{R}^{N\times j}$ (with $j = m, m+1$ and $\boldsymbol{U}_j\boldsymbol{e}_1 = \boldsymbol{b}/\|\boldsymbol{b}\|_2$) have orthonormal columns, and $\boldsymbol{B}_{m+1} \in \mathbb{R}^{(m+1)\times(m+1)}$ is lower bidiagonal (with $\boldsymbol{B}_m$ obtained by removing the last column of $\boldsymbol{B}_{m+1}$). The orthonormal columns of $\boldsymbol{V}_m$ are

such that

$$\mathcal{R}(\boldsymbol{V}_m) = \mathcal{K}_m\left(((\boldsymbol{W}_p^{\gamma})_k^{-1}\boldsymbol{S}_k\boldsymbol{A}^T)(\boldsymbol{A}\boldsymbol{S}_k^T(\boldsymbol{W}_p^{\gamma})_k^{-1}), ((\boldsymbol{W}_p^{\gamma})_k^{-1}\boldsymbol{S}_k\boldsymbol{A}^T)\boldsymbol{b}\right) .$$

We find an approximate solution of (4.9) by imposing $\widehat{\boldsymbol{x}} \in \mathcal{R}(\boldsymbol{V}_m)$, i.e., $\widehat{\boldsymbol{x}}_m = \boldsymbol{V}_m\boldsymbol{y}_m$, where, by exploiting the first decomposition in (4.11) and the properties of the matrices appearing therein, $\boldsymbol{y}_m \in \mathbb{R}^m$ is such that

$$\boldsymbol{y}_m = \arg\min_{\boldsymbol{y}\in\mathbb{R}^m} \|\boldsymbol{B}_m\boldsymbol{y} - \|\boldsymbol{b}\|_2\boldsymbol{e}_1\|_2^2 + \widehat{\lambda}_m\|\boldsymbol{y}\|_2^2 . \tag{4.12}$$

We used the notation $\widehat{\lambda}_m$ for the regularization parameter to highlight that its value can be adaptively set within the iterations. The approximate solution to problem (4.8) is such that

$$\boldsymbol{x} = \boldsymbol{S}_k^T(\boldsymbol{W}_p^{\gamma})_k^{-1}\widehat{\boldsymbol{x}} \in \mathcal{K}_m\left((\boldsymbol{S}_k^T(\boldsymbol{W}_p^{\gamma})_k^{-2}\boldsymbol{S}_k)\boldsymbol{A}^T\boldsymbol{A}, (\boldsymbol{S}_k^T(\boldsymbol{W}_p^{\gamma})_k^{-2}\boldsymbol{S}_k)\boldsymbol{A}^T\boldsymbol{b}\right) . \tag{4.13}$$

Looking at the above approximation subspace for the solution $\boldsymbol{x}$, it is evident that the "preconditioner" acts by first mapping into the "singular value domain" (by applying $\boldsymbol{S}_k$), enforcing sparsity in the singular values (by reweighting with $(\boldsymbol{W}_p^{\gamma})_k^{-2}$), and eventually transforming back into the "solution domain" (by applying $\boldsymbol{S}_k^T$).

4.3.2 Methods Based on the Arnoldi Algorithm

If $\boldsymbol{A}$ is square, the mth step of the Arnoldi algorithm applied to the matrix $\boldsymbol{A}\boldsymbol{S}_k^T(\boldsymbol{W}_p^{\gamma})_k^{-1}$ with starting vector $\boldsymbol{b}$ (i.e., taking $\boldsymbol{x}_0 = \boldsymbol{0}$) can be expressed by the following partial

matrix factorization

$$(\boldsymbol{A}\boldsymbol{S}_k^T(\boldsymbol{W}_p^\gamma)_k^{-1})\boldsymbol{V}_m = \boldsymbol{V}_{m+1}\boldsymbol{H}_m\,, \tag{4.14}$$

where $\boldsymbol{V}_j \in \mathbb{R}^{N\times j}$ (with $j = m, m+1$ and $\boldsymbol{V}_j\boldsymbol{e}_1 = \boldsymbol{b}/\|\boldsymbol{b}\|_2$) has orthonormal columns such that

$$\mathcal{R}(\boldsymbol{V}_m) = \mathcal{K}_m\left(\boldsymbol{A}\boldsymbol{S}_k^T(\boldsymbol{W}_p^\gamma)_k^{-1}, \boldsymbol{b}\right)\,,$$

and $\boldsymbol{H}_m \in \mathbb{R}^{(m+1)\times m}$ is upper Hessenberg. Similarly to the GKB case, we find an approximate solution of (4.9) by imposing $\widehat{\boldsymbol{x}} \in \mathcal{R}(\boldsymbol{V}_m)$ and by solving a projected Tikhonov problem of order m. The approximate solution to problem (4.8) is such that

$$\boldsymbol{x} = \boldsymbol{S}_k^T(\boldsymbol{W}_p^\gamma)_k^{-1}\widehat{\boldsymbol{x}} \in \boldsymbol{S}_k^T(\boldsymbol{W}_p^\gamma)_k^{-1}\mathcal{K}_m\left(\boldsymbol{A}\boldsymbol{S}_k^T(\boldsymbol{W}_p^\gamma)_k^{-1}, \boldsymbol{b}\right)\,,$$

where

$$\begin{aligned}
& \boldsymbol{S}_k^T(\boldsymbol{W}_p^\gamma)_k^{-1}\mathcal{K}_m\left(\boldsymbol{A}\boldsymbol{S}_k^T(\boldsymbol{W}_p^\gamma)_k^{-1}, \boldsymbol{b}\right) \\
= \;& \operatorname{span}\{\boldsymbol{S}_k^T(\boldsymbol{W}_p^\gamma)_k^{-1}\boldsymbol{b}, \ldots, \left(\boldsymbol{S}_k^T(\boldsymbol{W}_p^\gamma)_k^{-1}\boldsymbol{A}\right)^{m-1}\boldsymbol{S}_k^T(\boldsymbol{W}_p^\gamma)_k^{-1}\boldsymbol{b}\} \\
= \;& \mathcal{K}_m\left(\boldsymbol{S}_k^T(\boldsymbol{W}_p^\gamma)_k^{-1}\boldsymbol{A}, \boldsymbol{S}_k^T(\boldsymbol{W}_p^\gamma)_k^{-1}\boldsymbol{b}\right).
\end{aligned}$$

Contrarily to the GKB case, we immediately notice that, in this context, $\boldsymbol{x}$ does not belong to a meaningful approximation subspace. Indeed, just by looking at the first vector: $\boldsymbol{b}$ is in the image space and $(\boldsymbol{W}_p^\gamma)_k^{-1}$ is supposed to act on the singular value space of $\boldsymbol{X}_k$, so $(\boldsymbol{W}_p^\gamma)_k^{-1}\boldsymbol{b}$ is hard to interpret; furthermore, $\boldsymbol{S}_k^T$ is supposed to link the singular value space of $\boldsymbol{X}_k$ to the image space, so $\boldsymbol{S}_k^T(\boldsymbol{W}_p^\gamma)_k^{-1}\boldsymbol{b}$ is also hard for us to interpret. Although the generated solution subspace is not meaningful for our

applications, it may still have the potential to be a good subspace in other contexts. Similarly to what is proposed in [4, 11], where the Arnoldi algorithm is applied to a regularized problem that enforces sparsity in the wavelet domain, we propose to fix this issue by incorporating $\boldsymbol{S}_k$ also as an orthogonal left "preconditioner" for the original system (4.1) so that, by exploiting the invariance of the vector 2-norm under orthogonal transformations, problem (4.9) can be equivalently reformulated as

$$\widehat{\boldsymbol{x}}_{k+1} = \arg\min_{\widehat{\boldsymbol{x}}} \|\boldsymbol{S}_k(\boldsymbol{A}\boldsymbol{S}_k^T(\boldsymbol{W}_p^\gamma)_k^{-1}\widehat{\boldsymbol{x}} - \boldsymbol{b})\|_2^2 + \widehat{\lambda}\|\widehat{\boldsymbol{x}}\|_2^2, \text{ with } \widehat{\boldsymbol{x}} = (\boldsymbol{W}_p^\gamma)_k\boldsymbol{S}_k\boldsymbol{x}\,. \quad (4.15)$$

The (right and left) preconditioned Arnoldi algorithm applied to problem (4.15) can now be expressed by the following partial matrix factorization

$$(\boldsymbol{S}_k\boldsymbol{A}\boldsymbol{S}_k^T(\boldsymbol{W}_p^\gamma)_k^{-1})\boldsymbol{V}_m = \boldsymbol{V}_{m+1}\boldsymbol{H}_m\,. \quad (4.16)$$

We find an approximate solution of (4.15) by imposing

$$\widehat{\boldsymbol{x}} \in \mathcal{R}(\boldsymbol{V}_m) = \mathcal{K}_m(\boldsymbol{S}_k\boldsymbol{A}\boldsymbol{S}_k^T\boldsymbol{W}^{-1}, \boldsymbol{S}_k\boldsymbol{b}),$$

i.e., $\widehat{\boldsymbol{x}}_m = \boldsymbol{V}_m\boldsymbol{y}_m$, where, by exploiting (4.16) and the properties of the matrices appearing therein, $\boldsymbol{y}_m \in \mathbb{R}^m$ is such that

$$\boldsymbol{y}_m = \arg\min_{\boldsymbol{y}\in\mathbb{R}^m} \|\boldsymbol{H}_m\boldsymbol{y} - \|\boldsymbol{b}\|_2\boldsymbol{e}_1\|_2^2 + \widehat{\lambda}_m\|\boldsymbol{y}\|_2^2\,. \quad (4.17)$$

Hence

$$\begin{aligned} \boldsymbol{x} &\in \boldsymbol{S}_k^T(\boldsymbol{W}_p^\gamma)_k^{-1}\mathcal{K}_m(\boldsymbol{S}_k\boldsymbol{A}\boldsymbol{S}_k^T(\boldsymbol{W}_p^\gamma)_k^{-1}, \boldsymbol{S}_k\boldsymbol{b}) \\ &= \mathcal{K}_m(\boldsymbol{S}_k^T(\boldsymbol{W}_p^\gamma)_k^{-1}\boldsymbol{S}_k\boldsymbol{A}, \boldsymbol{S}_k^T(\boldsymbol{W}_p^\gamma)_k^{-1}\boldsymbol{S}_k\boldsymbol{b}) \ , \end{aligned} \tag{4.18}$$

which is suitable for approximating the solution. The new methods based on the GKB algorithm (for generic matrices) and Arnoldi algorithm (only if $\boldsymbol{A} \in \mathbb{R}^{N\times N}$) are called "IRN-LSQR-NNRp" and "IRN-GMRES-NNRp", respectively, and are summarized in Algorithm 7.

Algorithm 7 IRN-LSQR-NNRp and IRN-GMRES-NNRp

1: Inputs: $\boldsymbol{A}$, $\boldsymbol{b}$, $(\boldsymbol{W}_p^\gamma)_0 = \boldsymbol{I}$, $\boldsymbol{S}_0 = \boldsymbol{I}$
2: **for** $k = 0, 1, \ldots$ until a stopping criterion is satisfied **do**
3: **for** $m = 1, 2, \ldots$ until a stopping criterion is satisfied **do**
4: Update the factorizations (4.11) and (4.16), respectively
5: Solve the projected problem (4.12) and (4.17), respectively, tuning $\widehat{\lambda}_m$ if necessary
6: **end for**
7: "Decrease" γ
8: Update the new $(\boldsymbol{W}_p^\gamma)_{k+1}$ and $\boldsymbol{S}_{k+1}$
9: **end for**

4.4 Solution through flexible Krylov subspaces

Problem (4.3) reformulated as (4.9) allows us to naturally apply the flexible Golub-Kahan (FGK) and flexible Arnoldi algorithms. Indeed, instead of updating the "preconditioners" $\boldsymbol{S}_k$ and $(\boldsymbol{W}_p^\gamma)_k$ at the kth outer iteration of the nested iteration schemes of Algorithm 7, we propose to consider new "preconditioners" as soon as

a new approximation of the solution is available, i.e., at each iteration of a Krylov subspace solver. Therefore, at the $(i+1)$th iteration of the new solvers, the "preconditioners" $(\boldsymbol{W}_p^{\gamma})_i$ and $\boldsymbol{S}_i$ are computed as in (4.10), but using the SVD of the ith approximate solution

$$\boldsymbol{X}_i = \mathrm{vec}^{-1}(\boldsymbol{x}_i) = \boldsymbol{U}_{\boldsymbol{X}_i}\boldsymbol{\Sigma}_{\boldsymbol{X}_i}\boldsymbol{V}_{\boldsymbol{X}_i}^T, \quad \text{for } i = 1, \ldots, k-1\,,$$

with $(\boldsymbol{W}_p^{\gamma})_0 = \boldsymbol{I}$ and $\boldsymbol{S}_0 = \boldsymbol{I}$. In order to incorporate iteration-dependent preconditioning, the flexible versions of the Golub-Kahan and Arnoldi factorizations have to be used.

Namely, at the ith iteration, the new instance of the FGK algorithm updates partial factorizations of the form (2.19), i.e., $\boldsymbol{A}\boldsymbol{Z}_i = \boldsymbol{U}_{i+1}\boldsymbol{M}_i$ and $\boldsymbol{A}^T\boldsymbol{U}_{i+1} = \boldsymbol{V}_{i+1}\boldsymbol{T}_{i+1}$, where

$$\boldsymbol{Z}_i = [\boldsymbol{S}_0^T(\boldsymbol{W}_p^{\gamma})_0^{-2}\boldsymbol{S}_0\boldsymbol{v}_1, \ldots, \boldsymbol{S}_{i-1}^T(\boldsymbol{W}_p^{\gamma})_{i-1}^{-2}\boldsymbol{S}_{i-1}\boldsymbol{v}_i]\,, \quad \boldsymbol{v}_1 = \boldsymbol{A}^T\boldsymbol{b}/\|\boldsymbol{A}^T\boldsymbol{b}\|_2\,.$$

Taking $\boldsymbol{x}_0 = \boldsymbol{0}$, the ith approximate solution is such that $\boldsymbol{x}_i = \boldsymbol{Z}_i\boldsymbol{y}_i$, where

$$\boldsymbol{y}_i = \arg\min_{\boldsymbol{y}\in\mathbb{R}^i} \|\boldsymbol{M}_i\boldsymbol{y} - \|\boldsymbol{b}\|_2\boldsymbol{e}_1\|_2^2 + \widehat{\lambda}_i\|\boldsymbol{y}\|_2^2\,. \tag{4.19}$$

Note that the subspace for the solution $\mathcal{R}(\boldsymbol{Z}_i)$ can be regarded as a generalization of the subspace (4.13) computed when considering preconditioned GKB within the IRN-LSQR-NNRp method. The new method is called "FLSQR-NNRp", and is summarized in Algorithm 8.

For $\boldsymbol{A} \in \mathbb{R}^{N\times N}$ and $\boldsymbol{x}_0 = \boldsymbol{0}$, at the ith iteration, the new instance of the flexible Arnoldi algorithm updates a partial factorization of the form (2.18), with $k = i$, and generates

$$\boldsymbol{Z}_i = [\boldsymbol{S}_0^T(\boldsymbol{W}_p^\gamma)_0^{-1}\boldsymbol{S}_0\boldsymbol{v}_1, \ldots, \boldsymbol{S}_{i-1}^T(\boldsymbol{W}_p^\gamma)_{i-1}^{-1}\boldsymbol{S}_{i-1}\boldsymbol{v}_i]\,, \quad \boldsymbol{v}_1 = \boldsymbol{b}/\|\boldsymbol{b}\|_2\,,$$

where both right and left preconditioners are used analogously to IRN-GMRES-NNRp. The ith approximate solution is such that $\boldsymbol{x}_i = \boldsymbol{Z}_i\boldsymbol{y}_i$, where

$$\boldsymbol{y}_i = \arg\min_{\boldsymbol{y}\in\mathbb{R}^i} \|\boldsymbol{H}_i\boldsymbol{y} - \|\boldsymbol{b}\|_2\boldsymbol{e}_1\|_2^2 + \widehat{\lambda}_i\|\boldsymbol{y}\|_2^2\,. \tag{4.20}$$

Note that the subspace for the solution $\mathcal{R}(\boldsymbol{Z}_i)$ can be regarded as a generalization of the subspace (4.18) computed when considering the preconditioned Arnoldi algorithm within the IRN-GMRES-NNRp method. The new method is called "FGMRES-NNRp", and is summarized in Algorithm 8.

Algorithm 8 FLSQR-NNRp and FGMRES-NNRp

1: Inputs: $\boldsymbol{A}$, $\boldsymbol{b}$, $(\boldsymbol{W}_p^\gamma)_0 = \boldsymbol{I}$, $\boldsymbol{S}_0 = \boldsymbol{I}$
2: **for** $i = 1, 2, \ldots$ until a stopping criterion is satisfied **do**
3: Update a factorization of the form (2.19) and (2.18), respectively, to expand the space $\mathcal{R}(\boldsymbol{Z}_i)$
4: Solve the projected problem (4.19) and (4.20), respectively, tuning $\widehat{\lambda}_i$ if necessary
5: "Decrease" γ
6: Update the new $(\boldsymbol{W}_p^\gamma)_i$ and $\boldsymbol{S}_i$, using the SVD $\boldsymbol{X}_i = \text{vec}^{-1}(\boldsymbol{x}_i) = \boldsymbol{U}_{\boldsymbol{X}_i}\boldsymbol{\Sigma}_{\boldsymbol{X}_i}\boldsymbol{V}_{\boldsymbol{X}_i}^T$.
7: **end for**

Note that, although the approach of Algorithm 8 is quite heuristic, it avoids nested iteration cycles and computes only one approximation subspace for the so-

lution of (4.3), where low-rank penalization is adaptively incorporated. Because of this, in many situations, Algorithm 8 computes solutions of quality comparable to the ones computed by Algorithm 7, with a significant reduction in the number of iterations. We should also mention that, in the framework of affine rank minimization problems, [51] outlines an algorithm that avoids inner projected gradient iterations for the solution of each quadratic subproblem in the sequence generated within the IRN strategy.

Finally, we underline that, within the framework of flexible Krylov subspaces, the approximation subspaces $\mathcal{R}(\boldsymbol{Z}_i)$ for the ith approximate solution can be further modified, with some insight into the desired properties of the solution. Indeed, since the ith basis vector for the solution is of the form

$$\boldsymbol{z}_i = \boldsymbol{S}_{i-1}^T (\boldsymbol{W}_p^\gamma)_{i-1}^{-2} \boldsymbol{S}_{i-1} \boldsymbol{v}_i$$

for FLSQR-NNRp, and

$$\boldsymbol{z}_i = \boldsymbol{S}_{i-1}^T (\boldsymbol{W}_p^\gamma)_{i-1}^{-1} \boldsymbol{S}_{i-1} \boldsymbol{v}_i$$

for FGMRES-NNRp, one can consider alternative "preconditioners" $\boldsymbol{S}_{i-1}$ and $(\boldsymbol{W}_p^\gamma)_{i-1}$ that are still effective in delivering low-rank solutions. For instance, focusing on FGMRES, and given $\boldsymbol{v}_i = \boldsymbol{V}_i \boldsymbol{e}_i$, where $\boldsymbol{V}_i$ is the matrix appearing on the right-hand side of the factorization (2.19), and given the SVD of $\mathrm{vec}^{-1}(\boldsymbol{v}_i) = \boldsymbol{U}_{\boldsymbol{V}_i} \boldsymbol{\Sigma}_{\boldsymbol{V}_i} \boldsymbol{V}_{\boldsymbol{V}_i}^T$, one can take

$$\boldsymbol{S}_{i-1} = \boldsymbol{V}_{\boldsymbol{V}_i}^T \otimes \boldsymbol{U}_{\boldsymbol{V}_i}^T \quad \text{and} \quad (\boldsymbol{W}_p^\gamma)_{i-1}^{-1} = \boldsymbol{I} \otimes (\boldsymbol{\Sigma}_{\boldsymbol{V}_i})^{1-p/2}\,, \tag{4.21}$$

and as a result,

$$\begin{aligned}
\boldsymbol{S}_{i-1}\boldsymbol{v}_i &= \mathrm{vec}(\boldsymbol{U}_{\boldsymbol{V}_i}^T \mathrm{vec}^{-1}(\boldsymbol{v}_i)\boldsymbol{V}_{\boldsymbol{V}_i}) = \mathrm{vec}(\boldsymbol{\Sigma}_{V_i}), \\
(\boldsymbol{W}_p^{\gamma})_{i-1}^{-1}\boldsymbol{S}_{i-1}\boldsymbol{v}_i &= \mathrm{vec}((\boldsymbol{\Sigma}_{\boldsymbol{V}_i})^{1-p/2}\boldsymbol{\Sigma}_{\boldsymbol{V}_i}) = \mathrm{vec}((\boldsymbol{\Sigma}_{\boldsymbol{V}_i})^{2-p/2}), \\
\boldsymbol{S}_{i-1}^T(\boldsymbol{W}_p^{\gamma})_{i-1}^{-1}\boldsymbol{S}_{i-1}\boldsymbol{v}_i &= \mathrm{vec}(\boldsymbol{U}_{\boldsymbol{V}_i}((\boldsymbol{\Sigma}_{\boldsymbol{V}_i})^{2-p/2})\boldsymbol{V}_{\boldsymbol{V}_i}^T) = \boldsymbol{z}_i,
\end{aligned}$$

so that the singular values of $\mathrm{vec}^{-1}(\boldsymbol{v}_i)$ are rescaled: taking $0 < p \leqslant 1$, the power of $\boldsymbol{\Sigma}_{\boldsymbol{V}_i}$, $2 - p/2$, is always larger than 1, which means that large singular values get magnified and small singular values become even smaller. In this way, the gaps between singular values are emphasized and to some extent contribute to the low rank properties of the basis vectors. Similar derivations hold for FLSQR. Hence, methods analogous to LR-FLSQR and LR-FGMRES are obtained, and are called FGMRES-NNRp(v) and FLSQR-NNRp(v), respectively.

4.5 Implementation details

All the methods considered in Section 4.3 and 4.4 are iterative, and therefore at least one suitable stopping criterion should be set for the iterations. When considering hybrid formulations (like the ones in Algorithms 7 and 8), one could simultaneously set a good value for the regularization parameter $\widehat{\lambda}_j$ at the jth iteration, as well as properly stop the iterations. Strategies for achieving this are already available in the literature (see [17, 21]).

