Large-scale optimization algorithms for missing data completion and inverse problems

by

Curt Da Silva

B.Sc., The University of British Columbia, 2011

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF

Doctor of Philosophy

in

THE FACULTY OF GRADUATE AND POSTDOCTORAL STUDIES
(Mathematics)

The University of British Columbia
(Vancouver)

September 2017

Abstract

Inverse problems are an important class of problems found in many areas of science and engineering. In these problems, one aims to estimate unknown parameters of a physical system through indirect multi-experiment measurements. Inverse problems arise in a number of fields including seismology, medical imaging, and astronomy, among others.

An important aspect of inverse problems is the quality of the acquired data itself. Real-world data acquisition restrictions, such as time and budget constraints, often results in measured data with missing entries. Many inversion algorithms assume that the input data is fully sampled and relatively noise free and produce poor results when these assumptions are violated. Given the multidimensional nature of real-world data, we propose a new low-rank optimization method on the smooth manifold of Hierarchical Tucker tensors. Tensors that exhibit this low-rank structure can be recovered from solving this non-convex program in an efficient manner. We successfully interpolate realistically sized seismic data volumes using this approach.

If our low-rank tensor is corrupted with non-Gaussian noise, the resulting optimization program can be formulated as a convex-composite problem. This class of problems involves minimizing a non-smooth but convex objective composed with a nonlinear smooth mapping. In this thesis, we develop a level set method for solving composite-convex problems and prove that the resulting subproblems converge linearly. We demonstrate that this method is competitive when applied to examples in noisy tensor completion, analysis-based compressed sensing, audio declipping, total-variation deblurring and denoising, and one-bit compressed sensing.

With respect to solving the inverse problem itself, we introduce a new software design framework that manages the cognitive complexity of the various components involved. Our framework is modular by design, which enables us to easily integrate and replace components such as linear solvers, finite difference stencils, preconditioners, and parallelization schemes. As a result, a researcher using this framework can formulate her algorithms with respect to high-level components such as objective functions and hessian operators. We showcase the ease with which one can prototype such algorithms in a 2D test problem and, with little code modification, apply the same method to large-scale 3D problems.

Lay Summary

Inverse problems are a class of important problems in science and engineering applications, wherein we measure the response of a physical system and want to infer the intrinsic parameters of the system that produced those measurements. For instance, medical imaging attempts to reconstruct an image of the body's internal structure given electric field data measured along the skin. The measured data itself must be of sufficiently high quality and coverage in order to estimate these parameters accurately. In this thesis, we develop methods for completing data that has not been fully collected, due to time or budget constraints. We also study a class of optimization problems that can handle when our input data is contaminated with large noisy outliers. The final topic presented in this thesis involves designing software for solving these inverse problems in an efficient, flexible, and scalable manner.

Preface

This thesis consists of my original research, conducted at the Department of Mathematics at the University of British Columbia, Vancouver, Canada, under the supervision of Professor Felix Herrmann as part of the Seismic Laboratory for Imaging and Modelling (SLIM). The following chapters contain previously published or submitted work for which I was the principal investigator and author. I formulated this research program, developed the resulting theory and numerical software, and wrote the entirety of the articles below, subject to minor editorial suggestions from my supervisor. Tristan van Leeuwen gave some valuable input on the presentation of Chapter (5) and Aleksandr Aravkin gave helpful feedback on an early version of Chapter (4).

A version of Chapter (2) was published in [82], which significantly extends earlier conference proceedings in [79, 81]. A version of Chapter (5) has been submitted for publication as [83], which expands upon the conference proceedings in [80], and is currently under peer review.

Table of Contents

List of Tables

List of Figures

List of Algorithms

Acknowledgements

I would first and foremost like to thank my supervisor, Dr. Felix Herrmann, who initially convinced me to join the SLIM group over the course of coffee and regaling me with all of the interesting problems that his group tackled. Collectively, the group has tackled a large number of them, but more seem to keep springing up. Alas, such is the course of research. You have been a avid and enthusiastic supervisor, always in pursuit of solving more interesting yet relevant problems, and that mentality is definitely infectious. Thank you for the constant doses of perspective, it's easy to miss the big picture sometimes and you've taught me to always keep it in mind.