Recall from Section 2.4.2 that, assuming that a good estimate for the norm of the noise $\boldsymbol{\eta}$ affecting the right-hand-side of (1.1) is available, i.e., $\varepsilon \simeq \|\boldsymbol{\eta}\|_2$, one can

consider the discrepancy principle and stop the iterative scheme at the first iteration j such that

$$\|\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_j\|_2 \leqslant \mu\varepsilon\,, \quad \text{where } \mu \gtrapprox 1 \text{ is a safety threshold.} \tag{4.22}$$

When running hybrid methods (see Algorithms 7 and 8), we employ the so-called "secant method", which updates the regularization parameter for the projected problem in such a way that stopping by the discrepancy principle is ensured. We highlight again that the quantities needed to implement the "secant method" (namely, the norm of the residual and the discrepancy associated to (4.9) at each iteration) can be conveniently monitored using projected quantities: this is obvious for IRN-LSQR-NNRp and FLSQR-NNRp, as only right-"preconditioning" is employed; it is less obvious for IRN-GMRES-NNRp and FGMRES-NNRp, but since the left-"preconditioner" is orthogonal, one can still write

$$\|\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_j\|_2 = \|\boldsymbol{S}_k\boldsymbol{b} - \boldsymbol{S}_k\boldsymbol{A}\boldsymbol{S}_k^T(\boldsymbol{W}_p^\gamma)_k^{-1}\widehat{\boldsymbol{x}}_j\|_2 = \|\|\boldsymbol{b}\|_2\boldsymbol{e}_1 - \boldsymbol{H}_j\boldsymbol{y}_j\|_2\,.$$

Note that all the methods in Algorithm 7 and 8 can also run with $\widehat{\lambda} = 0$, and still achieve low-rank approximate solutions: this is because the approximation subspace for the solution incorporates regularizing "preconditioning" (see [28, 32] for details on this approach in the case of smoothing "preconditioning" with finite-difference approximations of derivatives operators). Finally, when dealing with the inner-outer iteration scheme of Algorithm 7, in addition to a parameter choice strategy and stopping criterion for the hybrid projected problems (4.12) and (4.17), one should

also consider a stopping criterion for the outer iterations. We propose to do this by monitoring the norm of the difference of the singular values (normalized by the largest singular value so that $\sigma_1(\boldsymbol{\Sigma}_{\boldsymbol{X}_{k+1}}) = \sigma_1(\boldsymbol{\Sigma}_{\boldsymbol{X}_k}) = 1$) of two approximations of the solution of (4.3) obtained at two consecutive outer iterations of Algorithm 7, i.e., we stop as soon as

$$\|\mathrm{diag}(\boldsymbol{\Sigma}_{\boldsymbol{X}_{k+1}}) - \mathrm{diag}(\boldsymbol{\Sigma}_{\boldsymbol{X}_k})\|_2 < \tau_\sigma, \quad k = 1, 2, \ldots, \tag{4.23}$$

where $\mathrm{vec}^{-1}(\boldsymbol{x}_i) = \boldsymbol{X}_i = \boldsymbol{U}_{\boldsymbol{X}_i}\boldsymbol{\Sigma}_{\boldsymbol{X}_i}\boldsymbol{V}_{\boldsymbol{X}_i}^T$ $(i = k, k+1)$, and $\tau_\sigma > 0$ is a user-specified threshold. If no significant changes happen in the rank and singular values of two consecutive approximations of the solution, then (4.23) is satisfied.

We conclude this section with a few remarks about the computational cost of the proposed methods. Note that, if $\boldsymbol{A} \in \mathbb{R}^{N\times N}$, IRN-GMRES-NNR$p$ is intrinsically cheaper than IRN-LSQR-NNRp (since, at each iteration, the former requires only one matrix-vector product with $\boldsymbol{A}$, while the latter requires one matrix-vector product with $\boldsymbol{A}$ and one with $\boldsymbol{A}^T$). However, methods based on the Arnoldi algorithm are typically less successful than methods based on the GKB algorithm for regularization; see [40].

Other key operations for implementing our proposed methods are the computation of the SVDs of relevant quantities, and/or the application of the "preconditioners" in (4.21). Namely, each iteration of FLSQR-NNRp, and FGMRES-NNRp requires the computation of the SVD of an $n \times n$ matrix, which amounts to $\mathcal{O}(n^3) = \mathcal{O}(N^{3/2})$ floating point operations. When considering IRN-LSQR-NNRp and IRN-GMRES-NNRp, only the SVD of the approximate solution should be computed once

at each outer iteration. However, each inner iteration of IRN-LSQR-NNRp and IRN-GMRES-NNRp, as well as each iteration of FLSQR-NNRp and FGMRES-NNRp, requires the computation of matrix-vector products of the form $\boldsymbol{S}_k^T(\boldsymbol{W}_p^\gamma)_k^{-1}\boldsymbol{v}_i$: this can be achieved within a two-step process, where first the rescaling $\widetilde{\boldsymbol{v}}_i = (\boldsymbol{W}_p^\gamma)_k^{-1}\boldsymbol{v}_i$ is applied with $\mathcal{O}(N) = \mathcal{O}(n^2)$ floating-point operations, and then $\boldsymbol{S}_k^T\widetilde{\boldsymbol{v}}_i = (\boldsymbol{V}_{\boldsymbol{X}_k}\otimes\boldsymbol{U}_{\boldsymbol{X}_k})\widetilde{\boldsymbol{v}}_i$ is computed. While a straightforward implementation of the latter would require $\mathcal{O}(N^2) = \mathcal{O}(n^4)$ floating-point operations, exploiting Kronecker product properties can bring down the cost of this operation to $\mathcal{O}(n^3) = \mathcal{O}(N^{3/2})$, by computing $\boldsymbol{S}_k^T\widetilde{\boldsymbol{v}}_i = \mathrm{vec}(\boldsymbol{U}_{\boldsymbol{X}_k}^T\mathrm{vec}^{-1}(\boldsymbol{v}_i)\boldsymbol{V}_{\boldsymbol{X}_k})$.

We emphasize that the incorporation of the flexible "preconditioners" does not increase the order of computational complexity and is very practical, since operations are done on matrices of size $n \times n$ (n is the dimension of the image). In particular, the full SVD's of $n \times n$ matrices can be computed easily with MATLAB's built-in `svd` function (this is what we used in our numerical experiments); one can also use Lanczos bidiagonalization [47] or randomized SVD [27] to compute the approximate leading singular values and vectors.

Chapter 5

Low Rank Regularization: Numerical Experiments

In this Chapter, we present results of numerical experiments on several image processing problems to demonstrate the performance of the new IRN-GMRES-NNRp, IRN-LSQR-NNRp, FGMRES-NNRp, and FLSQR-NNRp methods. Variants of FGMRES-NNRp and FLSQR-NNRp (marked with "(v)") are also tested. To shorten the acronyms, we omit p when $p = 1$, which means IRN-GMRES-NNR denotes IRN-GMRES-NNRp when $p = 1$, etc. Examples are generated using *IR Tools* [17].

In general, we compare the performances of the proposed methods to standard Krylov subspace methods GMRES and LSQR, also used in a hybrid fashion. We also test against the low-rank-projected GMRES (RS-LR-GMRES) method proposed in [48], which leads us to the derivation of alternative low-rank projection solvers LR-FGMRES and LR-FLSQR described in the upcoming section (see Section 5.1). These

solvers can be (re)casted into the framework of flexible Krylov methods and can work with both square and rectangular systems (4.1). The singular value thresholding (SVT) algorithm [7] will also be addressed, which was originally proposed to minimize matrix nuclear norm for low-rank matrix completion problems, and can be extended to problems with linear constraints of the form

$$\min_{\boldsymbol{x}} \tau\|\mathrm{vec}^{-1}(\boldsymbol{x})\|_* + \frac{1}{2}\|\mathrm{vec}^{-1}(\boldsymbol{x})\|_F^2 \quad \text{subject to} \quad \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}, \text{ where } \mathrm{vec}^{-1}(\boldsymbol{x}) = \boldsymbol{X}. \tag{5.1}$$

The kth iteration of the SVT algorithm for (5.1) reads

$$\begin{cases} \boldsymbol{X}_k = \mathcal{D}_\tau(\boldsymbol{A}^T\boldsymbol{y}_{k-1}) \\ \boldsymbol{y}_k = \boldsymbol{y}_{k-1} + \delta_k(\boldsymbol{b} - A\boldsymbol{x}_k) \end{cases}, \tag{5.2}$$

where δ_k is a step size and $\mathcal{D}_\tau$ is the *singular value shrinkage operator*, defined as

$$\mathcal{D}_\tau(\boldsymbol{X}) = \boldsymbol{U}_{\boldsymbol{X}}\mathcal{D}_\tau(\boldsymbol{\Sigma}_{\boldsymbol{X}})\boldsymbol{V}_{\boldsymbol{X}}^T, \quad \mathcal{D}_\tau(\boldsymbol{\Sigma}_{\boldsymbol{X}}) = \max\{\boldsymbol{\Sigma}_{\boldsymbol{X}} - \tau\boldsymbol{I}, \boldsymbol{0}\},$$

where $\boldsymbol{X} = \boldsymbol{U}_{\boldsymbol{X}}\boldsymbol{\Sigma}_{\boldsymbol{X}}\boldsymbol{V}_{\boldsymbol{X}}^T$ is the SVD of $\boldsymbol{X}$, $\boldsymbol{0}$ is a matrix of zeros, and the maximum is taken component-wise. Although (5.1) is not the same problem as (4.2), they are similar in that both penalize the nuclear norm of $\mathrm{vec}^{-1}(\boldsymbol{x})$ and they respect the constraint $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$.

5.1 Low Rank Projection Methods: classical and new approaches

When solving square well-posed linear systems coming from the discretization of some instances of stochastic or time-dependent PDEs, a suitable rearrangement of the solution is expected to be low-rank: for this reason, schemes that incorporate low-rank projections within the basis vectors and the approximate solution obtained by a Krylov method have been proposed in the literature. In the following we summarize the working ideas underlying the so-called restarted low-rank-projected GMRES (RS-LR-GMRES) method proposed in [48].

Recall from Section 2.4.1 that the kth iteration of GMRES updates the partial Arnoldi factorization and computes the approximate solution as follows:

$$\boldsymbol{A}\boldsymbol{V}_k = \boldsymbol{V}_{k+1}\boldsymbol{H}_k\,, \quad \boldsymbol{x}_k = \boldsymbol{V}_k\boldsymbol{y}_k\,, \text{ where } \boldsymbol{y}_k = \arg\min_{\boldsymbol{y}\in\mathbb{R}^k} \|\boldsymbol{H}_k\boldsymbol{y} - \|\boldsymbol{b}\|_2\boldsymbol{e}_1\|_2\,. \tag{5.3}$$

Since matrix-vector products and vector sums of low-rank vectorized matrices increase the rank of the latter, there is no guarantee that the new basis vectors $\boldsymbol{v}_k$ for the solution nor the new solution $\boldsymbol{x}_k$ are low-rank. To force the basis vector for the solution and the approximate solution to be low-rank, a truncation operator should be incorporated into the GMRES algorithm. Given a vectorized matrix $\boldsymbol{c} = \mathrm{vec}(\boldsymbol{C})$, and given a desired low-rank κ for $\boldsymbol{C}$, one can define a truncation operator $\tau_\kappa(\boldsymbol{c})$ by

the following standard operations:

$$\left[\begin{array}{l} 1.\ \text{Take } \boldsymbol{C} = \text{vec}^{-1}(\boldsymbol{c}); \\ 2.\ \text{Compute the SVD of } \boldsymbol{C},\ \boldsymbol{C} = \boldsymbol{U_C}\boldsymbol{\Sigma_C}\boldsymbol{V_C}^T; \\ 3.\ \text{Compute } \boldsymbol{C}_\kappa = \boldsymbol{U_C}(:, 1:\kappa)\boldsymbol{\Sigma_C}(1:\kappa, 1:\kappa)\boldsymbol{V_C}(:, 1:\kappa)^T; \\ 4.\ \text{Take } \tau_\kappa(\boldsymbol{c}) = \text{vec}(\boldsymbol{C}_\kappa). \end{array}\right. \tag{5.4}$$

RS-LR-GMRES is a restarted version of the standard GMRES method where the basis vectors for the solution are truncated at each inner iteration, and the solution itself is truncated at the beginning of each outer iteration. Note that truncating the basis vectors does not guarantee that the solution has low rank (which is the reason we still need to truncate the approximate solution). The reason for truncating the basis vectors is to keep the original solution rank from increasing drastically, since it is computed as a linear combination of basis vectors. More precisely, at the ℓth outer iteration of RS-LR-GMRES, one takes $\boldsymbol{v}_1 = \boldsymbol{r}_{\ell-1}/\|\boldsymbol{r}_{\ell-1}\|_2$, where $\boldsymbol{r}_{\ell-1} = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_{\ell-1}$, and, at the kth inner iteration, one computes

$$\boldsymbol{v}_k = \tau_\kappa(\boldsymbol{v}_k), \tag{5.5}$$

and once m inner iterations are performed, the approximate solution at the ℓth outer iteration is computed as

$$\boldsymbol{x}_\ell = \tau_\kappa\left(\boldsymbol{x}_{\ell-1} + \boldsymbol{V}_m\boldsymbol{y}_m\right). \tag{5.6}$$

The operations in (5.5) and (5.6) heavily depend on the value κ of the truncated rank, which eventually coincides with the rank of the approximate solution. In the

framework of stochastic PDEs, a suitable estimate for κ can be obtained by first performing coarse-grid computations (see [48] for details, and [45, 67] for similar approaches). As in standard GMRES, RS-LR-GMRES computes a new basis vector for the solution by applying the linear operator $\boldsymbol{A}$ to the previous basis vector $\boldsymbol{v}_{k-1}$ and orthogonalizing it against the previous basis vectors $\boldsymbol{v}_i$, $i = 1, \ldots, k-1$. However, since the basis vectors are truncated to low rank, the matrix $\boldsymbol{V}_k$ does not have orthonormal columns anymore, and $\mathcal{R}(\boldsymbol{V}_m)$ is not a Krylov subspace anymore. This remark leads us to the derivation of alternative low-rank projection solvers, which can be (re)casted into the framework of flexible Krylov methods and can work with both square and rectangular systems (4.1).

5.1.1 Low-rank flexible GMRES (LR-FGMRES) and low-rank flexible LSQR (LR-FLSQR)

Recall from Section 2.4.3 that starting with $\boldsymbol{x}_0 = \boldsymbol{0}$, at the kth iteration, FGMRES updates a partial flexible Arnoldi factorization and computes the kth approximate solution as follows:

$$\boldsymbol{A}\boldsymbol{Z}_k = \boldsymbol{V}_{k+1}\boldsymbol{H}_k\,, \quad \boldsymbol{x}_k = \boldsymbol{Z}_k\boldsymbol{y}_k\,, \text{ where } \boldsymbol{y}_k = \arg\min_{\boldsymbol{y}\in\mathbb{R}^k} \|\boldsymbol{H}_k\boldsymbol{y} - \|\boldsymbol{b}\|_2\boldsymbol{e}_1\|_2\,, \tag{5.7}$$

where $\boldsymbol{V}_{k+1} = [\boldsymbol{v}_1,\ \ldots\ , \boldsymbol{v}_{k+1}] \in \mathbb{R}^{N\times(k+1)}$ has orthonormal columns, $\boldsymbol{H}_k \in \mathbb{R}^{(k+1)\times k}$ is upper Hessenberg, $\boldsymbol{Z}_k = [\boldsymbol{P}_1\boldsymbol{v}_1, \ldots, \boldsymbol{P}_k\boldsymbol{v}_k] \in \mathbb{R}^{N\times k}$ has columns that span the approximation subspace for the solution and $\boldsymbol{P}_i$ is an iteration-dependent preconditioner. In the particular case of low-rank truncation, we set $\boldsymbol{P}_i\boldsymbol{v}_i = \tau_{\kappa_B}(\boldsymbol{v}_i)$, i.e., the

truncation operator defined in (5.4), so that $\text{rank}(\text{vec}^{-1}(\boldsymbol{Z}_k\boldsymbol{e}_i)) = \kappa_B,\ i = 1, \ldots, k)$. The subscript B for the truncation rank κ_B suggests that the truncation is done on the original basis vectors $\boldsymbol{v}_i$'s. The resulting algorithm is called "LR-FGMRES", and it is summarized in Algorithm 9. Note that the approximate solution computed as in (2.18) is also truncated to guarantee rank κ (in general, we assume $\kappa_B \neq \kappa$). Also LR-FGMRES is started with $\boldsymbol{x}_0 = \boldsymbol{0}$, to guarantee that the basis vectors for the solution (rather than a correction thereof) are low-rank.

Algorithm 9 LR-FGMRES

1: Inputs: $\boldsymbol{A}$, $\boldsymbol{b}$, τ_{κ_B}, τ_κ
2: Take $\boldsymbol{v}_1 = \boldsymbol{b}/\|\boldsymbol{b}\|_2$
3: **for** $i = 1, 2, \ldots$ until a stopping criterion is satisfied **do**
4: Compute $\boldsymbol{z}_i = \tau_{\kappa_B}(\boldsymbol{v}_i)$ and $\boldsymbol{w} = A\boldsymbol{z}_i$
5: Compute $h_{ji} = \boldsymbol{w}^T\boldsymbol{v}_j$ for $j = 1, \ldots, i$ and set $\boldsymbol{w} = \boldsymbol{w} - \sum_{j=1}^{i} h_{ji}\boldsymbol{v}_j$
6: Compute $h_{i+1,i} = \|\boldsymbol{w}\|_2$, and if $h_{j+1,j} \neq 0$, take $\boldsymbol{v}_{i+1} = \boldsymbol{w}/h_{i+1,i}$
7: **end for**
8: Compute $\boldsymbol{y}_k = \arg\min_{\boldsymbol{y}} \|\boldsymbol{H}_k\boldsymbol{y} - \|\boldsymbol{b}\|_2\boldsymbol{e}_1\|_2^2$ and take $\boldsymbol{x}_k = \tau_\kappa(\boldsymbol{Z}_k\boldsymbol{y}_k)$

A few remarks are in order. Differently from the kth iteration in the inner cycle of the RS-LR-GMRES method (5.5), the kth iteration of LR-FGMRES expands the approximation subspace by modifying (i.e., truncating) the previous orthonormal basis vector for the space $\mathcal{R}([\boldsymbol{b}, \boldsymbol{A}\boldsymbol{Z}_k])$. Analogously to RS-LR-GMRES, the basis vectors for the approximate LR-FGMRES solution are all of rank κ_B, are not orthogonal, and do not span a Krylov subspace. Differently from RS-LR-GMRES, the basis vector for the space $\mathcal{R}([\boldsymbol{b}, \boldsymbol{A}\boldsymbol{Z}_k])$ are orthogonal. Also, the kth LR-FGMRES approximate solution is obtained by solving an order-k projected least squares problem that is formally analogous to the GMRES one (see (5.3) and (5.7)).

With LR-FGMRES in place, the extension to more general matrices $\boldsymbol{A} \in \mathbb{R}^{M\times N}$, with M not necessarily equal to N, can be naturally devised considering the flexible Golub-Kahan (FGK) process [11]. Recall that taking $\boldsymbol{x}_0 = \boldsymbol{0}$ as initial guess, the kth FGK iteration updates partial factorizations of the form

$$\boldsymbol{A}\boldsymbol{Z}_k = \boldsymbol{U}_{k+1}\boldsymbol{M}_k \quad \text{and} \quad \boldsymbol{A}^T\boldsymbol{U}_{k+1} = \boldsymbol{V}_{k+1}\boldsymbol{T}_{k+1}, \tag{5.8}$$

where the columns of $\boldsymbol{U}_{k+1} \in \mathbb{R}^{M\times(k+1)}$, $\boldsymbol{V}_{k+1} \in \mathbb{R}^{N\times(k+1)}$ are orthonormal, $\boldsymbol{M}_k \in \mathbb{R}^{(k+1)\times k}$ is upper Hessenberg, $\boldsymbol{T}_{k+1} \in \mathbb{R}^{(k+1)\times(k+1)}$ is upper triangular, $\boldsymbol{Z}_k = [\boldsymbol{P}_1\boldsymbol{v}_1, \ldots, \boldsymbol{P}_k\boldsymbol{v}_k] \in \mathbb{R}^{N\times k}$ has columns that span the approximation subspace for the solution, and $\boldsymbol{P}_i$ is an iteration-dependent preconditioner that is applied to $\boldsymbol{v}_i$. In the particular case of low-rank truncation, $\boldsymbol{P}_i\boldsymbol{v}_i = \tau_{\kappa_B}(\boldsymbol{v}_i)$, as defined in (5.4), so that $\operatorname{rank}(\operatorname{vec}^{-1}(\boldsymbol{Z}_k\boldsymbol{e}_i)) = \kappa_B,\ i = 1, \ldots, k$. The flexible LSQR method (FLSQR) uses the FGK process (5.8) to generate iterates of the form $\boldsymbol{x}_k = \boldsymbol{Z}_k\boldsymbol{y}_k$, where the vector $\boldsymbol{y}_k$ is computed as $\boldsymbol{y}_k = \arg\min_{\boldsymbol{y}} \left\| \boldsymbol{M}_k\boldsymbol{y} - \|\boldsymbol{b}\|_2\boldsymbol{e}_1 \right\|_2^2$. When rank-truncation of the basis vectors takes place at each iteration, and the final approximate solution is rank-truncated as well, the resulting algorithm is called "LR-FLSQR", and it is summarized in Algorithm 10.

Note that, similarly to RS-LR-GMRES, both LR-FGMRES and LR-FLSQR are quite heuristic. Although the low-rank projection idea can be formulated in the flexible framework, we lack a formal formulation of the problem that is being solved, and also a justification of why they work. Strategies for selecting κ_B and κ are not so clear either. To stabilize the behavior of LR-FGMRES as the iterations proceed, one may consider imposing additional Tikhonov regularization on the projected least-

Algorithm 10 LR-FLSQR

Inputs: $\boldsymbol{A}$, $\boldsymbol{b}$, τ_{κ_B}, τ_{κ}
Take $\boldsymbol{u}_1 = \boldsymbol{b}/\|\boldsymbol{b}\|_2$
for $i = 1, 2, \ldots,$ until a stopping criterion is satisfied **do**
    Compute $\boldsymbol{w} = \boldsymbol{A}^T\boldsymbol{u}_i$, $t_{ji} = \boldsymbol{w}^T\boldsymbol{v}_j$ for $j = 1, \ldots, i-1$
    Set $\boldsymbol{w} = \boldsymbol{w} - \sum_{j=1}^{i-1} t_{ji}\boldsymbol{v}_j$, compute $t_{ii} = \|\boldsymbol{w}\|$ and take $\boldsymbol{v}_i = \boldsymbol{w}/t_{ii}$
    Compute $\boldsymbol{z}_i = \tau_{\kappa_B}(\boldsymbol{v}_i)$ and $\boldsymbol{w} = \boldsymbol{A}\boldsymbol{z}_i$
    Compute $m_{ji} = \boldsymbol{w}^T\boldsymbol{u}_j$ for $j = 1, \ldots, i$ and set $\boldsymbol{w} = \boldsymbol{w} - \sum_{j=1}^{i} m_{ji}\boldsymbol{u}_j$
    Compute $m_{i+1,i} = \|\boldsymbol{w}\|$ and take $\boldsymbol{u}_{i+1} = \boldsymbol{w}/m_{i+1,i}$
end for
Compute $\boldsymbol{y}_k = \arg\min_{\boldsymbol{y}} \|\boldsymbol{M}_k\boldsymbol{y} - \|\boldsymbol{b}\|_2\boldsymbol{e}_1\|_2^2$ and take $\boldsymbol{x}_k = \tau_{\kappa}(\boldsymbol{Z}_k\boldsymbol{y}_k)$

squares problem in (2.18), in a hybrid fashion; the same holds for LR-FLSQR (see Sections 4.4 and 5.2 for more details).

5.2 Numerical Examples

In this section, three types of imaging inverse problems are presented, and a variety of solvers, including the newly developed low rank methods, are compared. Before getting started, we briefly remark on some details for some of the parameters in these methods.

The Schatten-p function is introduced in Section 4.2 as a smooth approximation for $\|\cdot\|_{*,p}$. The smooth approximation allows for further derivations including computation of optimality conditions, where the "smoothing coefficient" γ is crucial. However, γ is not so crucial numerically, and we can set it to 0 without affecting the results (compared to using a very small γ). However, to be consistent with Algorithms 7 and 8, in our experiments, we have set the initial value of γ to 10^{-10}, and every time we need to decrease γ, we divide the current γ value by 2.

Regarding the comparisons with the low-rank projection methods presented in Section 5.1.1, there are no universal and theoretically informed ways of choosing the truncation ranks for the solutions and for the basis vectors of the solution subspace. Hence, for all test problems, we experiment on a reasonable number of trials, each with different truncation rank choices, and select the best performing rank out of all ranks tested. For simplicity, we consider the same truncation rank for basis vectors and solutions ($\tau_{\kappa_B} = \tau_\kappa$). We follow the same process to choose the number of restarts and the number of iterations for each restart for RS-LR-GMRES, as well as the shrinkage threshold τ in SVT; strategies to select the step size for SVT are described in [7].

Example 1: Binary Star. We consider an image deblurring problem involving a simulated binary star test image of size 256×256: this test image has rank 2. The true image is displayed in the leftmost frame of Figure 5.2. A standard Gaussian blur is applied to the test image, and Gaussian white noise of level $\|\boldsymbol{\eta}\|_2/\|\boldsymbol{b}^{\text{ex}}\|_2 = 10^{-3}$ is added. The blurred and noisy images are shown in Figure 5.2, second frame from the left. Due to the presence of noise, the blurred image has full rank. For this example, the blurring operator $\boldsymbol{A}$ is square of size 65536×65536, hence GMRES-related methods are used for comparison, namely: GMRES, IRN-GMRES-NNR, FGMRES-NNR, LR-FGMRES and RS-LR-GMRES (i.e., we only consider the case $p = 1$ here). SVT is also taken into consideration. The truncation rank for LR-FGMRES and RS-LR-GMRES is set to 30 for both basis vectors and approximate solutions (i.e., $\tau_{\kappa_B} = \tau_\kappa = 30$). RS-LR-GMRES is restarted every 40 iterations. The step size for SVT is set to be $\delta_k = \delta = 2$ and the singular value shrinkage threshold τ

is 1. Note that, although the true solution has only rank 2, setting truncation rank to 2 for low rank methods produces solutions of worse quality (compared to setting the rank to 30). This might be because of the inherent ill-posedness of the problem, which makes it harder to obtain solutions with desired properties (e.g., with rank 2): indeed, if we do truncate to rank 2, a lot of information about the solution might be lost.

Figure 5.1 displays the histories of relative errors $\|\boldsymbol{x}^{\text{ex}} - \boldsymbol{x}_m\|_2/\|\boldsymbol{x}^{\text{ex}}\|_2$ for the first 200 iterations (i.e., $m = 1, \ldots, 200$) of these methods. For IRN-GMRES-NNR, 4 outer cycles were run, each with a maximum of 50 iterations: a new outer cycle is initiated as soon as the discrepancy principle is satisfied in the inner cycle. No additional regularization is used (i.e., $\widehat{\lambda} = 0$ for all methods).

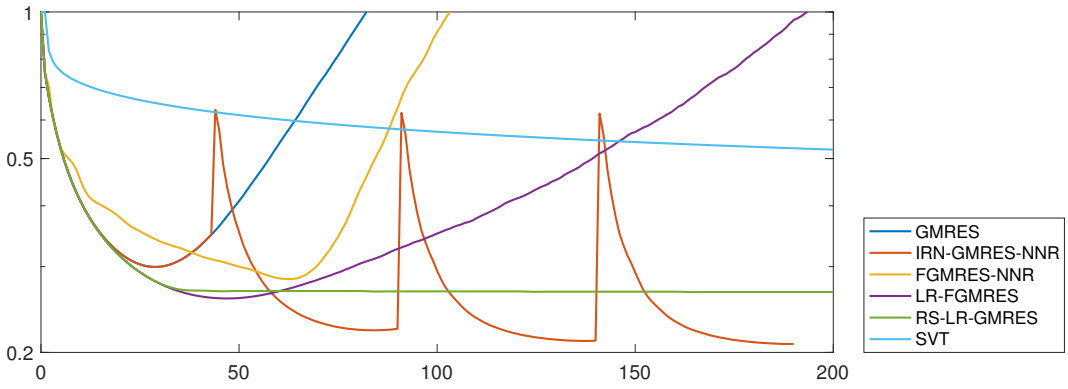


Figure 5.1: *Example 1.* Relative errors vs. number of iterations for GMRES-based methods and SVT.

We can observe from Figure 5.1 that when the truncation ranks are chosen reasonably, LR-FGMRES and RS-LR-GMRES both produce a less pronounced semi-convergence behavior than GMRES, with LR-FGMRES attaining a smaller relative

error than RS-LR-GMRES. FGMRES-NNR, on the other hand, shows slower semi-convergence than GMRES, but it also converges to a slightly better relative error. IRN-GMRES-NNR behaves especially well in this case, with significantly reduced relative errors even at the end of the second outer cycle. The "jumps" at the beginning of each outer IRN-GMRES-NNR iteration are due to the strategy used for restarts (the older basis vectors are cleared at each restart).

Figure 5.2 displays the exact and the corrupted images, as well as the best reconstructions computed by LR-FGMRES and IRN-GMRES-NNR: these are obtained at the 47th and the 189th (total) iteration of LR-FGMRES and IRN-GMRES-NNR, respectively. By looking at relative errors in Figure 5.1, we see that LR-FGMRES is the second best out of all methods, and yet the quality of the solution is inferior compared to IRN-GMRES-NNR. Compared to the LR-FGMRES solution, the IRN-GMRES-NNR one is a more truthful reconstruction of the exact image: it not only has less artifacts immediately around the stars, but also has less background noise, in the sense that the pixel intensities in the background are closer to the true ones (as it can be seen by looking at the background color).

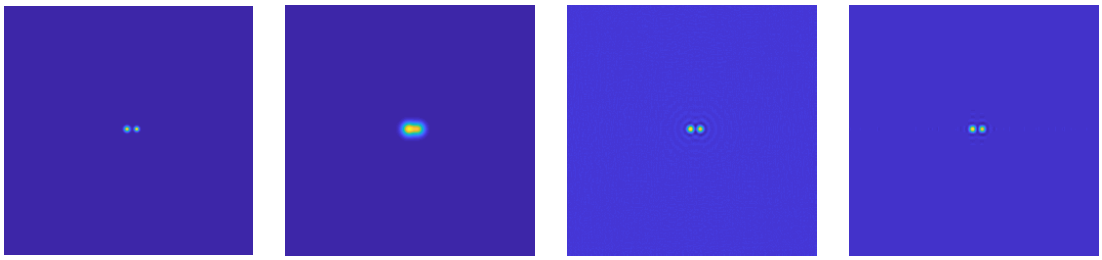


Figure 5.2: *Example 1.* Exact and corrupted test images, together with the best reconstructions obtained by the LR-FGMRES and the IRN-GMRES-NNR methods.

More details can be spotted if we zoom into the central part (51×51 pixels) of the computed images, as shown in Figure 5.3: here the best LR-FGMRES reconstruction, as well as the IRN-GMRES-NNR reconstructions at the end of the 2nd, 3rd and 4th inner cycles are displayed. It is clear that the IRN reconstructions are improving over each outer cycle, and that even the solution at the end of the 2nd cycle is significantly better than the LR-FGMRES solution, which means that not all four outer iterations need to be run to achieve solutions of superior qualities (even if more outer iterations allow further improvement in the solution).

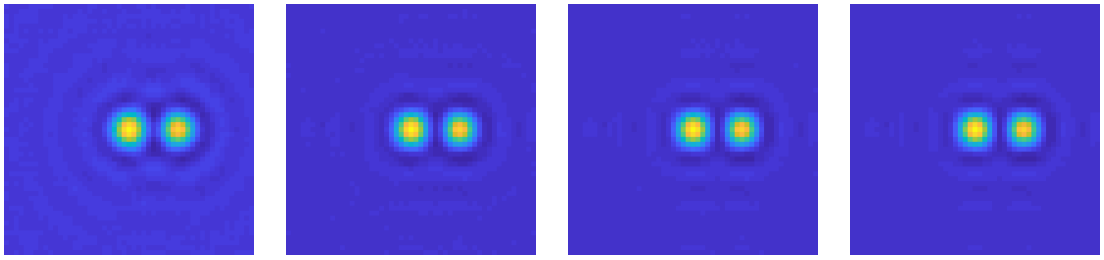


Figure 5.3: *Example 1.* Zoom-ins of the LR-FGMRES best solution, and the IRN-GMRES-NNR solutions at the end of each inner cycle.

Figure 5.4 displays surface plots of the central part (51×51 pixels) of the test problem data, as well as the best reconstructed images (for RS-LR-GMRES and FGMRES-NNR these are obtained at the 165th and the 63th (total) iterations, respectively). It can be seen that for all the solutions shown here, the reconstructed central two stars approximately have the same intensity, although they are somewhat less intense than in the exact image. These surface plots also confirm our earlier observation that IRN-GMRES-NNR does an exceptional job removing background

noise. In addition, FGMRES-NNR also gives a good background reconstruction.

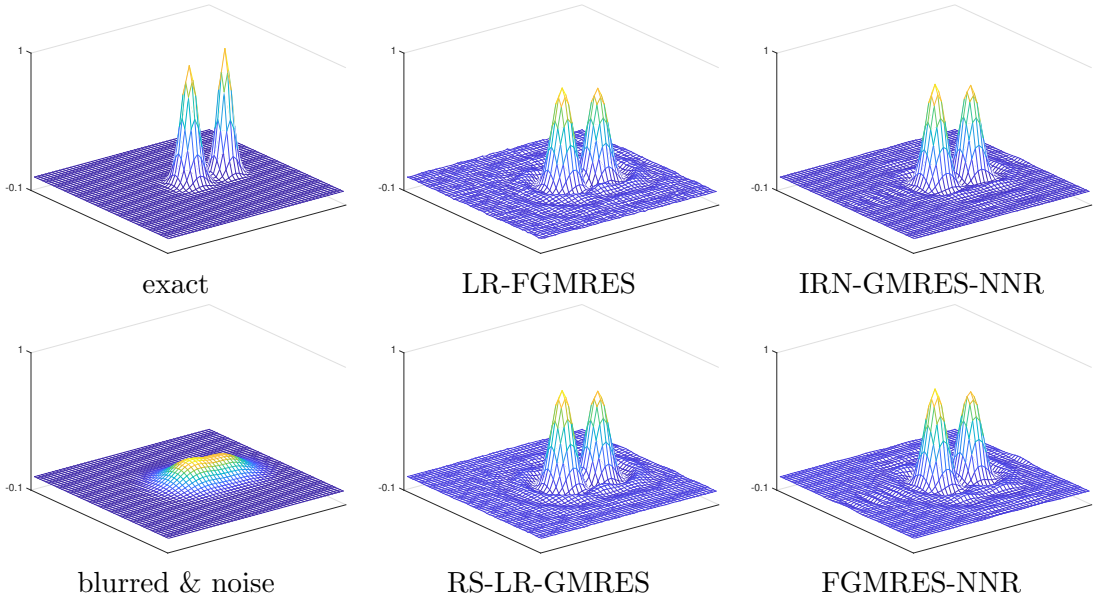


Figure 5.4: *Example 1.* Zoomed-in surfaces of the exact solution and the available data, as well as the best reconstructions obtained by the new GMRES-based methods.

Finally, Figure 5.5 displays the singular values of the best solutions obtained adopting different GMRES-based solvers, as well as the evolution of the singular values of the solution at the end of each inner IRN-GMRES-NNR cycle (matching the reconstructions displayed in Figure 5.3). The singular values are "normalized" (i.e., divided by the largest one), and the graphs are cropped to focus on the relevant values. Looking at the displayed values, we can conclude that the solutions computed by all the low-rank solvers have indeed some low-rank properties, with very quickly-decaying large singular values followed by slowly-decaying smaller singular values. Compared to GMRES, the new FGMRES-NNR and IRN-GMRES-NNR methods give solutions that have a more pronounced low rank, as shown by the large gaps

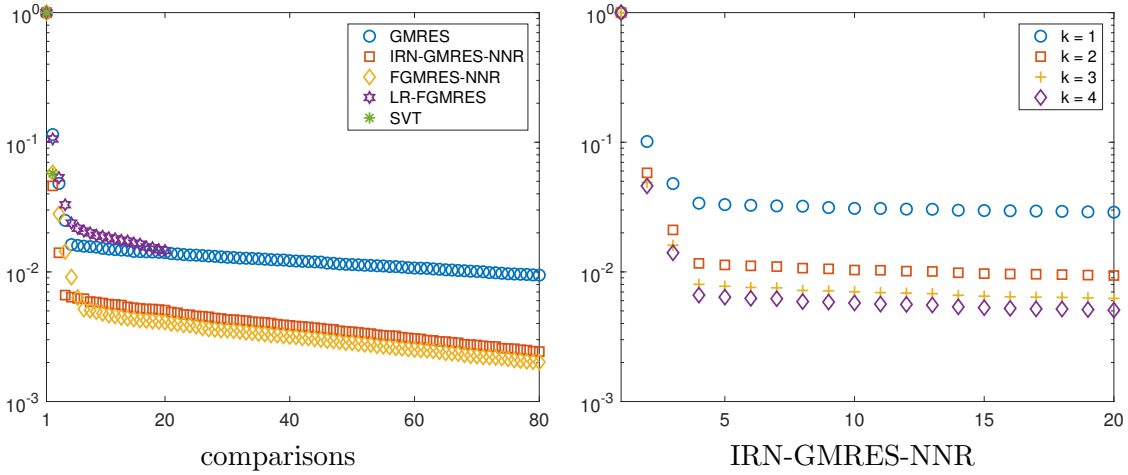


Figure 5.5: *Example 1.* Left frame: normalized singular values of the best solutions computed by each GMRES-based method. Right frame: evolution of the singular values of the solutions computed by IRN-GMRES-NNR at each outer iteration. Singular values less than 10^{-3} are omitted.

between the smaller singular values of the solutions computed by these methods. Regarding IRN-GMRES-NNR, the evolution of the singular values stabilizes as we move toward later outer iterations, which validates the stopping criterion proposed in Section 4.5.

Example 2: Limited Angle Parallel-Ray Tomography. We consider a computed tomography (CT) test problem, modeling an undersampled X-ray scan with parallel beam geometry. This is a so called "limited angle" CT reconstruction problem, where the viewing angles for the object span less than 180 degrees. A smooth and rank-4 phantom is considered, as shown in the leftmost frame of Figure 5.7 (note that the yellow straight lines in the northwestern corner do not belong to the phantom; they are shown for later purposes). Gaussian white noise of level 10^{-2} is

added to the data. The coefficient matrix $\boldsymbol{A}$ has size 32942×65536. Because of this, among the new solvers, only LR-FLSQR, FLSQR-NNRp, FLSQR-NNRp(v), and IRN-LSQR-NNRp will be tested, against their standard counterpart LSQR. Recall that FLSQR-NNRp(v) is the FLSQR-NNRp variant that defines the preconditioners using the basis vectors of the solution subspace. The hybrid strategy is not used here, meaning that we set $\widehat{\lambda} = 0$ for all methods. For this test problem, we consider both the values $p = 1$ and $p = 0.75$ (recall that, when $p = 1$, we omit p from the notation). The results obtained running the available low-rank solvers SVT and RS-LR-GMRES are shown, too. Note that RS-LR-GMRES only works for square matrices $\boldsymbol{A}$, hence this solver is tested on the normal equations $\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{x} = \boldsymbol{A}^T\boldsymbol{b}$, which is not the problem solved by the other methods (therefore this comparison may not be completely fair). Parameters for SVT are chosen to be: step size $\delta_k = \delta = 8 \times 10^{-5}$ and threshold $\tau = 100$. RS-LR-GMRES is set to restart every 20 iterations. The truncation rank is 10 for both basis vectors and solutions, and for both the LR-FLSQR and the RS-LR-GMRES methods. The maximum number of iterations is 100 for all methods.

Figure 5.6 displays the history of the relative errors for LSQR, LR-LSQR, FLSQR-NNRp, FLSQR-NNRp(v), and IRN-LSQR-NNRp, for $p = 1$ and $p = 0.75$. Figure 5.7 displays the exact phantom together with the best reconstructions obtained by LSQR, FLSQR-NNRp(v), and IRN-LSQR-NNR. Figure 5.8 displays surface plots of the northwestern corner of the exact and reconstructed phantoms (64×64 pixels, as highlighted in the leftmost frame of Figure 5.7).

Looking at relative errors in Figure 5.6, it is obvious that the winners are the

FLSQR-NNRp(v) methods, with both $p = 1$ and $p = 0.75$: they give the lowest relative errors, and the fastest semi-convergences. For this test problem, using a value of $p < 1$ lowers the relative error of FLSQR-NNRp(v); however, the same does not hold for IRN-LSQR-NNRp. Therefore we can conclude that the the choice of p is problem and solver dependent, and using $p < 1$ does not necessarily improve the quality of the solution. We regard $p = 1$ as a safe choice for this parameter. Although both the FLSQR-NNRp(v) methods with $p = 1$ and $p = 0.75$ perform well, the latter is able to further reduce the noise in the reconstructed solution, especially on the boundary.

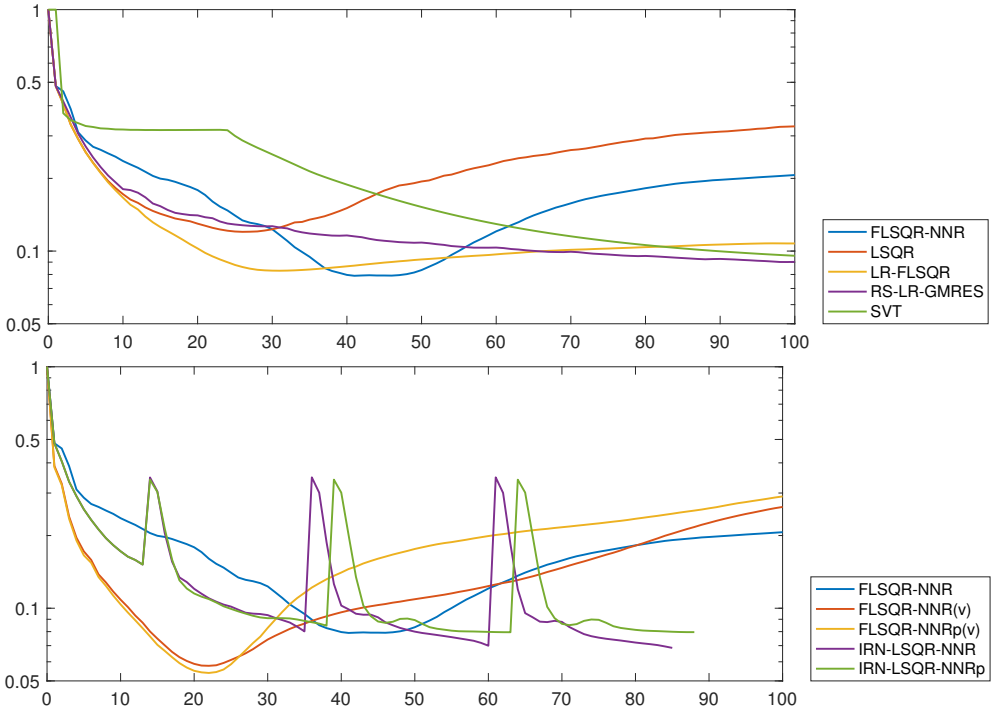


Figure 5.6: *Example 2.* Relative errors vs. number of iterations for different solvers. Upper frame: some of the new solvers are compared to the already available solvers. Lower frame: comparisons of different instances of the new solvers (here $p = 0.75$).

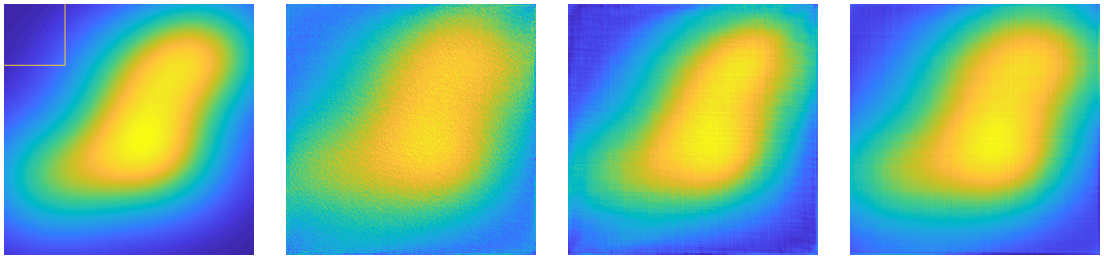


Figure 5.7: *Example 2.* Exact phantom and best reconstructions obtained by different solvers.

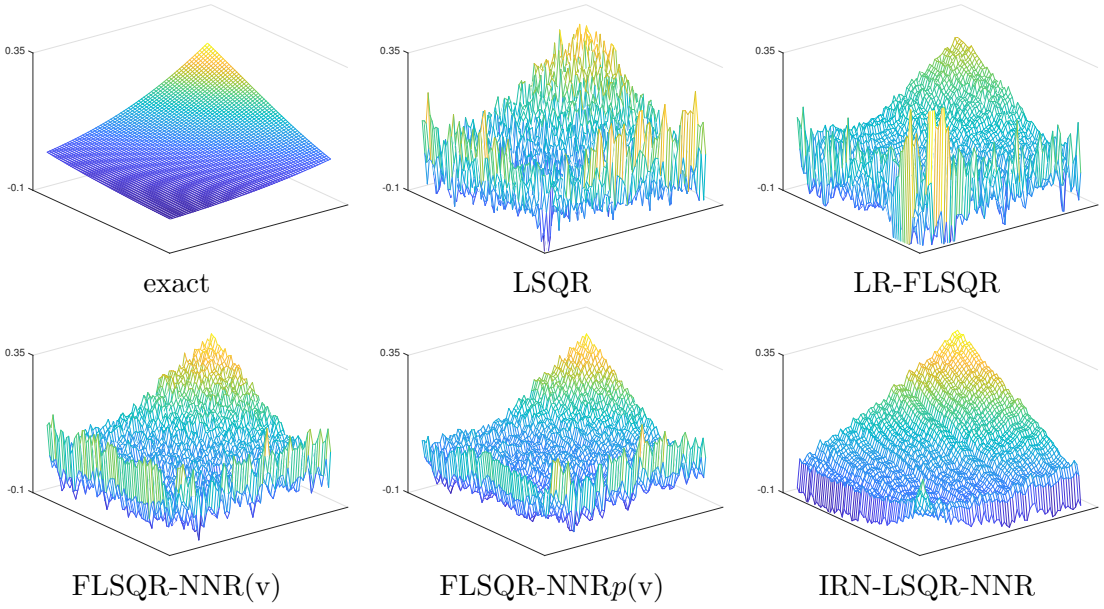


Figure 5.8: *Example 2.* Surface plots of the northwestern corner of the exact phantom (highlighted in Figure 5.7) and the best reconstructed phantoms computed by different solvers.