Thank you to my talented and insightful colleagues with whom I've worked closely, Rajiv Kumar and Zhilong Fang, you are all very talented and, more importantly, hard working. I know that you will all go far in life. Thank you as well to former postdoctorate fellows in the group Tristan van Leeuwen and Aleksandr Aravkin, who helped me develop my understanding of optimization during their time with SLIM and continued to provide valuable input on my work even after moving on to other ventures.

Thanks especially to my mother, father, and sister, who have always encouraged me to pursue my education and gave me every opportunity to do so through their hard work. This one's for you!

To my wonderful girlfriend Andrea, you have been so supportive and loving to me throughout my time in graduate school. Now it's time for more adventures together!

To all of my friends at Instant Theatre and elsewhere in the Vancouver comedy scene, you kept me sane during this whole process.

To my wonderful friends and family

Chapter 1

Introduction

Inverse problems are ubiquitous in many science and engineering applications. Inverse problems aim to estimate an unknown quantity of interest (e.g., sound speed of the earth, tissue conductivity) from indirect measurements on the surface or boundary of a region. These problems arise in a large number of fields, including seismology [261, 243, 244], medical imaging [71, 18, 102, 42], structural engineering [138], chemistry [33], radar imaging [35], quantum scattering theory [62], astronomy [78], and image processing [26, 100], among others.

Inverse problems are described by a *forward modelling operator*, denoted F, which maps parameters m to predicted data $d = F(m)$. Encoded in F is the entirety of our underlying assumptions about the physics of our model and the geometry of our acquisition. This mapping is typically nonlinear and differentiable. The goal of an inverse problem is to find parameters m^* that generate data that are as close as possible to our measured data d_{meas}, i.e.,

$$F(m^*) \approx d_{\text{meas}}.$$

More precisely, we often compute m^* as the solution to the optimization problem

$$\min_m \phi(F(m), d_{\text{meas}})$$

where ϕ is the measure of misfit between our predicted and observed data. Solving an inverse problem provides with us a noninvasive means to "see" into a hidden region of space. The estimated parameters then form the image of our region of interest, whether that be a conductivity map of a potentially cancerous tissue or a velocity model of the subsurface of the Earth. For a comprehensive review of computational methods for inverse problems, we refer the interested reader to [262].

An important aspect of such problems is the measured data itself, which is often required to be fully sampled and relatively noise-free in order for inversion algorithms to produce meaningful results. These data volumes are often multidimensional, depending on several spatial or angular coordinates as well as time. As

a result, sampling say N points in each of the d dimensions leads to storage costs on the order of N^d points, an exponential dependence on the number of dimensions. This is the so-called *curse of dimensionality*, which creates onerous storage and computational costs when processing these volumes. Insisting that the data be fully acquired in all dimensions incurs enormous time and budgetary costs as practitioners aim to reconstruct these volumes through classical Whittaker–Nyquist–Kotelnikov–Shannon sampling [226, 264, 181]. Broadly speaking, in order to reconstruct a one-dimensional signal with bandwidth B Hertz, one must acquire at least $2B$ samples per unit interval. Although mathematically elegant, this sampling paradigm is incredibly restrictive for reconstructing realistic data volumes due to their high dimensionality. As a result, fully sampling a data volume is an arduous task that we will instead avoid in favour of measuring far fewer samples of the data volume.

Real-world signals often possess much more structure than being merely band-limited, which can be exploited for recovering the signal from subsampled measurements. For instance, many natural images have been shown empirically to be sparse (i.e., possess a small number of non-zero coefficients relative to the ambient dimensionality) in the wavelet basis [99]. The field of compressed sensing (CS) aims to exploit the sparsity of the signal (in a particular basis) in order to acquire samples at sub-Nyquist rates. A practitioner samples the signal at a rate commensurate with the *sparsity of the signal* rather than the dimensionality of the ambient space and, from this subsampled data, one reconstructs the signal by solving an associated optimization program. There have been a large number of successful applications of CS to recover sparse signals from subsampled measurements, including medical imaging [177, 68], radar imaging [123], and geophysics [129, 125, 130]. The attractive theoretical and numerical results of CS have motivated researchers to consider various extensions to other low-dimensional structures embedded in high-dimensional spaces. The notion of low-rank for matrices is directly analogous to sparsity for one-dimensional signals, whereby one imposes a sparsity constraint on the singular values of a given matrix. Many similar theoretical guarantees can be proved for recovering low-rank matrices from a subset of missing entries, the so-called matrix completion problem, and a variety of successful techniques have arisen for solving these problems [52, 49, 12]. Extending these notions to higher-dimensional tensors proves challenging as one no longer has the benefit of having a unique singular value decomposition (SVD) for tensors. Given that a large amount of numerical and theoretical work for matrix completion is predicated on the decomposition of a matrix in to its constituent SVD components, even generalizing the definition of a low-rank tensor is non-trivial.