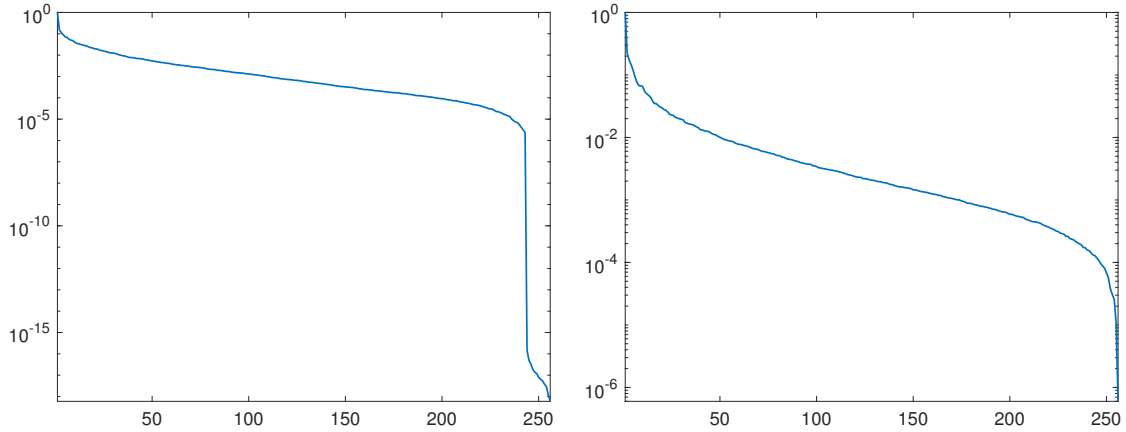
Looking at all the displayed results, the advantages of our new FLSQR-NNRp(v) and IRN-LSQR-NNR methods are evident. Namely, they produce smooth solutions

that preserve the original concave shape of the exact phantom, and they retain similar intensities of pixels at the same locations of the exact phantom (although the LR-FLSQR solution is smooth within the boundary, it fails to reconstruct intensity at the high point). Differences between FLSQR-NNRp(v) and IRN-LSQR-NNR reconstructions are clear, too: while both are smooth, the IRN-LSQR-NNR reconstruction has a less concave shape compared to that of FLSQR-NNRp(v), but a smoother boundary.

Example 3: Inpainting. We consider two different inpainting test problems. Inpainting is the process of restoring images that have missing or deteriorated parts. These images are likely to have quite a few lost pixels, either in the form of salt and pepper noise, or missing patches with regular or irregular shapes. The two examples considered here are of different nature: the first one has less structured and more randomly distributed missing patches, while the second one has more structured and regularly shaped missing parts. The corrupted images (shown in top-middle frames of Figures 5.11 and 5.13) are constructed by first applying a blur operator, and then superimposing the undersampling pattern to the ideally exact images (shown in the top-left frames of Figures 5.11 and 5.13). We follow this particular order of first blurring and then taking out pixels to simulate the real process of photo-taking. For both these test problems, white noise of level 10^{-2} is added to the data, and we consider purely iterative methods (i.e., $\widehat{\lambda} = 0$). We always take $p = 1$, and we run 100 iterations of all the methods.

Firstly, we consider a test problem where 58.2% of the pixels are missing (following some random and not very regular patterns). The exact image is commonly known as

the `house` test image, whose rank is 243 and has a total number of 65536 (256×256) pixels; the corrupted image has the same size and number of pixels, but out of which only 27395 are non-zero. A plot of the singular values of the exact image is shown in Figure 5.9(a). Correspondingly, the forward operator $\boldsymbol{A}$ is of size 27395×65536, so we have an underdetermined linear system: $\boldsymbol{A}$ is obtained by first applying a shaking blur, and by then undersampling the blurred image. This can be easily coded within the *IR Tools* framework.



(a). `house` (b). `peppers`

Figure 5.9: *Example 3.* Singular values of exact test images `house` and `peppers` scaled by the largest singular values respectively.

Figure 5.10 displays the history of the relative errors for LSQR, LR-FLSQR (with truncation of the basis vectors for the solution, as well as the solution, to rank 20), FLSQR-NNR, FLSQR-NNR(v) and IRN-LSQR-NNR. Figure 5.11 displays the exact and corrupted images, together with the best reconstructions obtained by the methods listed above: these correspond to the 16th, 32nd, 67th, 30th and 62nd iterations of LSQR, LR-FLSQR, FLSQR-NNR, FLSQR-NNR(v) and IRN-LSQR-

NNR, respectively (i.e., these are the iterations where the minimum relative error is attained over the total 100 iterations).

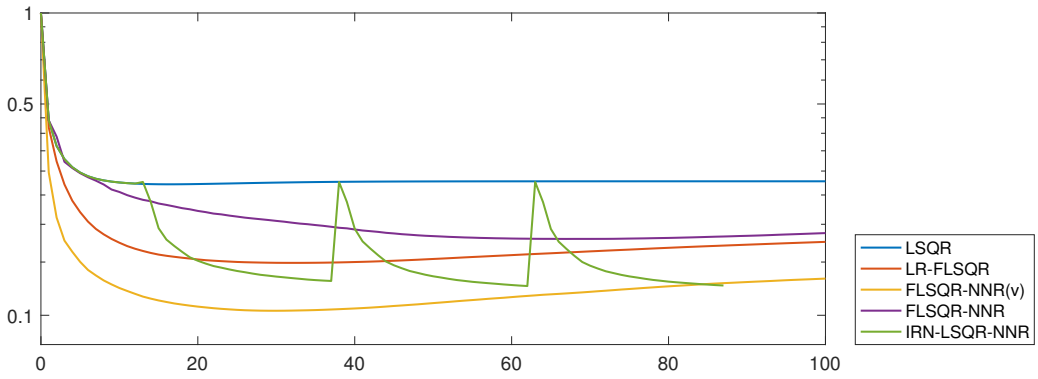


Figure 5.10: *Example 3* (`house`). Relative errors vs. number of iterations for different solvers.

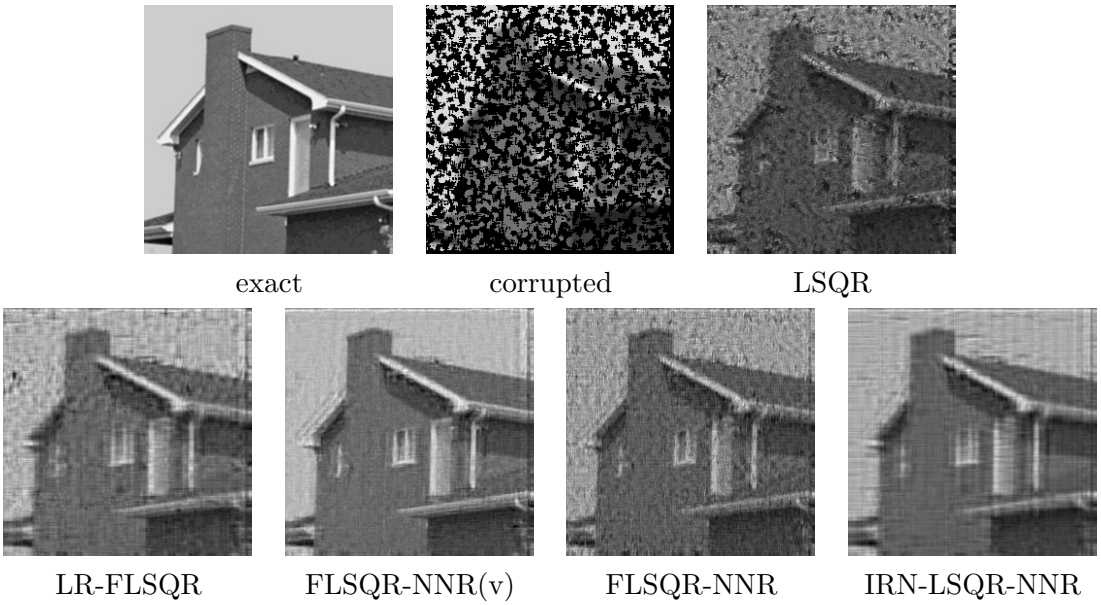


Figure 5.11: *Example 3* (`house`). Exact and corrupted images; best reconstructions obtained by standard and new solvers.

Secondly, we consider a test problem similar to the previous one, i.e., we take an exact image commonly known as the `peppers` test image, which has full rank (its singular values are shown in Figure 5.9(b)), and we obtain the forward operator $\boldsymbol{A}$ by first applying a shaking blur, and by then undersampling the blurred image. Here the exact image has a total number of 65536 (256×256) pixels, and only around 1.3% of pixels are missing and should be inpainted: differently from the previous problem, the missing pixels follow particular patterns (e.g., circles, squares, and rectangles), and this makes the inpainting task somewhat more challenging. Figure 5.12 displays the history of the relative errors for LSQR, LR-FLSQR (with truncation of the basis vectors for the solution, as well as the solution, to rank 50), FLSQR-NNR, FLSQR-NNR(v) and IRN-LSQR-NNR. Figure 5.13 displays the exact and corrupted images, together with the best reconstructions obtained by the methods listed above: these correspond to the 11th, 18th, 60th, 33rd and 34th iterations of LSQR, LR-FLSQR, FLSQR-NNR, FLSQR-NNR(v) and IRN-LSQR-NNR, respectively (i.e., these are the iterations where the minimum relative error is attained over the total 100 iterations).

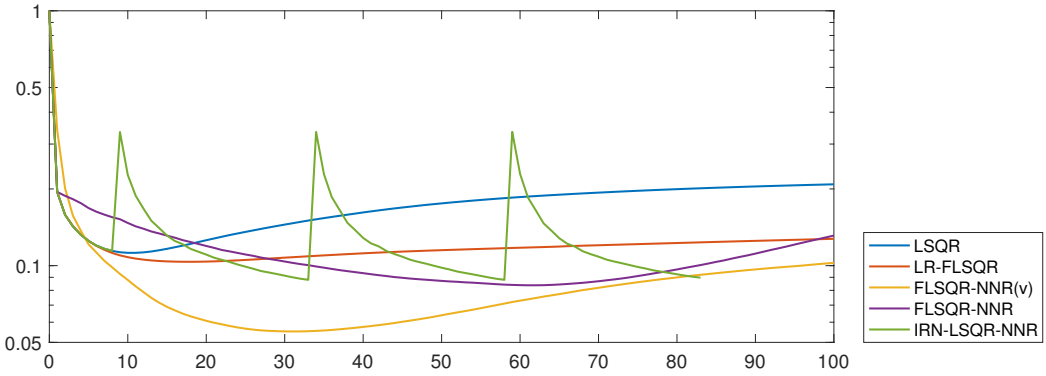


Figure 5.12: *Example 3* (`peppers`). Relative errors vs. number of iterations for different solvers.

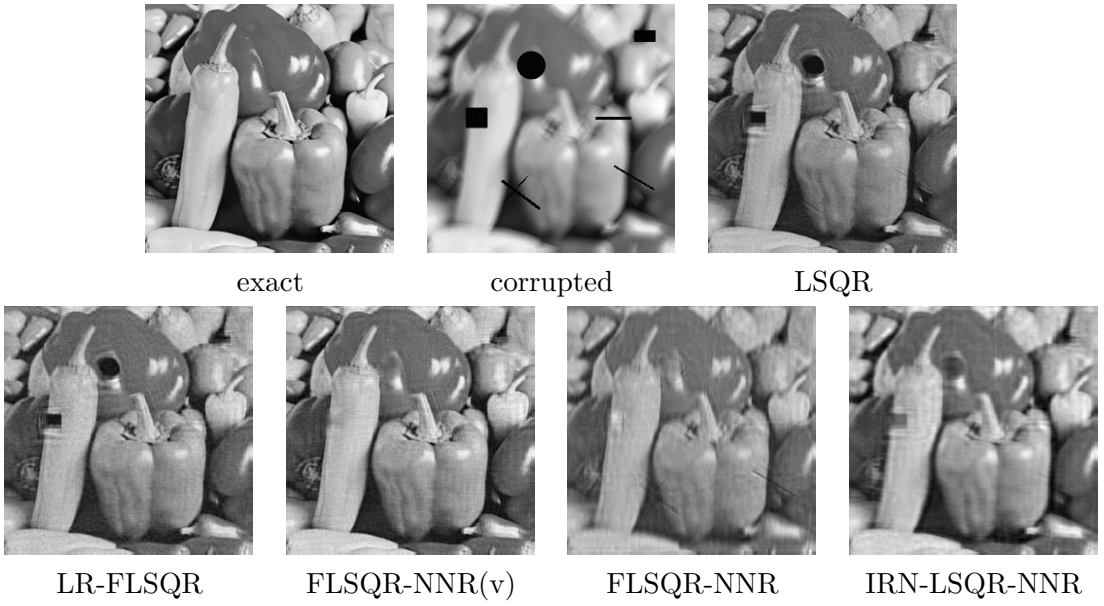


Figure 5.13: *Example 3* (`peppers`). Exact and corrupted images; best reconstructions obtained by standard and new solvers.

It is evident that FLSQR-NNR(v) achieves reconstructions of superior quality, including clarity, brightness, and smoothness. Its ability to fill-in missing spots with pixels that are of similar intensity to their surroundings is the best among all methods. The best reconstructions are computed by IRN-LSQR-NNR for the `house` test image, and by FLSQR-NNR for the `pepper` test image: in both cases, these methods are also good at removing noise and restoring missing pixels. However, for both test images, the reconstructions obtained by IRN-LSQR-NNR lack clarity compared to ones obtained by both FLSQR-NNR and FLSQR-NNR(v) methods; compared to the reconstructions obtained by LSQR and LR-FLSQR, they are anyway more desirable in terms of recovered brightness and fill-in of the missing pixels. Moreover, we have seen in these two examples that our newly proposed methods perform very well not

only for low rank, but also for full or nearly full rank image reconstruction, thanks to the regularizing properties of our newly derived "preconditioners" $(\boldsymbol{W}_p^{\gamma})_k$ and $\boldsymbol{S}_k$. Our methods can also be extensively tested for higher noise levels (for example, 10^{-1}) and yield similar results. However, for space considerations we are not able to show all of them here.

5.2.1 A Note on Regularization Parameters

In the previous examples we have seen that the IRN-NNR methods and the flexible Krylov NNR methods perform exceptionally well on image deblurring, tomography, and inpainting problems, producing superior reconstructions compared to existing methods including SVT, RS-LR-GMRES and the low-rank flexible Krylov methods inspired by RS-LR-GMRES, even without the use of additional regularization. In this section, we explore the effect of additional regularization (i.e., we set $\widehat{\lambda} \neq 0$) on the reconstructed images and the corresponding relative errors. In particular, additional regularization allows the new methods to be used in a hybrid fashion. We are going to observe that there is only little to negligible room for the methods to improve when they are used in a hybrid fashion (as their performance is already very good with $\widehat{\lambda} = 0$).

We consider three different ways of choosing the regularization parameter $\widehat{\lambda}$. (i) We take the "secant method" mentioned in Section 4.5, which updates the regularization parameter at each iteration using the discrepancy. (ii) We select the optimal regularization parameter which minimizes the 2-norm of the difference between the exact solution and the regularized solution at each iteration. Namely, when using

standard GMRES and LSQR, at the mth iteration we seek to minimize with respect to $\widehat{\lambda}$

$$\|\boldsymbol{x}^{\mathrm{ex}} - \boldsymbol{x}_{m,\widehat{\lambda}}\| = \|\boldsymbol{V}_m^T\boldsymbol{x}^{\mathrm{ex}} - \boldsymbol{V}_m^T\boldsymbol{x}_{m,\widehat{\lambda}}\| = \|\boldsymbol{V}_m^T\boldsymbol{x}^{\mathrm{ex}} - \boldsymbol{y}_{m,\widehat{\lambda}}\|\,;$$

when using the IRN methods we should incorporate the appropriate preconditioners $(\boldsymbol{W}_p^{\gamma})_k$ and $\boldsymbol{S}_k$ and, for all the iterations in the inner iteration cycle corresponding to the kth outer iteration, we seek to minimize with respect to $\widehat{\lambda}$

$$\|\widehat{\boldsymbol{x}}^{\mathrm{ex}} - \boldsymbol{V}_m\boldsymbol{y}_{m,\widehat{\lambda}}\| = \|\boldsymbol{V}_m^T\widehat{\boldsymbol{x}}^{\mathrm{ex}} - \boldsymbol{V}_m^T\boldsymbol{V}_m\boldsymbol{y}_{m,\widehat{\lambda}}\| = \|\boldsymbol{V}_m^T\widehat{\boldsymbol{x}}^{\mathrm{ex}} - \boldsymbol{y}_{m,\widehat{\lambda}}\|, \text{ where } \widehat{\boldsymbol{x}}^{\mathrm{ex}} = (\boldsymbol{W}_p^{\gamma})_k\boldsymbol{S}_k\boldsymbol{x}^{\mathrm{ex}}.$$

It is intrinsically difficult to implement this strategy for flexible Krylov subspace methods, because of the complexity of changing preconditioners at each iteration. (iii) We perform a manual exhaustive search. Namely, we first run the solvers multiple times using various regularization parameters $\widehat{\lambda}$, starting with a larger range and narrowing down to a smaller range containing the best parameter; we then record the minimum relative errors among all iterations for all values of $\widehat{\lambda}$, and select the corresponding $\widehat{\lambda}$. This approach is the most expensive one, and differs from the previous one in that the (optimal) regularization parameter $\widehat{\lambda}$ is fixed for all iterations. Of course, both the second and third approaches require the knowledge of the exact solution and we test them only to investigate the best possible performance of the hybrid approach.

Table 5.1 compares the performances (in terms of minimum relative error achieved by each method) of standard Krylov methods (GMRES and LSQR) and their IRN-NNR and flexible NNR (F-NNR) counterparts, with and without using a hybrid

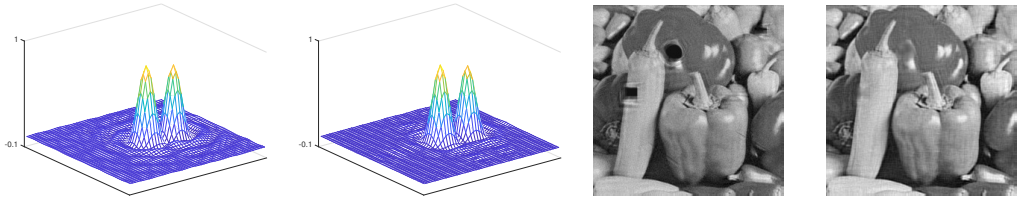
approach. In this way we can understand how the use of additional regularization affects each solver differently. The three parameter choice methods described above are called "Secant (i)", "Optimal (ii)" and "Fixed (iii)", respectively. All the previous examples are considered here. GMRES and its counterparts IRN-GMRES-NNR, FGMRES-NNR are used for Example 1, while LSQR and its counterparts IRN-LSQR-NNR, FLSQR-NNR(v) are used for Examples 2 and 3.

		$\widehat{\lambda}=0$	$\widehat{\lambda}\neq 0$	$\widehat{\lambda}=0$	$\widehat{\lambda}\neq 0$	$\widehat{\lambda}=0$	$\widehat{\lambda}\neq 0$	$\widehat{\lambda}=0$	$\widehat{\lambda}\neq 0$
		Example 1		*Example 2*		*Ex. 3* (`house`)		*Ex. 3* (`peppers`)	
Standard	(i)	0.2995	0.2528	0.1201	0.1389	0.2712	0.2715	0.1141	0.1138
	(ii)	0.2995	0.2268	0.1201	0.1201	0.2712	0.2710	0.1141	0.1138
	(iii)	0.2995	0.2268	0.1201	0.1183	0.2712	0.2710	0.1141	0.1138
IRN-NNR	(i)	0.2081	0.2096	0.0685	0.0696	0.1249	0.1250	0.0964	0.0967
	(ii)	0.2081	0.2292	0.0685	0.0685	0.1249	0.1249	0.0964	0.0964
	(iii)	0.2081	X	0.0685	0.0660	0.1249	X	0.0964	0.0960
F-NNR	(i)	0.2829	0.2658	0.0577	0.0684	0.1035	0.1046	0.0625	0.0618
	(iii)	0.2829	0.2640	0.0577	0.0568	0.1035	X	0.0625	0.0618

Table 5.1: Minimum relative errors without ($\widehat{\lambda}=0$) and with ($\widehat{\lambda}\neq 0$) a hybrid approach using various regularization parameter setting techniques: (i) secant; (ii) optimal; (iii) fixed. The mark "X" means that the optimal regularization parameter found by the "Fixed (iii)" method is 10^{-16}, hence there is no need for additional regularization.

It is easy to observe that the use of additional regularization is most effective for the standard GMRES solver, where the minimum relative error is reduced significantly. However, for the other solvers, the hybrid approach does not have a notable advantage over not using regularization. At times the "Fixed (iii)" parameter choice strategy delivers a regularization parameter of the order of 10^{-16}, which is numerically equivalent to not having regularization. This indicates that our new IRN-NNR and F-NNR methods are successful in computing good reconstructions and, even

without additional regularization, they perform much better than standard Krylov methods used in a hybrid fashion (comparing IRN-GMRES-NNR to GMRES in Example 1, and FLSQR-NNR(v) to LSQR in the other examples). Figure 5.14 shows a couple of such comparisons.



hybrid GMRES IRN-GMRES-NNR hybrid LSQR FLSQR-NNR(v)

Figure 5.14: Reconstructions obtained by standard hybrid Krylov methods and by the new methods without using additional regularization. Left side: zoomed in surface plots of the reconstructions of *Example 1*; right side: reconstructions of *Example 3* (`peppers`).

Chapter 6

Extended Application: Model Calibration for Computed Tomography

In Chapter 5, we explored the effect of Krylov subspace methods, including the newly developed low-rank, nuclear norm based methods (as described in Chapter 4) on a variety of image reconstruction problems: image deblurring, computed tomography, as well as inpainting. All these applications share the property that the forward operator $\boldsymbol{A}$ is fixed, and that we are performing minimization with respect to one unknown: $\boldsymbol{x}$.

However, this is not the case for all imaging applications. For example, the advancement of x-ray science and optics technologies has enabled improved spatial resolution and increased sample clarity, but meanwhile, reconstructions of the im-

aged object have become more susceptible to random errors and systematic errors such as drifts in scanning probes, leading to distorted images with decreased resolution, or even misrepresentation of structure. As another example, in the diagnosis and management of infectious diseases such as Covid-19, where portable tomosynthesis machines are brought to the location of patients to help minimize the risk of cross-infection [9, 38], more experimental errors such as inaccurate calibration of geometry parameters (e.g., source to object distance, source orientation) are introduced, making the image acquisition and reconstruction processes more challenging. Due to these uncertainties, forward models that describe the forward process only approximate the reality to some extent, and without proper model calibration, quality of the solutions will be degraded. Using mathematical language – when working on the corresponding inverse problem, we need to take into account an additional set of unknown parameters $\boldsymbol{p}$ that the forward operator $\boldsymbol{A}$ is dependent on. So instead of (1.3), we need to solve the following problem:

$$\min_{\boldsymbol{x},\boldsymbol{p}} \|\boldsymbol{A}(\boldsymbol{p})\boldsymbol{x} - \boldsymbol{b}\|_2^2 + \lambda \mathcal{R}(\boldsymbol{x}). \tag{6.1}$$

A lot of work has been done in various applications, such as scanning position error correction for image reconstruction in the fields of electron tomography [10, 26] and ptychography [3, 37], addressing the importance of accurate system modeling. In the field of X-ray tomography, scientists at Argonne National Laboratory have developed algorithms to calibrate the center of rotation errors [2] using numerical optimization, and drifts in scanning positions [36] using targeted calibration models. In the application where uncertain view angles need to be estimated for computed

tomography in addition to image reconstruction, approaches such as uncertainty quantification have been proposed, and they seek to quantify scanning angles via a model-discrepancy formulation [58, 59]. Though very different from each other in nature, many of these methods share a similar framework called the block coordinate descent (BCD), which alternatively optimizes for the unknown parameters and the image of interest. In this Chapter, we investigate two applications related to computed tomography (CT) reconstruction: parallel and fan-beamed, where the formulation (6.1) is appropriate since model calibration is required, and BCD type methods will be investigated.

6.1 Parallel Tomography

We consider the identical parallel-beam x-ray tomography problem as put forward in [36] by Huang, Di and Wild. Scanning positions that shoot parallel x-ray beams through the object are assumed to be equally-spaced with a fixed step size Δ. These theoretical scanning positions and theoretical beams are marked by green dots and green dashed lines in Figure 6.1. Mechanical errors in the equipment can result in errors (called drifts) in the scanning positions. As illustrated in Figure 6.1, the true, drifted scanning positions are marked by purple dots, true beams by purple dashed lines, and the amount of drift for the beam τ is marked Δ_τ. A range $[-\Delta_{\max}, \Delta_{\max}]$ exists for possible values of Δ_τ. The existence of these unknown drifts in the scanning positions can cause the image reconstruction quality to be deteriorated, and the larger the amount of drift, the worse the quality. This is because these unknown

drifts cause the true forward operator to be different from the theoretical forward model that is constructed without considering drifts. Aside from drift in scanning positions, additive white Gaussian noise is also present in the sinogram.

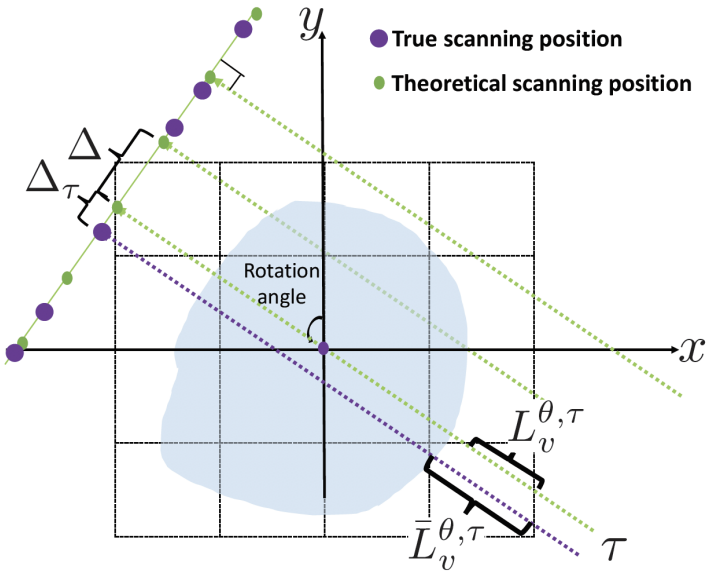


Figure 6.1: Parallel tomography illustration from [36].

The object is an image of size $n \times n$, which is vectorized and denoted by $\boldsymbol{x} \in \mathbb{R}^N$. The true forward operator $\boldsymbol{A}$ maps the object to the true sinogram $\boldsymbol{b}$ (vectorized from $\boldsymbol{B}$). Note that $\boldsymbol{A}$ and $\boldsymbol{b}$ are "true" quantities meaning that they are obtained from the experiments and are contaminated by both drift and noise. We denote the theoretical forward operator by $\boldsymbol{A}_0$ (the one that we know). The object is rotated N_θ times for scanning, each with N_τ parallel beams. Hence the sinogram is an image of size $N_\tau \times N_\theta$, and it is then vectorized into $\boldsymbol{b}$. Additive Gaussian noise in $\boldsymbol{b}$ is denoted by $\boldsymbol{\eta}$ so that $\boldsymbol{b} = \boldsymbol{b}^{\text{ex}} + \boldsymbol{\eta}$. For simplicity, it is assumed in [36] that after each rotation of the object, the drifts in the scanning positions remain the same, so the

drifts in beams don't depend on the rotation angle, and there are only N_τ unknown drifts. We denote the vector of drifts Δ_τ as $\boldsymbol{\delta}^\tau$.

To obtain the reconstruction $\boldsymbol{x}$, the following minimization problem is formulated:

$$\min_{\boldsymbol{x},\boldsymbol{\delta}^\tau} \frac{1}{2}\|\boldsymbol{A}(\boldsymbol{\delta}^\tau)\boldsymbol{x} - \boldsymbol{b}\|^2 + \lambda\mathcal{R}(\boldsymbol{x}), \tag{6.2}$$

where $\mathcal{R}(\boldsymbol{x})$ is the regularization term and $\lambda > 0$ is the regularization parameter. Due to the presence of unknown drifts, the following algorithm (Algorithm 11) is proposed in [36], which is essentially a block coordinate descent (BCD) algorithm that alternatively optimizes with respect to $\boldsymbol{x}$ and $\boldsymbol{\delta}^\tau$ using drift information obtained in each iteration.

Algorithm 11 Optimization Algorithm for Parallel CT Problem

1: Input: $\boldsymbol{b}$, $\boldsymbol{A}_0$, $\boldsymbol{\delta}_0^\tau = \boldsymbol{0}$ λ_0, $k_{\max}$
2: Solve for $\boldsymbol{x}_0 = \frac{1}{2}\arg\min_{\boldsymbol{x}} \|\boldsymbol{A}_0\boldsymbol{x} - \boldsymbol{b}\|^2 + \lambda_0\mathcal{R}(\boldsymbol{x})$
3: **for** $k = 1, \cdots, k_{\max}$ **do**
4: Estimate $\boldsymbol{\delta}_k^\tau$, construct $\boldsymbol{P}_k = \boldsymbol{P}(\boldsymbol{\delta}_k^\tau)$, and update$\boldsymbol{A}_k = \boldsymbol{P}_k\boldsymbol{A}_{k-1}$
5: Choose λ_k
6: Solve for $\boldsymbol{x}_k = \frac{1}{2}\arg\min_{\boldsymbol{x}} \|\boldsymbol{A}_k\boldsymbol{x} - \boldsymbol{b}\|^2 + \lambda_k\mathcal{R}(\boldsymbol{x})$
7: **end for**
8: Output: $\boldsymbol{x}_k$

To construct $\boldsymbol{P}_k$ in step 4 of Algorithm 11, firstly, drifts are estimated by matching the true sinogram $\boldsymbol{B}$ (i.e., observed sinogram) to the "updated" sinogram $\boldsymbol{B}^*$ obtained by reshaping $\boldsymbol{A}_{k-1}\boldsymbol{x}_{k-1}$ to matrix form. That is, for the τth beam, the estimated drift is computed by minimizing (among all possible drift values) the difference between the τth column of $\boldsymbol{B}$ and an interpolation of corresponding columns of $\boldsymbol{B}^*$. After the drift is estimated for every beam, the interpolation matrix $\boldsymbol{P}_k$ is

then computed by a closed-form formula. For more details on constructing $\boldsymbol{P}_k$ from $\boldsymbol{\delta}_k^\tau$, we refer the readers to [36].

When it comes to choosing a regularization parameter λ_k in each iteration in Step 5, the paper [36] uses a heuristic approach based on the idea that as k (the number of iterations) increases, $\boldsymbol{x}_k$ becomes a better approximation, and the problem becomes less ill-posed. Therefore, it is natural to decrease λ_k as we move forward. For this approach, a first λ_0 and last $\lambda_{k\max}$ ($\lambda_0 > \lambda_{k\max}$) need to be picked beforehand, and intermediate λ_k's are chosen so that they are equally spaced. Using MATLAB notation, the λ_k's for $k = 1, \cdots, k_{\max}$ take values in the vector

$$\texttt{linspace}(\lambda_0, \lambda_{\texttt{k}\max}, \texttt{1} + \texttt{k}\max). \tag{6.3}$$

As we attempt to come up with a more systematic way of choosing λ_k's, we are immediately reminded of the discrepancy principle based adpative approach that is used for hybrid Krylov subspace methods [21] described in Section 2.4. A similar idea can be applied here. Firstly, let $\boldsymbol{x}_{k,\lambda}$ be the solution to

$$\min_{\boldsymbol{x}} \frac{1}{2}\|\boldsymbol{A}_k\boldsymbol{x} - \boldsymbol{b}\|^2 + \lambda\Phi(\boldsymbol{x}), \tag{6.4}$$

and define $\psi_k(\lambda)$ in the same way as (2.15):

$$\psi_k(\lambda) = \|\boldsymbol{A}_k\boldsymbol{x}_{k,\lambda} - \boldsymbol{b}\|. \tag{6.5}$$

Since derivation steps are pretty much the same as in Section 2.4, we omit the

redundant steps here and present the final formula directly. Given noise level $\epsilon = \|\boldsymbol{\eta}\|_2$ and $\mu \gtrapprox 1$, λ_k can be updated as follows:

$$\lambda_k = \left| \frac{\mu\epsilon - \psi_k(0)}{\psi_k(\lambda_{k-1}) - \psi_k(0)} \right| \lambda_{k-1}. \tag{6.6}$$

For both Krylov subspace methods and Algorithm 11, $\psi_k(\lambda_{k-1})$ can be obtained relatively easily. An advantage of Krylov subspace methods is that the residual $\psi_k(0) = \|\boldsymbol{A}_k\boldsymbol{x}_{k,0} - \boldsymbol{b}\|$ can be obtained directly without solving for $\boldsymbol{x}_{k,0}$. However, for Algorithm 11, we need to solve for $\boldsymbol{x}_{k,0}$ in order to compute $\psi_k(0)$. The same solver for step 6 can be used here. And to speed it up, there are a few things we can do. For example, we could

- increase the tolerance for the solver's stopping criterion (since we only need the residual, the solution doesn't need to be very accurate), and
- take $\boldsymbol{x}_{k-1,0}$ to be the starting guess for the solver (as observed with this discrepancy-base parameter selecting approach, the value of $\psi_k(0)$ would gradually stabilize as k increases).

For step 6 in Algorithm 11, the Two-Step Iterative Shrinkage/Thresholding (TwIST) solver [5] used with TV regularization is considered in [36]. However, other linear solvers, such as Krylov subspace methods (e.g. LSQR) can also be applied in this step. In the upcoming subsection, we will investigate the performance of Algorithm 11 on the parallel tomography reconstruction problem that requires parameter calibration. We will first follow [36] to use TwIST as the linear solver, and explore the effect of the adaptive λ choosing approach (6.6) against the old approach (6.3) when

applied to the BCD algorithm. In addition, we will also observe how solutions differ if we adopt Krylov methods in the linear step.

6.1.1 Numerical Examples

Using TwIST with TV Regularization. In this part, we demonstrate the performance of Algorithm 11, in which TwIST [5] with TV regularization is used in the linear step (step 6). We perform tests for different drift levels, namely, $\Delta_{\max} = \max \Delta_\tau \in \{\Delta, 3\Delta, 5\Delta\}$. Recall that Δ_τ is drift in beam τ, and Δ is equal to the theoretical step size between adjacent x-ray beams. For each case, to further test the robustness of the proposed method, we superimpose two levels of Gaussian noise to the sinogram $\boldsymbol{b}$ (simulated with the assigned drifts) with standard deviation σ equal to 0.01 and 0.02 respectively. Note that $\boldsymbol{b}$ is first "normalized" by its largest element before noise is added, after which $\boldsymbol{b}$ is scaled back to its original scale. Two test images of size 256 by 256 are used: Brain and Phantom, and they are displayed in Figure 6.2.

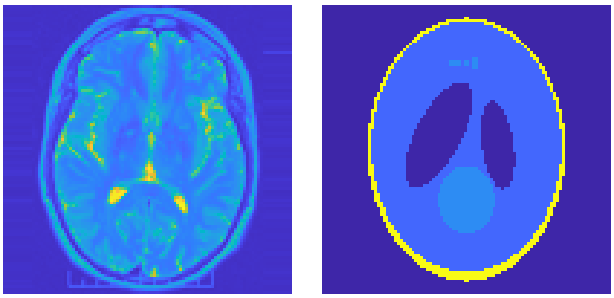


Figure 6.2: Test images Brain (left) and Phantom (right).

We adopt the test metric PSNR (peak signal to noise ratio) to be consistent with

[36]. The larger PSNR is, the better the image reconstruction. The stopping criterion for TwIST is when the number of iterations reaches 500 or when the relative change in the objective function 6.2 is less than or equal to 10^{-4}. And the stopping criterion for the outer iterations (i.e., BCD iterations) is if the change in relative residual is small, i.e.,

$$\left|\frac{1}{2}\|\boldsymbol{A}_k\boldsymbol{x}_k - \boldsymbol{b}\|_2^2 - \frac{1}{2}\|\boldsymbol{A}_{k-1}\boldsymbol{x}_{k-1} - \boldsymbol{b}\|_2^2\right| \leqslant 10^{-3}. \tag{6.7}$$

Note that in this experimental setting, in order to use the adaptive λ setting approach (6.6), we need the noise level $\epsilon = \|\boldsymbol{\eta}\|_2$, but it is not readily available. We are only given standard deviation of the noise, σ, and an estimate of the noise level ϵ can be obtained using the following formula,

$$\epsilon = \|\boldsymbol{\eta}\|_2 \approx b_{\max}\sqrt{N_\tau N_\theta}\sigma, \tag{6.8}$$

where $b_{\max}$ is the maximum element in $\boldsymbol{b}$. Since this is very likely an overestimate (because it is estimated with $b_{\max}$), in practice, we can actually take $\mu\epsilon \approx b_{\max}\sqrt{N_\tau N_\theta}\sigma$ without explicitly defining $\mu \gtrapprox 1$ when updating λ_k using (6.6). It is also worth mentioning how we set the initial regularization parameter λ_0. In order to have the initial λ_0 reflect well the level of noise from two main sources, the white noise and experimental drifts, in our problem, we propose the following formula for constructing λ_0:

$$\lambda_0 = 0.001(1+\sigma)(1+\Delta_{\max}). \tag{6.9}$$

Table 6.1 displays the PSNR values (in dB) for a variety of test cases with different objects, different maximum drift levels and different noise levels. We compare results

of the baseline approach (i.e., solving the linear system directly without calibrating drift) and the heuristic approach (i.e., Algorithm 11 using the heuristic λ choosing method (6.3) against the adaptive regularization parameter setting approach (6.6). The presented PSNR values for the adaptive approach are given by the convergent iterations for each test case. For the heuristic approach, we take the 10th iteration, at which λ_0 is decreased to $0.01\lambda_0$.

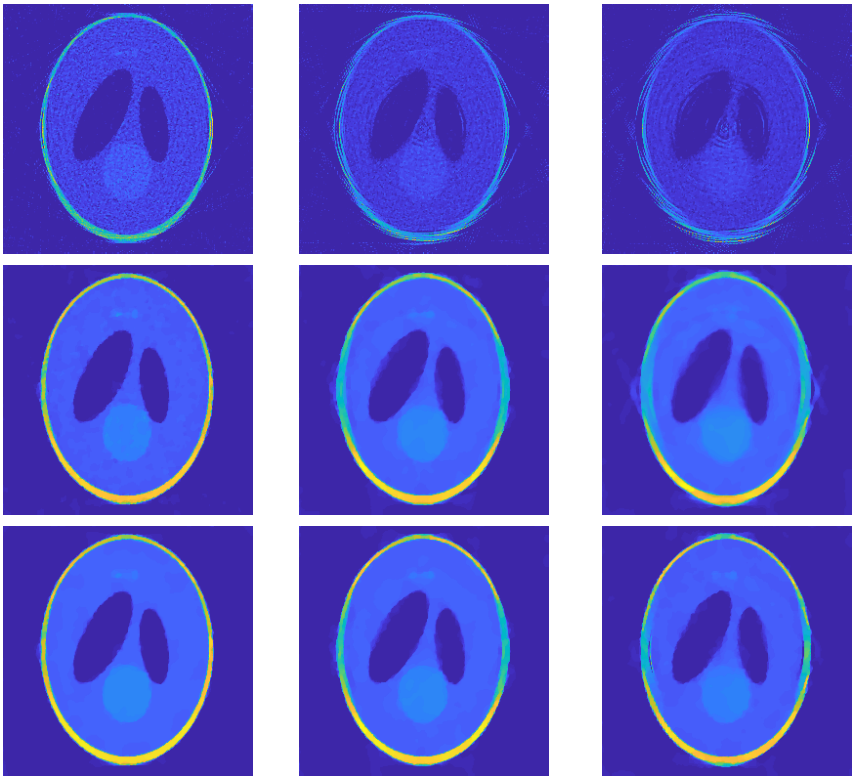
We can observe in Table 6.1 that in general, the adaptive approach outperforms both heuristic approach and baseline approach. However, for a few cases with noise $\sigma = 0.02$, the PSNR given by "adaptive" is slightly less than the "heuristic" approach's PSNR value. This is probably because the estimate for ϵ is less accurate for larger values of σ.

Max Drift	Δ			3Δ			5Δ		
Noise σ	0	0.01	0.02	0	0.01	0.02	0	0.01	0.02
Brain (adap.)	29.43	26.78	24.61	27.01	25.54	23.53	25.46	24.90	23.30
Brain (heur.)	26.99	26.34	25.50	24.44	24.38	23.87	23.57	23.51	23.34
Brain (base.)	26.27	24.34	19.90	20.73	20.22	18.53	18.06	17.80	17.08
Phan (adap.)	26.77	26.05	24.44	22.80	22.59	21.96	21.75	21.54	21.67
Phan (heur.)	24.94	24.98	24.82	21.19	20.90	20.89	19.75	19.86	19.66
Phan (base.)	21.90	21.51	20.53	16.58	16.53	16.23	14.26	14.22	14.04

Table 6.1: Performance comparison of new adaptive (adap.) method with the baseline (base.) and heuristic (heur.) methods for three drift levels and three noise levels on Brain and Phantom (Phan) test images. Test Metric: PSNR (dB).

We visualize the reconstructions of phantom in Figure 6.3. It is noticeable that with drift calibration, we are able to recover the fine feature of the object, and in particular, with the adaptive λ_k approach, the fine details can be recovered to an

enhanced contrast comparing to the old approach in average.



$\Delta_{\max} = \Delta,\ \sigma = 0.02$ $\Delta_{\max} = 3\Delta,\ \sigma = 0.02$ $\Delta_{\max} = 5\Delta,\ \sigma = 0.02$

Figure 6.3: Phantom reconstructions given by Algorithm 11 using TwIST. Top row: baseline; middle row: heuristic λ_k's; bottom row: adaptive λ_k's.

We further investigate the iterative performance of the two approaches choosing λ_k's in Figure 6.4. Remarkably, our proposed adaptive way is able to gradually decrease λ until stabilized. Comparing to the heuristic approach, the proposed formula promotes the convergence of the performance in a more automatic way. To perform fair visual comparison, we allow the algorithm to continue until reaching the maximum allowed number of iterations. We indicate the convergence of our proposed adaptive approach on λ_k by a red diamond according to stopping criteria 6.7.

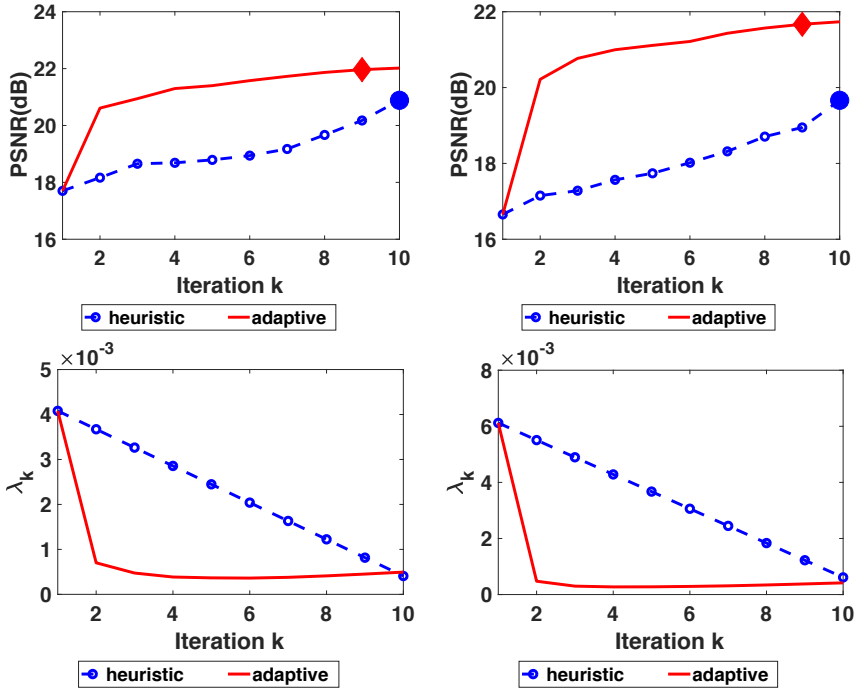
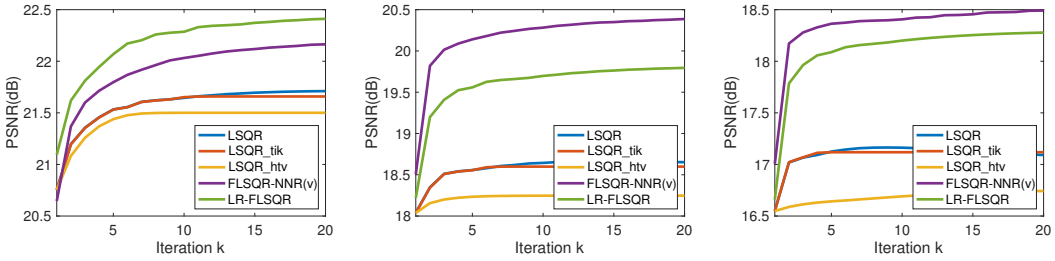


Figure 6.4: Convergence results for Algorithm 11 on Phantom test image. Top row: PSNR; bottom row: λ_k's

Using Krylov Subspace Methods. Since the main contribution of this thesis is the newly developed low-rank Krylov methods, we also put the new methods to test in this parallel tomography calibration problem. Note that we are applying these Krylov methods in step 6 of Algorithm 11, replacing TwIST. Krylov subspace methods are intrinsically different from the previous TwIST solver, so in this part, we compare the performance of the new low-rank methods to other Krylov methods in its family. Namely, we compare the following methods: LSQR (without any form of regularization), LSQR_tik (LSQR with Tikhonov regularization), LSQR_htv (LSQR with heuristic TV regularization), as well as the two low rank methods: FLSQR-

NNR(v) (Section 4.4) and LR-FLSQR (Section 5.1.1). Implementations of LSQR_tik and LSQR_htv can be found in IR Tools [17].