This question of whether one should interpolate the data or use some other technique to solve the corresponding inverse problem is open, in general, despite the working assumption by practitioners that having more data leads to more robust results. The authors in [218] show that insisting that the data be fully sampled can be relaxed in some inverse problem scenarios involving DC-resistivity. The work of [6] explores using a regularized least-squares approach to solving a financial inverse problem compared to data completion. The authors also consider the problem

of uncertainty in the locations of the measured data, and similarly in [7]. For the linearized seismic imaging problem, [189] demonstrates that using standard inversion algorithms with missing data results in exceptionally poor results, leading to a large imprint of the gap in source/receiver coverage in the inverted model, i.e., a large so-called acquisition footprint. Thus far, there are few definitive answers as to whether one should complete missing data prior to inversion for general problems. In this thesis, we will proceed as if the process of interpolating data, whether to produce a fully sampled volume for inversion or for its own sake, is a worthwhile endeavour.

The presence of noise in acquired signals induces a corresponding noise model in the resulting optimization program. Due to its simplicity, the most common assumption made due is that the noise obeys a multivariate Gaussian distribution, often with an identity covariance matrix, which results in minimizing an $\ell 2$-norm difference between the observed and predicted data. Although desirable from a practical point of view, since the data misfit is smooth and easy to minimize, this noise model is often not realistic. In particular, the $\ell 2$ norm is particularly brittle in the presence of *gross* outliers, where a small subset of the signal of interest is corrupted by high magnitude values, e.g., dead pixels in a video stream or a few receivers in an array failing in a seismic experiment [143]. Introducing a penalty that is more robust to such outliers such as the $\ell 1$ norm [266] can alleviate such issues as it penalizes large residual values far less than the corresponding $\ell 2$ penalty. Solving such data recovery problems in this case, however, becomes algorithmically challenging due to the non-smoothness of the resulting data misfit and resulting constraints that do not possess simple and efficient projection operators. Even in the ideal noise-free case, promoting sparsity or low-rank in an given optimization program from first principles is an involved procedure as it involves non-smooth optimization.

Assuming that we have successfully reconstructed a reasonable approximation of our full-sampled data volume, we focus our attention on solving the resulting inverse problem itself, which is an interesting software design challenge. There are a large number of research areas in which one must be well-versed in order to solve inverse problems, but software design is often not among them. As such, many academic researchers have designed codes that are mathematically correct, but scale poorly in terms of memory usage or computational complexity to realistically-sized problems. In industrial settings, the large computational costs of solving such problems often results in code that is exceedingly fast after many programmers hours spent hand-tuning higher-level code. This process produces software that lacks flexibility and is not easily modifiable, and in many cases may not even satisfy essential mathematical properties such as the computed gradient being the true gradient of the objective functional.

The main purpose of this thesis is to develop new techniques in the problems of tensor completion, composite-convex optimization, and software design for inverse problems. Our developments in tensor completion will allow us to recover low-rank tensors with missing entries by exploiting the geometric structure of the particular tensor format we consider. These algorithms will allow us to efficiently complete

large data volumes. For our composite-convex work, we propose a novel method for solving this class of optimization programs that will enable us to solve robust tensor completion problems, allowing us to denoise and interpolate input data that has been corrupted by high-amplitude noise. This class of optimization programs is also very general, which will allow us to solve similar problems such as one-bit compressed sensing and audio declipping. The software design that we propose in this thesis helps manage the complexity of constructing a framework for solving these problems through a hierarchical design. Our approach bridges the gap between purely performance-oriented codes and the mathematical objects that they represent, allowing us to solve large-scale inverse problems on a distributed cluster in a straightforward manner. Although seemingly disparate topics at first glance, they all comprise important elements of the entire process, from start to finish, of solving inverse problems in real-world contexts. One of the primary focuses in this work is developing algorithms and mathematical software that scales effectively to large-scale problems.

1.1 Missing Data Completion

The field of compressed sensing has revolutionized the field of signal acquisition and reconstruction since the initial papers were published in the mid-2000s [57, 58, 90], wherein the authors studied the reconstruction of sparse signals from a sub-Nyquist number of random measurements. This work was insightful not merely for the application of $\ell 1$-based signal reconstruction, which was observed to succeed empirically in the Geophysical literature since the 1970s [74, 245], but also for the subsequent rigor used in theoretically proving that this method reconstructs the signal with high probability under certain conditions on the sampling operator. This work also generated significant interest for practitioners by showing that one could forgo solving the original sparsity-promoting program, which was NP-Hard, and instead seek the solution of its convex relaxation, which can be solved in polynomial time. Under particular conditions on the sampling and measurement operators, these two solutions are identical.

Generalizing such ideas to the recovery of matrix-valued signals, i.e., two-dimensional signals versus the one-dimensional vectors considered in CS, leads to the notion of matrix completion. One of the most well known instances of matrix completion is the Netflix problem [27], which aims to fill-in a user-movie matrix X^*, where the entries $X^*_{i,j}$ correspond to the rating, between 1 and 5 stars, that the i^{th} user gave the j^{th} movie. Given that any given Netflix user watches a small number of movies relative to the total number of movies available, this matrix has a large number of missing entries. Netflix originally offered a \$1 million prize to the researchers who developed an algorithm that would improve on the accuracy of their internal recommender system by 10%. One of the techniques that spurred a large amount of researcher interest, although ultimately did not win the prize, was the assumption that the underlying, unknown matrix X was low-rank

[158]. That is to say, given the singular value decomposition of $X = USV^T$, with $U^TU = I_m$, $V^TV = I_n$, and $S = \text{diag}(\sigma)$ with $\sigma = (\sigma_1, \sigma_2, \ldots, \sigma_p)$ the vector of singular values $\sigma_1 \geq \sigma_2 \geq \ldots \sigma_p \geq 0$, $p = \min(m, n)$, the rank r, i.e., the integer such that $\sigma_r > \sigma_{r+1} = \cdots = \sigma_p = 0$, satisfies $r \ll p$. We denote $\sigma_{\max}(X) = \sigma_1$ and $\sigma_{\min}(X) = \sigma_p$. Under this low-rank assumption, one can estimate X from solving the following optimization problem

$$\begin{aligned} &\min_X \|X\|_* \\ &\text{such that } \; P_\Omega X = b. \end{aligned} \tag{1.1}$$

Here $P_\Omega X$ is the matrix satisfying

$$(P_\Omega X)_{i,j} = \begin{cases} X_{i,j} & \text{if } (i,j) \in \Omega \\ 0 & \text{otherwise} \end{cases},$$

where $\Omega = \{(i,j) : (i,j) \text{ is known}\}$ and $b = P_\Omega X^*$, for simplicity. The nuclear norm $\|X\|_* = \sum_{i=1}^k \sigma_i$ is the sum of the singular values of X, i.e., $\|X\|_* = \|\sigma\|_1$. Under appropriate conditions on the sampling operator, specifically that Ω is chosen uniformly at random with $|\Omega| \geq Ck(m+n)\log^2(n)$ for some constant C, and the incoherence of the matrix X^* itself, the solution to the convex problem (1.1) is exactly X^* [213]. Many other applications of matrix completion have arisen in geophysics [270, 164, 9], genomics [50], and phase retrieval [60], to name a few.

Problem (1.1) is a convex program and, as such, any local minimizer is a global minimizer. Algorithms that attempt to solve (1.1) via nuclear norm thresholding, such as in [49], scale poorly as the ambient dimensions m, n become large owing to the necessity of computing per-iteration SVDs. Techniques that circumvent this bottleneck, and thus scale much more reasonably, involve representing X in terms of its low-rank factors $X = LR^T$ [214, 12] or in terms of its SVD factors $X = USV^T$, with the accompanying orthogonality requirements [260]. In this case, the number of parameters is drastically reduced compared to the ambient space and the resulting operations do not involve computing SVDs of large matrices.