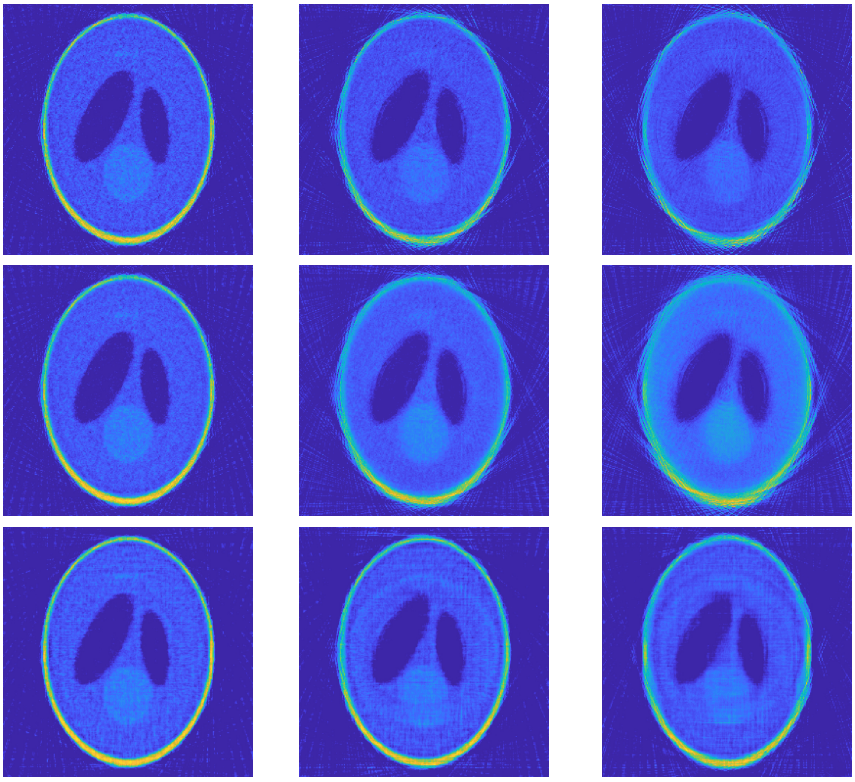
We test the Krylov solvers using the Phantom test image as seen in Figure 6.2. We fix the noise standard deviation σ to be 0.02, and vary the maximum drift level so that $\Delta_{\max} = \max \Delta_\tau \in \{\Delta, 3\Delta, 5\Delta\}$. We set the number of outer iterations to be 20. In terms of inner iterations (i.e., Krylov iterations), we also use 20, but rather than imposing a stopping criterion, we pick the iteration whose solution has the lowest relative error compared against the true solution. We do so because we are interested in comparing the performance of Krylov methods in the best case scenario (i.e., stopping at lowest relative error). Also, the methods LSQR_tik and LSQR_htv require regularization parameters, and we follow (6.6) to adaptively pick a fixed λ_k for each outer iteration. This is different from the automatic update of regularization parameters in the inner Krylov iterations.



$\Delta_{\max} = \Delta$, $\sigma = 0.02$ $\Delta_{\max} = 3\Delta$, $\sigma = 0.02$ $\Delta_{\max} = 5\Delta$, $\sigma = 0.02$

Figure 6.5: PSNR convergence plots (Phantom) using Krylov methods. For LR-FLSQR, truncation ranks $\kappa = \kappa_B = 50$, picked by trial and error.

Figure 6.5 displays the convergence of PSNR values of the various Krylov methods. We can easily observe that out of the 5 Krylov methods, LSQR, LSQR_tik and LSQR_htv behave quite similarly for different drift levels, and all three of these



$\Delta_{\max} = \Delta,\ \sigma = 0.02$ $\Delta_{\max} = 3\Delta,\ \sigma = 0.02$ $\Delta_{\max} = 5\Delta,\ \sigma = 0.02$

Figure 6.6: Phantom reconstructions using Krylov methods. Upper row: LSQR solutions (no regularization); middle row: LSQR_htv solutions; bottom row: FLSQR-NNR(v) solutions.

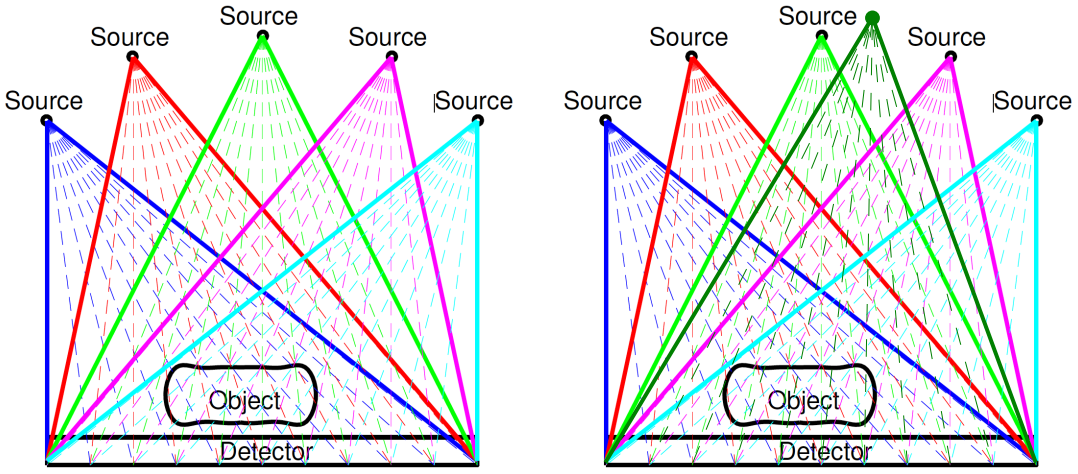
methods are outperformed by the new low-rank Krylov methods FLSQR-NNR(v) and LR-FLSQR. It is also interesting that for the low drift level $\Delta_{\max} = \Delta$, the best-performing method is LR-FLSQR, while for the higher drift levels $\Delta_{\max} = 3\Delta$ and 5Δ, FLSQR-NNR(v) is the best of all. Furthermore, the improvement in the new low rank methods over traditional Krylov methods is more apparent for the larger drift levels. In spite of the improvements, we should also recognize that even when we are exploiting the Krylov methods to their fullest potential, all Krylov solutions

are still of lower quality compared to TwIST. The best-performing Krylov methods give PSNR values that are 1.5-3.5 lower than TwIST.

In Figure 6.6, we present solutions at the 20th outer iterations of Algorithm 11 using LSQR, LSQR_htv, and FLSQR-NNR(v). It is easy to see that FLSQR-NNR(v) renders solutions of the highest clarity compared to the other two – there is less noise in the background, and the phantom shape is better preserved. However, when compared to TwIST reconstructions in Figure 6.3, FLSQR-NNR(v) is still lacking in details (e.g., the three small circles inside the phantom), and are more noisy.

6.2 Fan-Beam Tomography

In this section, we describe a different tomography setup – fan-beam tomography. That is, instead of a panel that emits parallel x-ray beams, we now have a point source that emits fan beams, as can be seen in Figure 6.7(a). Ideally, for each scan, the source should be of equal distance to the object and rotates a fixed angle. However, this may not be the case for portable CT machines that are subject to more experimental errors caused by machine transportation. In Figure 6.7(b), the theoretical location of the center scanning position is shown in light green, but during the scanning process it may be shifted to the dark green location. Hence, when reconstructing the imaged object, we also need to take into consideration geometry parameter calibration.



(a). (b).

Figure 6.7: Fan-beam Tomography illustration.

Consider the fan-beam CT image reconstruction problem

$$\min_{\boldsymbol{x},\boldsymbol{p}} \|\boldsymbol{A}(\boldsymbol{p})\boldsymbol{x} - \boldsymbol{b}\|_2^2 + \lambda\mathcal{R}(\boldsymbol{x}), \tag{6.10}$$

where $\boldsymbol{A} \in \mathbb{R}^{M\times N}$ is the forward model that depends on geometry parameters $\boldsymbol{p}$, $\boldsymbol{b} \in \mathbb{R}^M$ is the observed data (sinogram) perturbed by random noise $\boldsymbol{\eta}$ (i.e., $\boldsymbol{b} = \boldsymbol{b}^{\text{ex}} + \boldsymbol{\eta}$), $\boldsymbol{x} \in \mathbb{R}^N$ is the quantity of interest (imaged object), and $\lambda\mathcal{R}(\boldsymbol{x})$ is a regularization term. In experimental settings, the geometry parameters $\boldsymbol{p}$ may not be exactly known, but we have an estimate, i.e., their theoretical values. As a result, we need to both solve for the solution image $\boldsymbol{x}$ and calibrate parameters $\boldsymbol{p}$. Note also that $\boldsymbol{b}$ and $\boldsymbol{x}$ are vectorized quantities of the sinogram $\boldsymbol{B} \in \mathbb{R}^{N_\tau\times N_\theta}$ (where N_τ is the number of beams and N_θ is the number of scanning angles) and the solution image $\boldsymbol{X} \in \mathbb{R}^{n\times n}$, where $N = n^2$.

For this problem, we consider two types of uncertainties in the geometry parameters: source to object distances, and scanning angles. Hence the unknown geometry parameters $\boldsymbol{p} \in \mathbb{R}^{2N_p}$ consist of two components $\boldsymbol{r}$ and $\boldsymbol{d}$, i.e., $\boldsymbol{p} = [\boldsymbol{r};\ \boldsymbol{d}]$, where $\boldsymbol{r} \in \mathbb{R}^{N_p} = [r_1; \cdots ; r_{N_p}]$ are source to object distances with theoretical value $2n$, and $\boldsymbol{d} \in \mathbb{R}^{N_p} = [\delta_1^\theta; \cdots ; \delta_{N_p}^\theta]$ are perturbation in scanning angles with theoretical value 0. We assume that $r_i \in [1.5n, 2.5n]$ (n is dimension of image) and $\delta_i^\theta \in [-0.5, 0.5]$.

Note that technically, N_p could be equal to the total number of scanning angles and can be as large as 180 or 360. For simplification purposes, we start by choosing N_p much smaller than total number of scanning angles (e.g., $N_p = 3$ or $N_p = 6$). For instance, if we use $N_\theta = 180$ scanning angles 0:2:359 and $N_p = 3$, then $\boldsymbol{p} \in \mathbb{R}^6$, $\boldsymbol{r},\ \boldsymbol{d} \in \mathbb{R}^3$, and (r_i, δ_i^θ) would be constant for a set of $180/N_p = 60$ angles.

Similar to what's described in Section 6.1, a block coordinate descent (BCD) optimization algorithm may be used to alternatively solve for $\boldsymbol{x}$ and $\boldsymbol{p}$ in Problem (6.10). The idea is simple: for the unknowns $\boldsymbol{x}$ and $\boldsymbol{p}$, we can fix one and solve for the other, and repeat this process multiple times. Since we are given the theoretical value of $\boldsymbol{p}$, we can use it as an initial guess $\boldsymbol{p}_0$, and start from here to solve for $\boldsymbol{x}_0$, then $\boldsymbol{p}_1,\ \boldsymbol{x}_1,\ \boldsymbol{p}_2,\ \boldsymbol{x}_2, \cdots$, etc. The BCD method is summarized in Algorithm 12. Note that our version of BCD (Algorithm 12) is sometimes also called alternating minimization.

The linear steps 2 and 6 of Algorithm 12 can be solved with a linear inverse problem solver such as LSQR. But for the model update step (step 4), since $\boldsymbol{A}$ depends nonlinearly on $\boldsymbol{p}$, we need to use a nonlinear solver such as Gauss-Newton based `imfil` [42], and this nonlinear step can be expensive for large values of N_p.

Algorithm 12 BCD for fan-beam CT problem

1: **Input:** $\boldsymbol{b}$, $\boldsymbol{p}_0$, $\boldsymbol{A}_0 = \boldsymbol{A}(\boldsymbol{p}_0)$
2: Solve for $\boldsymbol{x}_0 = \arg\min_{\boldsymbol{x}} \|\boldsymbol{A}_0\boldsymbol{x} - \boldsymbol{b}\|^2 + \lambda_0\|\boldsymbol{x}\|_2^2$
3: **for** $k = 1, \cdots$ **do**
4: Solve for $\boldsymbol{p}_k = \arg\min_{\boldsymbol{p}} \|\boldsymbol{A}(\boldsymbol{p})\boldsymbol{x}_k - \boldsymbol{b}\|^2$
5: Update $\boldsymbol{A}_k = \boldsymbol{A}(\boldsymbol{p}_k)$
6: Solve for $\boldsymbol{x}_k = \arg\min_{\boldsymbol{x}} \|\boldsymbol{A}_k\boldsymbol{x} - \boldsymbol{b}\|^2 + \lambda_k\|\boldsymbol{x}\|_2^2$
7: **end for**
8: **return** $\boldsymbol{x}_k$

Note that for this step, we can actually reduce problem size by exploiting separability of $\boldsymbol{A}(\boldsymbol{p})$. That is, the minimization problem in step 4 of Algorithm 12 can be equivalently written as

$$\min_{\boldsymbol{p}} \left\| \begin{bmatrix} \boldsymbol{A}(p_1) \\ \boldsymbol{A}(p_2) \\ \vdots \\ \boldsymbol{A}(p_{N_p}) \end{bmatrix} \boldsymbol{x} - \begin{bmatrix} \boldsymbol{b}_1 \\ \boldsymbol{b}_2 \\ \vdots \\ \boldsymbol{b}_{N_p} \end{bmatrix} \right\|^2 = \sum_{i=1}^{N_p} \|\boldsymbol{A}(p_i)\boldsymbol{x} - \boldsymbol{b}_i\|^2. \tag{6.11}$$

By doing so, we divide the big nonlinear problem into N_p small problems, which can be solved in parallel. The separability property works very well with our proposed BCD framework, but other approaches such as variable projection [53, 55] would not be able to utilize this nice property. On the other hand, the nonlinear solver can depend heavily on the initial guess, and may converge to a local minimum rather than the global minimum, which affects all subsequent $\boldsymbol{x}_k$'s and $\boldsymbol{p}_k$'s. Hence, although we can conveniently define $\boldsymbol{p}_0$ as the theoretical value for $\boldsymbol{p}$, it might be worthwhile to look for other ways of defining $\boldsymbol{p}_0$ that allows for better convergence of the overall

BCD solver.

Since the nonlinear optimization step is very costly and takes a long time to run, it would be even better if we can find a way to avoid running this step. Inspired by [1], which trains neural networks to learn regularization parameters for inverse problems, we are interested in building machine learning models to learn the unknown CT geometry parameters. In other words, by training a machine learning model $\widehat{\boldsymbol{\Phi}} : \boldsymbol{b} \rightarrow \boldsymbol{p}$ that maps sinogram $\boldsymbol{b}$ to geometry parameters $\boldsymbol{p}$, we may be able to avoid running the alternating minimization scheme. And as a result, we only need to run one linear step to solve for $\boldsymbol{x}$. This framework is summarized in Algorithm 13.

Algorithm 13 Learning Geometry Parameters

1: *offline phase*
2: For $j = 1, \cdots, J$, randomly generate geometry parameters $\boldsymbol{p}_j$, noise $\boldsymbol{\eta}_j$, and generate training data $\boldsymbol{b}_j = \boldsymbol{A}(\boldsymbol{p}_j)\boldsymbol{x}_j^{\text{ex}} + \boldsymbol{\eta}_j$
3: Train model $\widehat{\boldsymbol{\Phi}}$ which maps observation $\boldsymbol{b}$ onto geometry parameters $\boldsymbol{p}$
4: *online phase*
5: For new data $\boldsymbol{b}_{j'}$, predict parameters $\boldsymbol{p}_{j'} = \widehat{\boldsymbol{\Phi}}(\boldsymbol{b}_{j'})$
6: Solve inverse problem $\min_{\boldsymbol{x}} \|\boldsymbol{A}(\boldsymbol{p}_{j'})\boldsymbol{x} - \boldsymbol{b}\|^2 + \lambda\mathcal{R}(\boldsymbol{x})$

To train the model $\widehat{\boldsymbol{\Phi}} : \boldsymbol{b} \rightarrow \boldsymbol{p}$, we need the following training data:

- $\widehat{\boldsymbol{B}} \in \mathbb{R}^{J\times M}$ is (feature) matrix with J rows. Each row is a vectorized sinogram $\boldsymbol{b}_j$ of length M, constructed using random $\boldsymbol{r}_j$ and $\boldsymbol{\delta}_j^{\theta}$, with superimposed Gaussian noise, i.e., $\boldsymbol{b}_j = \boldsymbol{A}(\boldsymbol{r}_j, \boldsymbol{\delta}_j^{\theta})\boldsymbol{x}_j^{\text{ex}} + \boldsymbol{\eta}_j$, with $\boldsymbol{x}_j^{\text{ex}}$ the true phantom and $\boldsymbol{\eta}_j$ the random noise vector.

- $\boldsymbol{R}, \boldsymbol{D} \in \mathbb{R}^{J\times N_p}$ are matrices of true geometry parameters, with rows $\boldsymbol{r}_j$ and $\boldsymbol{\delta}_j^{\theta}$ corresponding to sinogram $\boldsymbol{b}_j$.

One important consideration during the training phase is choosing the phantom images $\boldsymbol{x}_j^{\text{ex}}$. If we fix $\boldsymbol{x}_j^{\text{ex}}$ to be a constant phantom image for all j, then we would expect that the trained model only works well for the phantom that the model is trained on. In other words, the model would be best at predicting CT geometry parameters when the phantom being reconstructed is the same phantom that the training data is built upon, and may not generalize well to other phantoms. Since in a real-world CT reconstruction problem, we do not know what the phantom is, using a model that is trained upon a fixed phantom may not render good solutions.

However, if we use different phantoms $\boldsymbol{x}_j^{\text{ex}}$'s when generating the training data, the model may become confused due to the problem of non-uniqueness. It is possible that there exist $(\boldsymbol{p}_1, \boldsymbol{x}_1)$, $(\boldsymbol{p}_2, \boldsymbol{x}_2)$ with $\boldsymbol{p}_1 \neq \boldsymbol{p}_2$ and $\boldsymbol{x}_1 \neq \boldsymbol{x}_2$, such that $\boldsymbol{A}(\boldsymbol{p}_1)\boldsymbol{x}_1 = \boldsymbol{A}(\boldsymbol{p}_2)\boldsymbol{x}_2 = \boldsymbol{b}$. This means that the mapping $\widehat{\boldsymbol{\Phi}} : \boldsymbol{b} \to \boldsymbol{p}$ may not be one-to-one. Therefore, the model may predict parameters $\boldsymbol{p}$ that are very far from the true value, thus rendering far-from-true solution $\boldsymbol{x}$ in Algorithm 13.

Disturbance-based strategies have been used in applications like Power Systems [35, 71] to help with non-uniqueness. If we translate this strategy into our context, we may apply "disturbances", i.e., different phantoms and different noise levels to generate many training samples that correspond to the same set of geometry parameters $\boldsymbol{p}$. That is, for each $\boldsymbol{p}$, we generate l training samples $\boldsymbol{b}_i = \boldsymbol{A}(\boldsymbol{p})\boldsymbol{x}_i + \boldsymbol{\eta}_i$ for $i = 1, \cdots, l$ using l different phantoms and noise levels. We will see how the disturbance-based strategy performs in Section 6.2.2.

6.2.1 Hybrid ML-BCD Method

In this Section, we describe a hybrid method that combines machine learning with block coordinate descent for solving (6.10). We will also explain technical details for training the machine learning model.

We have mentioned previously that the machine learning problem of learning geometry parameters when the phantom $\boldsymbol{x}_j^{\text{ex}}$ is not fixed in the training data may suffer from non-uniqueness. Although in our experience, using Algorithm 13 with an ML model trained using non-fixed $\boldsymbol{x}_j^{\text{ex}}$'s does not deliver solutions that are too far from the true images, we do notice that relative errors in the predicted parameters on the testing set have increased compared to the case with a fixed $\boldsymbol{x}_j^{\text{ex}}$, but the relative errors are still very low compared to the theoretical values (more on this in Section 6.2.2).

One natural approach that comes to mind is using a hybrid method that combines a machine learning model and BCD. That is, we may first use the trained ML model to predict $\boldsymbol{p}$, then feed the predicted $\boldsymbol{p}$ as a starting guess to the BCD algorithm to run a few iterations to refine the solution. This hybrid framework is described in Algorithm 14.