Attempts to generalize the theoretical and numerical machinery from matrix completion to completing multidimensional tensors is fraught with a number of complications. The rank of a matrix X is defined in terms of its singular value decomposition, which does not possess a unique extension in more than two dimensions. Attempts to generalize a particular aspect of the SVD result in differing tensor formats, each of which have their own advantages and disadvantages. Requiring that a tensor $\mathbf{X}$ be written as a sum of outer products of rank-one tensors leads to the so-called Candecomp/Parafac (CP) decomposition [119], which leads to a low number of effective parameters but the notion of tensor rank is difficult to analyze for generic tensors. The Tucker format [134, 250] generalizes the CP format, removing some of its theoretical difficulties at the expense of restoring the exponential dependence of the number of parameters on the dimension. The Hierarchical Tucker format [115] overcomes the shortcomings of these other formats by recursively splitting subsets of

dimensions. As we shall see, this format possesses a large amount of structure that we can exploit for completing low-rank tensors with missing entries. Specifically, as shown in [253], the set of such tensors is a smooth manifold, which is a smooth, nonlinear set that locally behaves like standard Euclidean space $\mathbb{R}^n$. We formulate algorithms on this tensor manifold in a similar manner to those specified on various matrix manifolds [1].

1.2 Convex Composite Optimization

Consider the following data-fitting problem

$$d = \psi(x) + n$$

where ψ captures the linear or nonlinear mapping of parameters x to data $\psi(x)$ and n is unknown noise. When n is impulsive, that is to say, it has high amplitudes but is spatially sparse, n is often modelled as an independently and identically distributed random variable with a Laplace probability distribution, where the probability density function for the i^{th} component of n, denoted n_i, satisfies

$$p(n_i) \propto e^{-\alpha|n_i|_1}.$$

The maximum likelihood estimator for x is the solution to the problem

$$\max_x e^{-\alpha\|\psi(x)-d\|_1}$$

or, equivalently, taking the negative logarithm of the above expression

$$\min_x \|\psi(x) - d\|_1. \tag{1.2}$$

This idea was developed in [247] in the context of the robust LASSO estimator, which is equivalent to using a Laplacian prior on the residuals. When ψ is a linear mapping, $\psi \in \mathbb{R}^{m\times n}$, this is the famous least absolute deviation regression problem [206]. Minimizing the $\ell 1$ norm can be written as a linear program [24], for which there have been many methods developed [176]. Solving (1.2) is computationally challenging when the matrix ψ contains a large number of rows or columns and various techniques have been proposed to reduce the dimensionality of the problem through randomized subsampling [269, 75]. In particular, although the problem is convex, the $\ell 1$ norm is not smooth and applying straightforward subgradient methods converge at a sublinear rate [40, 232], which is far too slow for realistic problems.

Problem (1.2) is a particular instance of composite-convex optimization, a general class of problems of the form

$$\min_x h(c(x))$$

where h is a convex, typically non-smooth function and c is a linear or smooth nonlinear mapping. We will develop algorithms for this class of problems further in this thesis.

1.3 Inverse Problems

Inverse problems can generally be considered as data-fitting problems, wherein we aim to find unknown parameters m that minimizes our measure of deviation, typically a norm, between the true and predicted data. We consider the following multi-experiment system

$$d_i = F_i(m) + \eta_i, i = 1, 2, \ldots, n_s$$

where $d_i \in \mathbb{C}^{n_r}$ is the measured data corresponding to the i^{th} source, $F_i(m)$ is the predicted data for the i^{th} experiment from the parameters m, η_i is the noise in the i^{th} source experiment, and n_s is the number of sources employed. We assume that the total number of source experiments is large, so that $n_s \gg 1$, and the index i can be a multi-index of varying dimensionality, depending on the experimental setup considered. The goal of this problem is to estimate model parameters m^* such that

$$m^* = \underset{m}{\operatorname{argmin}} \sum_{i=1}^{n_s} \phi(F_i(m), d_i) \tag{1.3}$$

where σ is related to the noise level and $\phi(x, y)$ is a misfit function between inputs x and y, i.e., $\phi(x, y) \geq 0$ with $\phi(x, y) = 0 \leftrightarrow x = y$, $\phi(x, y) = \phi(y, x)$. If the noise vectors η_i are identically and independently distributed (i.i.d.) with probability density function $\rho(z)$ then setting $\phi(x, y) = -\log \rho(|x - y|)$ yields the interpretation of (1.3) as an instance of maximum likelihood estimation, which is seen by taking the exponential of the negative objective in (1.3). The classical $\ell 2$ norm misfit arises from assuming a zero-mean Gaussian model on the noise vectors whereas an $\ell 1$ misfit corresponds to assuming a Laplacian distribution. More robust options are also available when the noise statistics are only partially known, such as the students-t misfit [10] and the Huber norm [114].

The challenge of implementing algorithms that solve problem (1.3) and its variants are related to usability and performance. In general, a researcher is typically interested in applying high-level algorithms with queries to the abstract structures present in (1.3). For example, one might be interested in applying an algorithm such as stochastic gradient descent [271], which relies on having algorithm-driven oracle access to the gradients with respect to the j^{th} sample, namely being able to compute $\nabla_m \phi(F_j(m), d_j)$ for arbitrary j. In this case, the details of, say, parallel data distribution, the discretization of the specific PDE describing this system, or the linear solvers used to compute solutions of the PDE, are all irrelevant to the data scientist who wants to operate on this abstract level, despite being necessary for the software to produce correct results. If, on the other hand, another researcher devel-

ops a new Krylov method or preconditioners for the PDE system, she would like to easily integrate such changes in to the overall inversion framework. Researchers should be able to "plug and play" with various components of the system without large amounts of code modification or duplication.

Simultaneously, the high computational costs of these problems merit a software framework that uses efficient operations related to solving the PDEs. For realistic industry-sized 3D problems, even the storing the model vector m becomes onerous, on the order of $\mathcal{O}(10^9)$ points. With a large number of sources, say $\mathcal{O}(10^6)$ a data volume can easily require storage and computation of $\mathcal{O}(10^{15})$ points, rendering algorithms that are designed for small 2D problems inadequate. Direct solvers become largely impractical as the fill-in becomes memory-intensive, even on a large cluster [191]. Krylov methods become the linear solution method of choice as a result, but these are challenging for the Helmholtz equation in particular, as the indefinite nature of the PDE system makes preconditioning a necessity.

1.4 Thesis Outline and Overview

The main body of this thesis comprises four chapters following the present introduction. Chapter 2 outlines the development of the framework for solving optimization problems on the manifold of low rank *Hierarchical Tucker* (HT) tensors [115]. The Hierarchical Tucker format is formed by a recursive low-rank splitting of dimensions, yielding a tensor format with a number of desirable theoretical and practical properties. Specifically, the number of parameters needed to describe a given tensor has a linear dependence on the dimension, compared to the exponential dependence storing the pointwise values of the tensor. Moreover, the set of HT tensors with ranks at most k form a closed set, unlike the CP format, which implies that there exists a best, at most $k-$ rank, approximation to a given input tensor $\mathbf{X}$. Building on the theoretical developments on the manifold structure of HT tensors from [253], we equip this manifold with a Riemannian metric and derive the subsequent gradient and Gauss-Newton Hessian expressions for solving tensor completion problems. Our methods converge quickly, owing to the explicit form of applying the Gauss-Newton Hessian inverse, and, depending on the size of the full tensor, can avoid constructing quantities of the size of the large ambient dimension. We derive expressions for a regularizer that improves recovery quality when there is very little data available and prove that our algorithms converge to a stationary point. Numerical examples showcase the benefits of this approach when interpolating realistically-sized seismic data volumes.

The results in Chapter 3 are a precursor to those in Chapter 4. We develop lemmas on the subject of the Polyak-Lojasiewicz (PL) inequality, which has been recently shown in [147] to be an important property of smooth convex functions used to prove linear convergence of gradient descent methods. Some of the results will be used in the forthcoming chapter, while others are interesting on their own right, independent from the rest of this thesis. Specifically, since the constant in the

PL-inequality determines the contraction factor for steepest descent, one aims to have as large of a constant as possible. We demonstrate that the Moreau envelope of a function decreases this parameter and that there are natural upper and lower bounds on the possible PL constant resulting from this process. Various examples will show the resulting tightness of these bounds.

We propose a new method for solving composite convex optimization problems in Chapter 4. This class of problems is quite general, encompassing robust principal component analysis, TV-regularization, and one-bit compressed sensing, to name a few examples. Our method uses a level-set approach, first explored in the SPGℓ1 algorithm [255], coupled with the variable projection method [14] to speed up convergence. As a result, we do not have to resort to optimizing non-smooth functions, which have slow convergence rates for large scale problems, and instead exploit fast solvers for solving the resulting smooth subproblems. In this chapter, we prove linear convergence of gradient descent applied to these subproblems under some general conditions by using the recent analysis of the Polyak-Lojasiewicz inequality. Coupled with a natural superlinear convergence of the outer iterations through the use of the secant method, our algorithm performs very competitively on a number of large scale problems. We showcase this technique on a variety of seismic interpolation problems, including noisy tensor completion, co-sparsity signal recovery, as well as other examples in image and audio signal processing. This method is able to efficiently handle both convex and non-convex problems.