Algorithm 14 ML-BCD hybrid framework

offline phase
  Use offline phase in Algorithm 13 to train ML model $\widehat{\boldsymbol{\Phi}} : \boldsymbol{b} \to \boldsymbol{p}$
online phase
  (predict) For new data $\boldsymbol{b}_{j'}$, predict parameters $\boldsymbol{p}_{j'} = \widehat{\boldsymbol{\Phi}}(\boldsymbol{b}_{j'})$
  (refine) Run Algorithm 12 with $\boldsymbol{p}_0 = \boldsymbol{p}_{j'}$ and $\boldsymbol{A}_0 = \boldsymbol{A}(\boldsymbol{p}_{j'})$

While there are many machine learning models to choose from, we consider one

of the simplest models – a multi-output linear regression model, which can also be thought of as a one-layer neural network. This ML model assumes a linear relationship between input and output, and for simplicity, we take a zero bias term. To build this model, we learn weights $\boldsymbol{W} \in \mathbb{R}^{M \times 2N_p}$ by solving the least squares problem

$$\min_{\boldsymbol{W}} \|\widehat{\boldsymbol{B}}\boldsymbol{W} - [\boldsymbol{R}\ \boldsymbol{D}]\|_F^2. \tag{6.12}$$

Then, given new data $\boldsymbol{b}_{j'}$, we simply need to compute $\boldsymbol{p}_{j'} = [\boldsymbol{r}_{j'}; \boldsymbol{\delta}_{j'}^{\theta}] = \boldsymbol{b}_{j'}^T \boldsymbol{W}$ to obtain the predicted geometry parameters, which can be fed as initial guess $\boldsymbol{p}_0$ into the BCD algorithm. We have chosen the loss function to be $\|\cdot\|_F^2$, which makes the minimization problem separable (similar to the nonlinear step of BCD). Since $\boldsymbol{R} = [\boldsymbol{R}_1 \cdots \boldsymbol{R}_{N_p}]$ with column vectors $\boldsymbol{R}_i$, and $\boldsymbol{D} = [\boldsymbol{D}_1 \cdots \boldsymbol{D}_{N_p}]$ with column vectors $\boldsymbol{D}_i$, (6.12) is equivalent to

$$\sum_{i=1}^{N_p} \min_{\boldsymbol{W}_{r,i} \in \mathbb{R}^M} \|\widehat{\boldsymbol{B}}\boldsymbol{W}_{r,i} - \boldsymbol{R}_i\|_2^2 + \sum_{i=1}^{N_p} \min_{\boldsymbol{W}_{d,i} \in \mathbb{R}^M} \|\widehat{\boldsymbol{B}}\boldsymbol{W}_{d,i} - \boldsymbol{D}_i\|_2^2, \tag{6.13}$$

where $\boldsymbol{W}_{r,i}$ and $\boldsymbol{W}_{d,i}$ are column vectors of $\boldsymbol{W}_R$ and $\boldsymbol{W}_D$ such that $\boldsymbol{W}_R = [\boldsymbol{W}_{r,1} \cdots \boldsymbol{W}_{r,N_p}]$, $\boldsymbol{W}_D = [\boldsymbol{W}_{d,1} \cdots \boldsymbol{W}_{d,N_p}]$, and $\boldsymbol{W} = [\boldsymbol{W}_R\ \boldsymbol{W}_D]$. The sub-problems $\min_{\boldsymbol{W}_{r,i}} \|\widehat{\boldsymbol{B}}\boldsymbol{W}_{r,i} - \boldsymbol{R}_i\|_2^2$ and $\min_{\boldsymbol{W}_{d,i}} \|\widehat{\boldsymbol{B}}\boldsymbol{W}_{d,i} - \boldsymbol{D}_i\|_2^2$ are large and non-square linear systems, and we need to use an iterative solver such as LSQR to solve each of them. Algorithm 15 contains more detailed description of the hybrid method.

Algorithm 15 ML-BCD hybrid algorithm

offline phase - training ML model
  Generate training data $\widehat{\boldsymbol{B}}$, $\boldsymbol{R}$ and $\boldsymbol{D}$
  Compute ML model weights $\boldsymbol{W}$ that is the solution to $\min_{\boldsymbol{W}} \|\widehat{\boldsymbol{B}}\boldsymbol{W} - [\boldsymbol{R}\ \boldsymbol{D}]\|_F^2$

online phase - prediction and BCD
  For new data $\boldsymbol{b}_{j'}$, predict geometry parameters $\boldsymbol{p}_{j'} = [\boldsymbol{r}_{j'}; \boldsymbol{\delta}_{j'}^{\theta}] = \boldsymbol{b}_{j'}^T \boldsymbol{W}$
  Set $\boldsymbol{p}_0 = \boldsymbol{p}_{j'}$ and $\boldsymbol{A}_0 = \boldsymbol{A}(\boldsymbol{p}_{j'})$, and solve for $\boldsymbol{x}_0 = \arg\min_{\boldsymbol{x}} \|\boldsymbol{A}_0\boldsymbol{x} - \boldsymbol{b}\|^2 + \lambda_0 \|\boldsymbol{x}\|_2^2$

  Alternatively optimize for $\boldsymbol{p}_k$ and $\boldsymbol{x}_k$ for $k = 1, 2, \cdots$ as in Algorithm 12.
  Return $\boldsymbol{x}_k$

6.2.2 Numerical Examples

In this Section, we present several numerical experiments comparing performances of Algorithms 12, 13 and 15. We have used the `PRtomo` function of IRTools [17] to generate all training data in this section. The linear solver used is `IRhybrid_lsqr` from IRTools and the nonlinear solver is `imfil` [42]. While there are many linear solvers to choose from, the purpose of the numerical experiments is to compare different algorithms that simultaneously reconstructs the image and calibrates geometry parameters, not comparing performances of different linear solvers, so we will have a fair comparison as along as we fix the linear solver.

Test 1: 64 by 64 fixed phantom problem with $N_p = 3$. We first consider a 64×64 fixed Shepp-Logan phantom as $\boldsymbol{x}_j^{\text{ex}}\ \forall j$ when generating the training data. We use 180 scanning angles, i.e., 0:2:359 and $N_p = 3$, so there are 6 unknown geometry parameters in total: 3 r_i's and 3 δ_i^{θ}'s, with each pair of (r_i, δ_i^{θ}) corresponding to 60 adjacent scanning angles. As a result, each $\boldsymbol{b}_j \in \mathbb{R}^{16380}$, so $\widehat{\boldsymbol{B}} \in \mathbb{R}^{J \times 16380}$, $\boldsymbol{R}$ and

$\boldsymbol{D} \in \mathbb{R}^{16380\times 3}$. Random white noise with fixed noise level 0.01 is superimposed onto each $\boldsymbol{b}_j$.

We have generated training data $\widehat{\boldsymbol{B}}$, $\boldsymbol{R}$ and $\boldsymbol{D}$ with $J = 20000$ training samples, and an additional testing set $\widehat{\boldsymbol{B}}_{\text{test}}$, $\boldsymbol{R}_{\text{test}}$, and $\boldsymbol{D}_{\text{test}}$ of 2000 samples. We train ML model weights using the method described in Section 6.2.1, i.e., computing weight matrix $\boldsymbol{W} = [\boldsymbol{W}_R \ \boldsymbol{W}_D]$ by solving every subproblem of (6.13). We train weights $\boldsymbol{W}$ using training sets of different sizes: 2000, 4000, 6000, $\cdots$, 20000, compute the predicted geometry parameters

$$\boldsymbol{R}_{\text{pred}} = \widehat{\boldsymbol{B}}_{\text{test}}\boldsymbol{W}_R \quad \text{and} \quad \boldsymbol{D}_{\text{pred}} = \widehat{\boldsymbol{B}}_{\text{test}}\boldsymbol{W}_D,$$

and evaluate testing errors

$$\frac{\|\boldsymbol{R}_{\text{pred}} - \boldsymbol{R}_{\text{test}}\|_F}{\|\boldsymbol{R}_{\text{test}}\|_F} \quad \text{and} \quad \frac{\|\boldsymbol{D}_{\text{pred}} - \boldsymbol{D}_{\text{test}}\|_F}{\|\boldsymbol{D}_{\text{test}}\|_F}$$

on the testing set of 2000 samples. We plot testing errors against training set size in Figure 6.8.

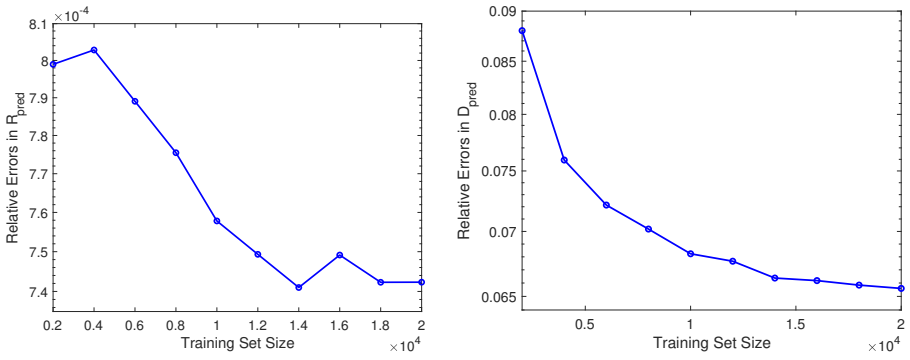
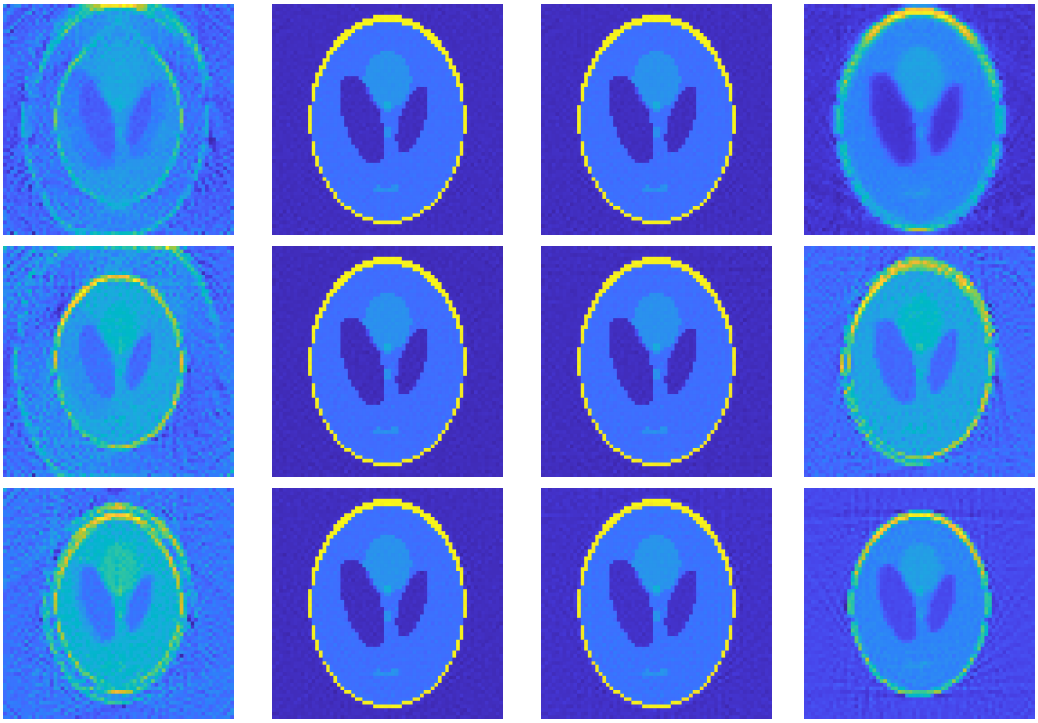


Figure 6.8: Relative errors in $\boldsymbol{R}_{\text{pred}}$ (left) and $\boldsymbol{D}_{\text{pred}}$ (right) on testing set.

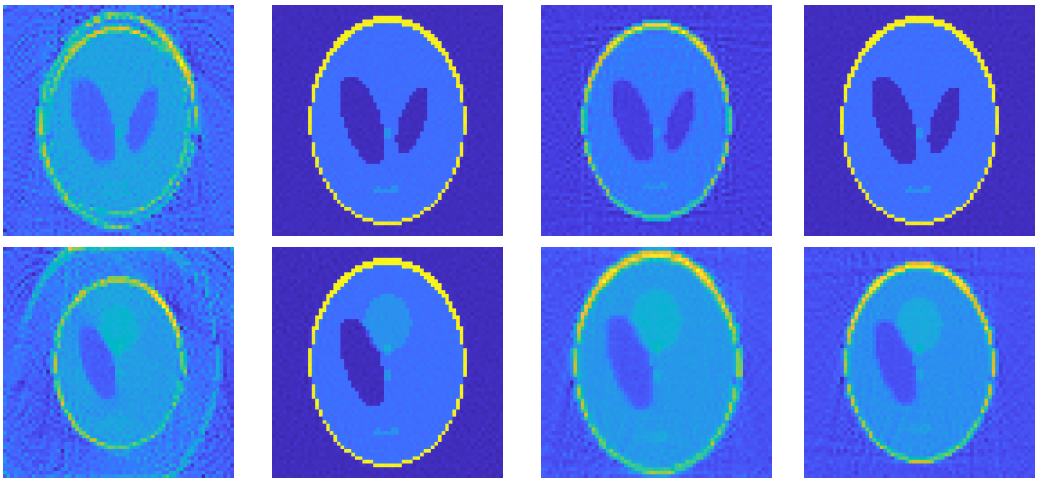
We can observe that as the training set size increases, relative error in the predicted geometry parameters decreases. This is because as the training set becomes larger, it covers more variations in the different combinations of geometry parameters, which makes predictions more accurate. Also, the relative errors in $\boldsymbol{R}_{\text{pred}}$ and $\boldsymbol{D}_{\text{pred}}$ are very different in magnitude, meaning that it is much easier to predict $\boldsymbol{r}$ parameters more accurately.



Uncalibrated. Perfect calibration. ML solution. BCD solution.

Figure 6.9: Phantom reconstructions of 3 testing samples that share the same underlying phantom with training set (1 testing sample per row). 1st column: solutions using theoretical parameters (i.e., uncalibrated). 2nd column: solutions using true parameters (i.e. perfect calibration). 3rd column: solutions using ML-predicted parameters (Algorithm 13). 4th column: solutions using 10 BCD iterations (Algorithm 12).

Figure 6.9 shows solutions using theoretical parameters, true parameters, ML-predicted parameters (Algorithm 13), and BCD (Algorithm 12). The ML model weights are trained on a training set of size 20000. We can observe in Figure 6.9 that if we leave the geometry parameters uncalibrated, the solution images will have very poor quality. If we use the BCD scheme (Algorithm 12), the reconstruction quality seems to have improved, but the solutions are still fuzzy and unclear. Using Algorithm 13 to predict geometry parameters using learned ML model weights, we obtain highly accurate solutions, resembling solutions computed using the true parameters (which represents to the case of perfect calibration).



Uncalibrated. Perfect calibration. ML solution. BCD solution.

Figure 6.10: Phantom reconstructions of 2 testing samples that have different underlying phantom than training set (1 testing sample per row). Column order same as Figure 6.9.

The model weights trained in this test is based on training data generated with the same phantom. Therefore, if the solution image we are looking for is different from the phantom that the weights are trained on, then the ML model may not gen-

eralize well. As we observe in Figure 6.10, the test samples have different underlying phantoms (different from what's in the training data). As a result, Algorithm 13 no longer yields accurate solutions, and no longer has an advantage over BCD.

Test 2: Testing Disturbances. While we want to train models that generalize well to different phantoms, the problem of non-uniqueness may affect the training of machine learning models. In this test, we explore the effect of disturbances by generating the training data using different phantoms $\boldsymbol{x}_j^{\text{ex}}$'s. Examples of such phantoms used are shown in Figure 6.11. It can be seen that each phantom is created by altering the original phantom by randomly removing ellipses and slightly varying ellipse sizes and angles. Random noise of noise level chosen randomly in the range 0.1% to 2% is superimposed to each sample.

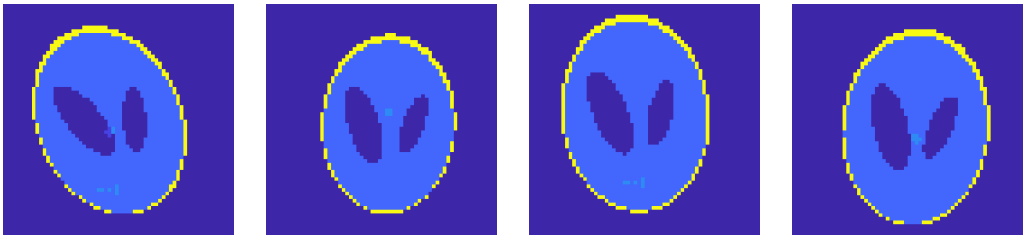


Figure 6.11: Examples of different phantoms used.

Two training sets are created. We apply the disturbance strategy in the first training set by generating 5 different phantoms and noise levels for each $\boldsymbol{p}_j$, with 7000 $\boldsymbol{p}_j$'s in total, i.e., there are 35000 samples in the data set. As a frame of reference, we compare results given by another training set of the same size, but without disturbances. That is, 35000 samples each with a different $\boldsymbol{p}_j$, phantom and noise level.

We use these two training sets and subsets of the data sets to train ML model weights. We compare the relative errors of the ML-predicted $\boldsymbol{p}$ for the testing set consisting of 1000 samples (generated with different samples in the same way as illustrated in Figure 6.11). In Figure 6.12, we show convergence of relative errors in the geometry parameters $\boldsymbol{R}_{\text{pred}}$ and $\boldsymbol{D}_{\text{pred}}$ predicted by ML models trained using 5000:5000:35000 samples of the two data sets. Note that, for example, with the training set size 15000, one training set consists of 3000 $\boldsymbol{p}_j$'s each having 5 disturbances (i.e., phantoms and noise levels), the other training set has 15000 $\boldsymbol{p}_j$'s (each having their own phantom and noise level). We also notice that when the model is trained on training data generated using different phantoms, the relative errors in the geometry parameters are around 10 times larger than the relative errors presented in Test 1 due to the variance of phantoms.

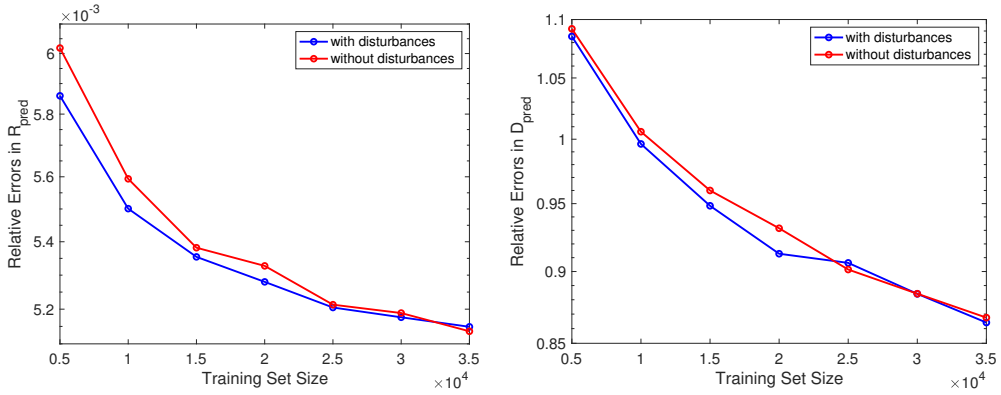


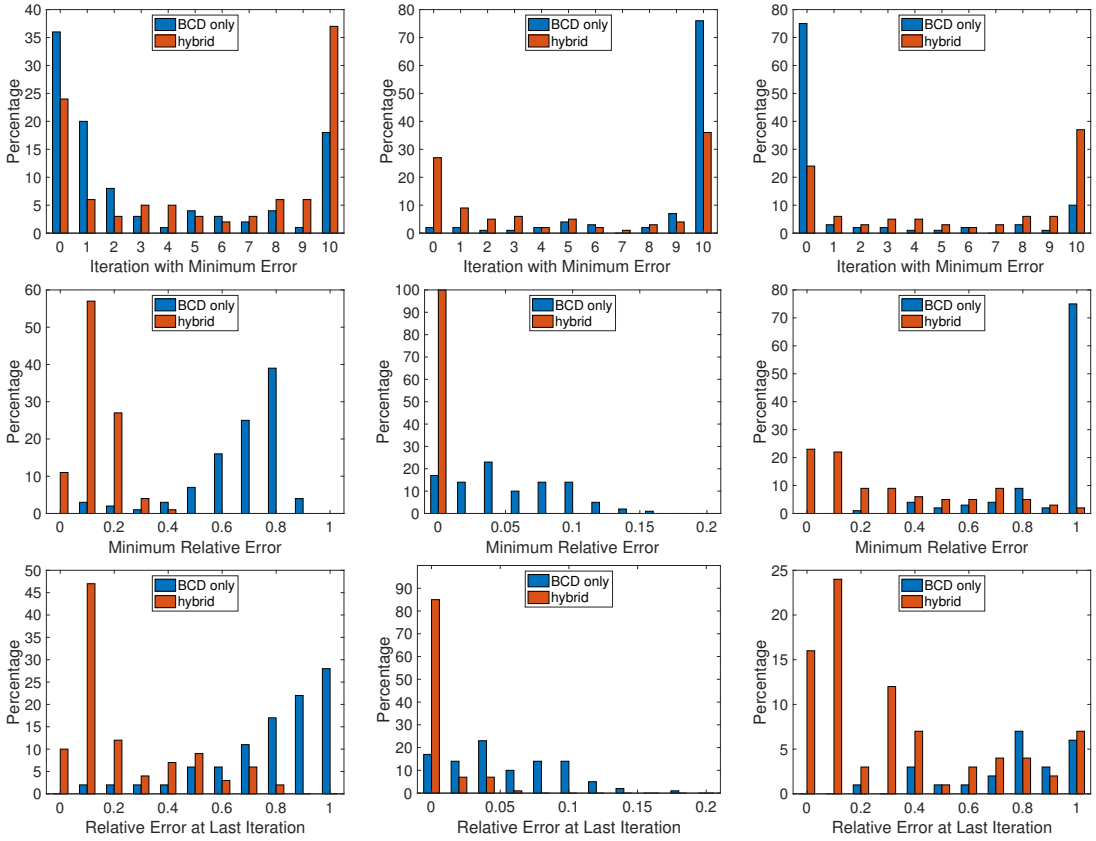
Figure 6.12: Relative errors in $\boldsymbol{R}_{\text{pred}}$ and $\boldsymbol{D}_{\text{pred}}$ on testing set.

We see in Figure 6.12 that the relative errors decrease for both training sets as training set size increases. If we compare the relative errors vertically, we can see that the errors are not so different for the two training sets. The slight difference may be

caused by randomness in generating the two sets. We believe it's only fair to compare results using training sets of the same size, since, for example, if we compare results given by one training set of 5000 samples with results given by another training set of 5000 samples each with 5 disturbances, the decrease in relative errors may be caused by the increased training set size, not disturbance. Therefore, given the results in this test, it is hard to conclude that disturbances help improve accuracy in the predicted geometry parameters.

Test 3: 128 by 238 varying phantom problem with $N_p = 6$. We have observed in Tests 1 and 2 that errors in the predicted geometry parameters would increase if the ML model is not trained and tested on the same phantom images. We have also seen that, in this case, although errors in $\boldsymbol{\delta}^\theta$ are quite large, relative errors in $\boldsymbol{r}$ are still relatively low.

In this test, we focus on the case where phantoms vary across training and testing samples. We increase phantom size to $n = 128$ and number of geometry parameters to $N_p = 6$. Hence, with 180 scanning angles 0:2:358, each pair of (r_i, δ_i^θ) will be constant for 30 adjacent angles. and compare solutions of the BCD (Algorithm 12), ML (Algorithm 13) and ML-BCD hybrid (Algorithm 15) methods. The ML weights are trained using the 40000 training samples. We evaluate the ML-BCD hybrid method and BCD method (each with 10 iterations) on a testing set of 100 samples, calculate the relative errors in $\boldsymbol{x}_k$, $\boldsymbol{r}_k$ and $\boldsymbol{\delta}_i^\theta$ for each BCD iteration k, and generate the histograms in Figure 6.13.



Solution $\boldsymbol{x}$. Parameter $\boldsymbol{r}$. Parameter $\boldsymbol{\delta}^{\theta}$.

Figure 6.13: 1st row: BCD iteration number with the minimum relative error. 2nd row: minimum relative error across all iterations. 3rd row: relative error at the 10th BCD iteration.

In the first row of Figure 6.13, frequencies of the iteration at which the minimum error occurs are plotted. Note that iteration 0 represents the initial guess, i.e., for BCD, this would be the uncalibrated solution; and for ML-BCD hybrid, this would be obtained directly from using ML-predicted parameters without further refining with BCD. We can draw the conclusion that the hybrid method has better convergence properties for $\boldsymbol{x}$, where for nearly 40% percent of testing samples, BCD reaches the

minimum relative error of $\boldsymbol{x}$ in the last iteration. While for BCD, in more than 50% percent of testing samples, the minimum relative error in $\boldsymbol{x}$ is either reached at the initial guess or the first iteration, which means that BCD is not very effective at improving solution accuracy. This is further confirmed in the distribution of relative errors in the second and third rows of Figure 6.13, in which we notice that in general, the relative errors in both the solutions and the calibrated geometry parameters given by the ML-BCD hybrid method are very low compared to BCD.

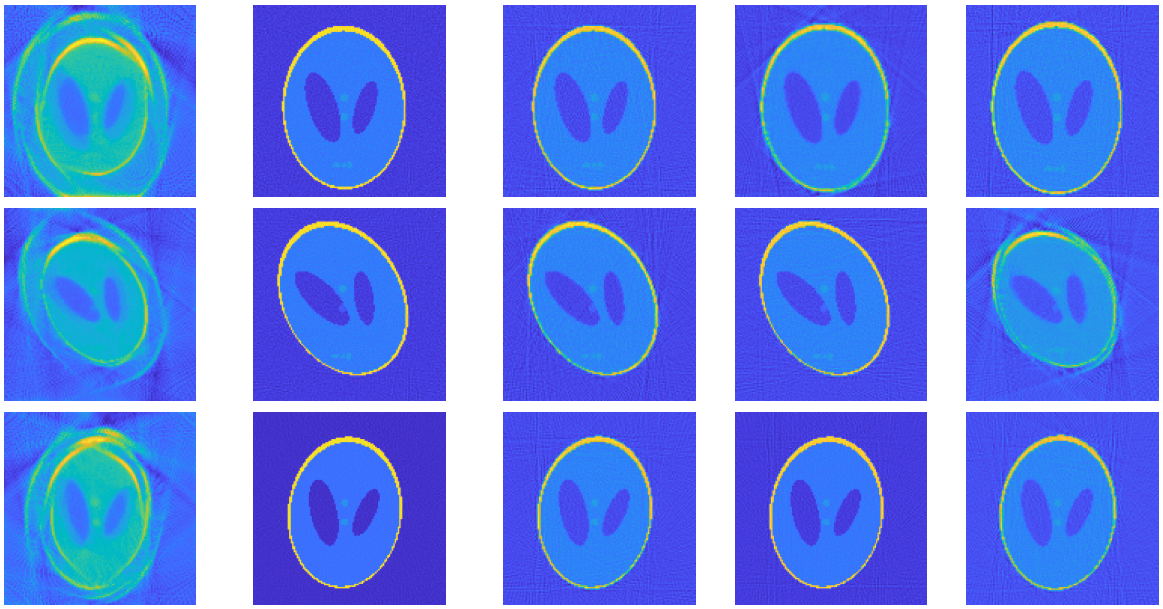


Figure 6.14: Phantom reconstructions of 3 testing samples (1 test sample per row). 1st column: solutions using theoretical parameters (i.e., uncalibrated). 2nd column: solutions using true parameters (i.e. perfect calibration). 3rd column: solutions using ML-predicted parameters (Algorithm 13). 4th column: solutions using ML-BCD hybrid method with 10 BCD iterations (Algorithm 15). 5th column: solutions using 10 BCD iterations (Algorithm 12).

In Figure 6.14, we display solutions of 3 test samples using different methods:

ML-only (Algorithm 13), ML-BCD hybrid (Algorithm 15) and BCD-only (Algorithm 12) (solutions at the last BCD iteration are shown). As a point of reference, we also present the solutions obtained using the true parameters to showcase the best solution possible. The 3 test samples are randomly picked from the 100 testing samples, and we see that for 2 out of 3, the BCD refinement step of the hybrid method has improved solution accuracy over using ML alone, and for these 2 samples, the hybrid solutions have significantly higher accuracy compared to the BCD solutions. For the first test sample, however, the BCD-refinement step of the hybrid method seems to not have improved solution accuracy. This is also expected because we have observed in the histograms of Figure 6.13 that in some cases, the hybrid method may not work so well – but it is evident that ML-BCD hybrid is more advantageous over BCD on average in the tasks of parameter calibration and image reconstruction.

Chapter 7

Conclusions and Future Work

In this thesis, we have introduced novel low-rank solvers for large scale linear systems. Firstly, we introduced a new Kronecker product summation based technique for approximating the TSVD for a large matrix $\boldsymbol{A}$. The TSVD can be thought of as a low-rank approximation to the original operator $\boldsymbol{A}$, and by filtering out the small singular values, it has a regularizing effect on the solution $\boldsymbol{x}$ to the linear system (1.1). By exploiting a reordering technique, we can effectively utilize more terms in the Kronecker summation for $\boldsymbol{A}$, and produce approximations of higher quality than existing Kronecker product based approximation methods.

Next, we explored the nuclear norm regularized problem (4.2) and derived new solvers, based on Krylov subspace methods, for the computation of approximate low-rank solutions to large-scale linear systems of equations. The starting point of our derivations was an IRN approach to the NNRp problem (4.3). In this way, the original problem (4.3) is reduced to the solution of a sequence of quadratic problems,

where an appropriate smoothed linear transformation is introduced to approximate the nondifferentiable nuclear norm regularization term. Our new methods make smart use of Kronecker product properties to reformulate each quadratic problem in the IRN sequence as a Tikhonov-regularized problem in standard form. We developed both Krylov methods with fixed "preconditioners" within an inner-outer iteration scheme, and Krylov methods with flexible iteration-dependent "preconditioners" within a single iteration scheme. Some of these methods can be used in a hybrid framework, so that the Tikhonov regularization parameter can be efficiently, effectively, and adaptively chosen. These new solvers are shown to perform exceptionally well on the test problems such as image deblurring and inpainting, and they give reconstructions of significantly improved quality over existing methods.

In the application of computed tomography, where the forward model $\boldsymbol{A}$ is dependent on unknown geometry parameters $\boldsymbol{p}$, model calibration is required at the time of image reconstruction. A block coordinate descent type algorithm can be applied, where the image $\boldsymbol{x}$ and unknown parameters $\boldsymbol{p}$ are alternatively minimized in each step. For the minimization with respect to $\boldsymbol{x}$, our new low-rank Krylov methods also show improved convergence result and reconstruction quality compared to standard Krylov methods with classical regularization such as ℓ_2. Furthermore, we also proposed an enhanced, hybrid BCD framework that incorporates machine learning for solving the model calibration problem. By training a machine learning model (multi-output linear regression model) that maps the observation $\boldsymbol{b}$ to $\boldsymbol{p}$ and feeding the predicted parameters as initial guesses to BCD, we are able to improve the accuracy of both the calibrated geometry parameters and the reconstructed image.

Interesting future directions include, for example, extending present low-rank Krylov methods to handle cases where the solution of (4.1) is low-rank but rectangular, i.e., $\text{vec}^{-1}(\boldsymbol{x}) = \boldsymbol{X} \in \mathbb{R}^{m\times n}$ with $m \neq n$. Also, while a solid theoretical justification is provided for IRN-LSQR-NNRp and IRN-GMRES-NNRp, the same is not true for FGMRES-NNRp and FLSQR-NNRp: further analysis will be needed to deeply understand the regularization properties of these flexible solvers. In the context of model calibration, we also see great potential of flexible Krylov subspace methods, since we may be able to design iteration-dependent "preconditioners" so that they contain updated information about both the approximate solution $\boldsymbol{x}$ and model parameters $\boldsymbol{p}$. State-of-the-art machine learning techniques could be accommodated to to learn the necessary preconditioners that not only enforce regularization properties on the solutions, but also improve model accuracy. Inverse problems can be found in a wide range scientific fields, and is an active, broad and fast-evolving research area. Although this thesis only scratched the surface of what can be done with inverse problems, we hope the new methods proposed here will inspire and promote future research in many other interesting topics.

Bibliography

[1] B. M. Afkham, J. Chung, and M. Chung, *Learning Regularization Parameters of Inverse Problems via Deep Neural Networks*, 2021.

[2] A. P. Austin, Z. W. Di, S. Leyffer, and S. M. Wild, *Simultaneous Sensing Error Recovery and Tomographic Inversion Using an Optimization-Based Approach*, SIAM Journal on Scientific Computing, 41 (2019), pp. B497–B521.

[3] M. Beckers, T. Senkbeil, T. Gorniak, K. Giewekemeyer, T. Salditt, and A. Rosenhahn, *Drift Correction in Ptychographic Diffractive Imaging*, Ultramicroscopy, 126 (2013), pp. 44 – 47.

[4] M. Belge, M. E. Kilmer, and E. L. Miller, *Wavelet Domain Image Restoration with Adaptive Edge-Preserving Regularization*, IEEE Trans. Image Process., 9 (2000), pp. 597–608.

[5] J. M. Bioucas-Dias and M. A. T. Figueiredo, *A New TwIST: Two-Step Iterative Shrinkage/Thresholding Algorithms for Image Restoration*, IEEE Transactions on Image Processing, 16 (2007), pp. 2992–3004.

[6] J. D. Blanchard, J. Tanner, and K. Wei, *CGIHT: Conjugate Gradient Iterative Hard Thresholding for Compressed Sensing and Matrix Completion*, Information and Inference: A Journal of the IMA, 4 (2015), pp. 289–327.

[7] J. Cai, E. Candès, and Z. Shen, *A Singular Value Thresholding Algorithm for Matrix Completion*, SIAM Journal on Optimization, 20 (2010), pp. 1956–1982.

[8] D. Calvetti, S. Morigi, L. Reichel, and F. Sgallari, *Tikhonov Regularization and the L-Curve for Large Discrete Ill-Posed Problems*, Journal of Computational and Applied Mathematics, 123 (2000), pp. 423–446. Numerical Analysis 2000. Vol. III: Linear Algebra.

[9] J. Cant, A. Snoeckx, G. Behiels, P. M. Parizel, and J. Sijbers, *Can Portable Tomosynthesis Improve the Diagnostic Value of Bedside Chest X-ray in the Intensive Care Unit? A Proof of Concept Study*, European radiology experimental, (2017), p. 20.

[10] M. Cao, A. Takaoka, H. Zhang, and R. Nishi, *An Automatic Method of Detecting and Tracking Fiducial Markers for Alignment in Electron Tomography*, Microscopy, 60 (2011), pp. 39–46.

[11] J. Chung and S. Gazzola, *Flexible Krylov Methods for ℓ_p Regularization*, SIAM Journal on Scientific Computing, 41 (2019), pp. S149–S171.

[12] J. Chung, J. G. Nagy, and D. M. O'Leary, *A weighted-gcv method for lanczos-hybrid regularization.*, Electronic Transactions on Numerical Analysis, 28 (2007), pp. 149–167.

[13] R. D. Fierro and J. R. Bunch, *Bounding the Subspaces from Rank Revealing Two-Sided Orthogonal Decompositions*, SIAM J. Matrix Anal. Appl., 16 (1995), pp. 743–759.

[14] M. Fornasier, H. Rauhut, and R. Ward, *Low-Rank Matrix Recovery via Iteratively Reweighted Least Squares Minimization*, SIAM Journal on Optimization, 21 (2011), pp. 1614–1640.

[15] C. Garvey, *Truncated Singular Value Decomposition Approximation for Structured Matrices via Kronecker Product Summation Decomposition*, Theses and Dissertations, Emory University, (2018).

[16] C. Garvey, C. Meng, and J. G. Nagy, *Singular Value Decomposition Approximation via Kronecker Summations for Imaging Applications*, SIAM Journal on Matrix Analysis and Applications, 39 (2018), pp. 1836–1857.

[17] S. Gazzola, P. C. Hansen, and J. G. Nagy, *IR Tools: A MATLAB Package of Iterative Regularization Methods and Large-Scale Test Problems*, Numerical Algorithms, 81 (2019), pp. 773–811.

[18] S. Gazzola, C. Meng, and J. G. Nagy, *Krylov Methods for Low-Rank Regularization*, SIAM Journal on Matrix Analysis and Applications, 41 (2020), pp. 1477–1504.

[19] S. Gazzola and J. G. Nagy, *Generalized Arnoldi-Tikhonov Method for Sparse Reconstruction*, SIAM J. Sci. Comput., 36 (2014), pp. 225–247.

[20] S. Gazzola and P. Novati, *Automatic Parameter Setting for Arnoldi-Tikhonov Methods*, J. Comput. Appl. Math., 256 (2014), pp. 180–195.

[21] S. Gazzola, P. Novati, and M. R. Russo, *On Krylov Projection Methods and Tikhonov Regularization*, Electronic Transactions on Numerical Analysis, 44 (2014), pp. 82–123.

[22] S. Gazzola and M. Sabaté Landman, *Flexible GMRES for Total Variation Regularization*, BIT Numerical Mathematics, 59 (2019), pp. 721–746.

[23] D. Goldfarb and S. Ma, *Convergence of Fixed-Point Continuation Algorithms for Matrix Rank Minimization*, Foundations of Computational Mathematics, 11 (2011), pp. 183–210.

[24] G. H. Golub and C. F. Van Loan, *Matrix Computations*, John Hopkins University Press, 4 ed., 2013.

[25] I. F. Gorodnitsky and B. D. Rao, *A New Iterative Weighted Norm Minimization Agorithm and its Applications*, in [1992] IEEE Sixth SP Workshop on Statistical Signal and Array Processing, Oct 1992, pp. 412–415.

[26] D. Gürsoy, Y. P. Hong, K. He, K. Hujsak, S. Yoo, S. Chen, Y. Li, M. Ge, L. M. Miller, Y. S. Chu, V. De Andrade, K. He, O. Cossairt, A. K. Katsaggelos, and C. Jacobsen, *Rapid Alignment of Nanotomogra-*

phy Data Using Joint Iterative Reconstruction and Reprojection, Scientific Reports, 7 (2017), p. 11818.

[27] N. Halko, P. G. Martinsson, and J. A. Tropp, *Finding Structure with Randomness: Probabilistic Algorithms for Constructing Approximate Matrix Decompositions*, SIAM Review, 53 (2011), pp. 217–288.

[28] M. Hanke and P. C. Hansen, *Regularization Methods for Large-Scale Problems*, Surv. Math. Ind., 3 (1993), pp. 253–315.

[29] P. C. Hansen, *REGULARIZATION TOOLS: A Matlab Package for Analysis and Solution of Discrete Ill-Posed Problems*, Numerical Algorithms, 6 (1994), pp. 1–35.

[30] ——, *Rank-Deficient and Discrete Ill-Posed Problems: Numerical Aspects of Linear Inversion*, Society for Industrial and Applied Mathematics, USA, 1999.

[31] ——, *Discrete Inverse Problems: Insight and Algorithms*, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2010.

[32] P. C. Hansen and T. K. Jensen, *Smoothing-Norm Preconditioning for Regularizing Minimum-Residual Methods*, SIAM J. Matrix Anal. Appl., 29 (2006), pp. 1–14.

[33] P. C. Hansen, J. G. Nagy, and D. P. O'Leary, *Deblurring Images: Matrices, Spectra, and Filtering*, Society for Industrial and Applied Mathematics, Philadelphia, PA, 2006.

[34] P. C. Hansen and D. P. OLeary, *The Use of the L-Curve in the Regularization of Discrete Ill-Posed Problems*, SIAM Journal on Scientific Computing, 14 (1993), pp. 1487–1503.

[35] R. Huang, R. Diao, Y. Li, J. Sanchez-Gasca, Z. Huang, B. Thomas, P. Etingov, S. Kincic, S. Wang, R. Fan, G. Matthews, D. Kosterev, S. Yang, and J. Zhao, *Calibrating Parameters of Power System Stability Models Using Advanced Ensemble Kalman Filter*, IEEE Transactions on Power Systems, 33 (2018), pp. 2895–2905.

[36] X. Huang, S. M. Wild, and Z. W. Di, *Calibrating Sensing Drift in Tomographic Inversion*, in 2019 IEEE International Conference on Image Processing (ICIP), IEEE, 2019, pp. 1267–1271.

[37] A. C. Hurst, T. B. Edo, T. Walther, F. Sweeney, and J. M. Rodenburg, *Probe Position Recovery for Ptychographical Imaging*, Journal of Physics: Conference Series, 241 (2010), p. 012004.

[38] A. Jacobi, M. Chung, A. Bernheim, and C. Eber, *Portable Chest X-ray in Coronavirus Disease-19 (Covid-19): A Pictorial Review*, Clinical imaging, 64 (2020), pp. 35–42.

[39] A. K. Jain, *Fundamentals of Digital Image Processing*, Prentice-Hall, Inc., USA, 1989.

[40] T. K. Jensen and P. C. Hansen, *Iterative Regularization with Minimum-Residual Methods*, BIT, 47 (2007), pp. 103–120.

[41] J. Kamm and J. G. Nagy, *Kronecker Product and SVD Approximations in Image Restoration*, Linear Algebra and its Applications, 284 (1998), pp. 177–192.

[42] C. T. Kelley, *Users' Guide for imfil Version 1.0*, 2011.

[43] R. H. Keshavan, A. Montanari, and S. Oh, *Matrix Completion from Noisy Entries*, J. Mach. Learn. Res., 11 (2010), pp. 2057–2078.

[44] M. E. Kilmer and D. O'Leary, *Choosing Regularization Parameters in Iterative Methods for Ill-Posed Problems*, SIAM J. Matrix Anal. Appl., 22 (2001), pp. 1204–1221.

[45] D. Kressner and C. Tobler, *Low-Rank Tensor Krylov Suspace Methods for Parameterized Linear Systems*, SIAM J. Matrix Anal. Appl., 32 (2011), pp. 1288–1316.

[46] L. Landweber, *An Iteration Rormula for Fredholm Integral Equations of the First Kind*, American Journal of Mathematics, 73 (1951), pp. 615–624.

[47] R. M. Larsen, *Lanczos Bidiagonalization with Partial Reorthogonalization*, Department of Computer Science, Aarhus University, Technical report, (1998).

[48] K. Lee and H. C. Elman, *A Preconditioned Low-Rank Projection Method with a Rank-Reduction Scheme for Stochastic Partial Differential Equations*, SIAM J. Sci. Comput., 9 (2017), pp. 828–850.

[49] Z. Li, Q. Xu, and Y. Wei, *A Note on Stable Perturbations of Moore-Penrose Inverses*, Numer. Linear Algebra Appl., 20 (2011), pp. 18–26.

[50] S. Ma, D. Goldfarb, and L. Chen, *Fixed Point and Bregman Iterative Methods for Matrix Rank Minimization*, Mathematical Programming, 128 (2011), pp. 321–353.

[51] K. Mohan and M. Fazel, *Iterative Reweighted Algorithms for Matrix Rank Minimization*, J. Mach. Learn. Res., 13 (2012), pp. 3441–3473.

[52] J. G. Nagy, *Decomposition of Block Toeplitz Matrices into a Sum of Kronecker Products with Applications in Image Restoration*, tech. rep., Southern Methodist University, 1996.

[53] D. O'leary and B. Rust, *Variable Projection for Nonlinear Least Squares Problems*, Computational Optimization and Applications, 54 (2013), pp. 579–593.

[54] C. C. Paige and M. A. Saunders, *LSQR: An Algorithm for Sparse Linear Equations and Sparse Least Squares*, ACM Trans. Math. Softw., 8 (1982), pp. 43–71.

[55] V. PATHURI-BHUVANA, S. SCHUSTER, and A. OCH, *Joint calibration and tomography based on separable least squares approach with constraints on linear and non-linear parameters*, in 2020 28th European Signal Processing Conference (EUSIPCO), 2021, pp. 1931–1935.

[56] J. Radon, *Über die Bestimmung von Funktionen durch ihre Integralwerte längs gewisser Mannigfaltigkeiten*, Akad. Wiss., 69 (1917), pp. 262–277.

[57] B. Recht, M. Fazel, and P. Parrilo, *Guaranteed Minimum-Rank Solutions of Linear Matrix Equations via Nuclear Norm Minimization*, SIAM Review, 52 (2010), pp. 471–501.

[58] N. A. Riis, Y. Dong, and P. C. Hansen, *Computed Tomography Reconstruction with Uncertain View Angles by Iteratively Updated Model Discrepancy*, Journal of Mathematical Imaging and Vision, 63 (2021), pp. 133–143.

[59] N. A. B. Riis, Y. Dong, and P. C. Hansen, *Computed Tomography with View Angle Estimation Using Uncertainty Quantification*, Inverse Problems, 37 (2021), p. 065007.

[60] R. T. Rockafellar, *Convex Analysis*, vol. 28, Princeton university press, 1970.

[61] P. Rodríguez and B. Wohlberg, *An Efficient Algorithm for Sparse Representations with ℓ_p Data Fidelity Term*, in Proceedings of 4th IEEE Andean Technical Conference (ANDESCON), Cusco, Perú, Oct 2008.

[62] M. C. Roggemann and B. Welsh, *Imaging Through Turbulence*, CRC Press, Boca Raton, FL, 1996.

[63] Y. Saad, *A Flexible Inner-Outer Preconditioned GMRES Algorithm*, SIAM Journal on Scientific Computing, 14 (1993), pp. 461–469.

[64] Y. Saad, *Iterative Methods*, Society for Industrial and Applied Mathematics, 2 ed., 2003.

[65] Y. Saad and M. H. Schultz, *GMRES: A Generalized Minimal Residual Algorithm for Solving Nonsymmetric Linear Systems*, Siam Journal on Scientific and Statistical Computing, 7 (1986), pp. 856–869.

[66] V. Simoncini and D. B. Szyld, *Recent Computational Developments in Krylov Subspace Methods for Linear Systems*, Numerical Linear Algebra with Applications, 14 (2007), pp. 1–59.

[67] M. Stoll and T. Breiten, *A Low-Rank in Time Approach to PDE-Constrained Optimization*, SIAM J. Sci. Comput., 27 (2015), pp. B1–B29.

[68] C. F. Van Loan and N. Pitsianis, *Approximation with Kronecker Products*, in Linear Algebra for Large Scale and Real-Time Applications, Springer, 1993, pp. 293–314.

[69] C. R. Vogel, *Computational Methods for Inverse Problems*, Society for Industrial and Applied Mathematics, 2002.

[70] P.-Å. Wedin, *Perturbation Theory for Pseudoinverses*, BIT, 13 (1973), pp. 217–232.

[71] Y. Wu, *Model Parameter Calibration in Power Systems*, Graduate College Dissertations and Theses, University of Vermont, (2020).