In Chapter 5, we consider the computational aspects of solving the actual inverse problem itself. Assuming that our efforts from Chapters 2 and 4 have been successful in providing us with a high-quality representation of our fully-sampled data set as input, we focus on designing a software environment that enables us, as researchers, to quickly prototype high-level algorithms to solve these inverse problems. Our software framework deconstructs various aspects of the overall problem in to hierarchically-tiered modular components. In this fashion, our software environment is quite flexible, whereby one can easily swap out PDE stencils, preconditioners, linear solution methods, parallelization schemes, or even the PDE itself. The byproduct of making modularity a priority in our design is that integrating high performance routines into our code becomes straightforward, rather than being an afterthought as in many academic codebases. When dealing with 2D problems, we use efficient sparse-matrix routines for multiplication and division while for 3D problems, where the system matrix is much too large to be stored explicitly, we employ multi-threaded C-based matrix-vector products that construct the coefficients on-the-fly. The entire framework is agnostic to the underlying dimensionality of the problem, however, which makes applying algorithms from small test problems to realistically-sized problems a matter of changing a few lines of code. We demonstrate the effectiveness of this design by implementing a number of algorithms on 2D and 3D seismic problems and an electromagnetic inversion problem. The resulting code for each example reflects the mathematics of the underlying problem and reduces the cognitive load on the user.

Chapter 2

Low-rank Tensor Completion

2.1 Introduction

Acquiring a multidimensional signal from a real-world experiment can be a costly affair. When the signal of interest is a discretized continuous signal, there can be a number of constraints, physical or otherwise, that limit our ability to ideally sample it. For instance in the seismic case, the tensor of interest is a multidimensional wavefield in the earth's subsurface sampled at an array of receivers located at the surface. In real-world seismic experiments, budgetary constraints or environmental obstructions can limit both the total amount of time available for data acquisition as well as the number and placement of active sources and receivers. Since seismic processing, among other domains, relies on having fully sampled data for drawing accurate inferences, tensor completion is an important technique for a variety of scientific fields that acquire multidimensional data.

We consider the problem of interpolating a $d-$dimensional tensor from samples of its entries. That is, we aim to solve,

$$\min_{\mathbf{X}\in\mathcal{H}} \frac{1}{2}\|P_\Omega \mathbf{X} - b\|_2^2, \tag{2.1}$$

where P_Ω is a linear operator $P_\Omega : \mathbb{R}^{n_1\times n_2\times\ldots\times n_d} \to \mathbb{R}^m$, $b \in \mathbb{R}^m$ is our subsampled data satisfying $b = P_\Omega \mathbf{X}^*$ for some "solution" tensor $\mathbf{X}^*$ and $\mathcal{H}$ is a specific class of low-rank tensors to be specified later. Under the assumption that $\mathbf{X}^*$ is well approximated by an element in $\mathcal{H}$, our goal is to recover $\mathbf{X}^*$ by solving (2.1). For concreteness, we concern ourselves with the case when P_Ω is a restriction operator that samples the elements at the multi-indices specified by Ω i.e.,

$$P_\Omega^* P_\Omega \mathbf{X} = \begin{cases} \mathbf{X}_{i_1,i_2,\ldots,i_d} & \text{if } (i_1, i_2, \ldots, i_d) \in \Omega, \\ 0 & \text{otherwise} \end{cases}$$

and $\Omega \subset [n_1] \times [n_2] \times \cdots \times [n_d]$ is the so-called sampling set, where $[n] = \{1, \ldots, n\}$. In the above equation, we suppose that $|\Omega| = m \ll n_1 n_2 \ldots n_d$, so that P_Ω is a subsampling operator.

Unlike the matrix case, there is no unique notion of rank for tensors, as we shall see in Section (2.2). There are multiple tensor formats that generalize a particular notion of separability from the matrix case—i.e, there is no unique extension of the SVD to tensors. Although each tensor format can lead to compressible representations of their respective class of low-rank signals, the truncation of a general signal to one of these formats requires access to the fully sampled tensor $\mathbf{X}$ (or at the very least query-based access to the tensor in order to achieve reasonable accuracy [22]). This is primarily due to the use of truncated SVDs acting on various reshapings of the tensor. As in matrix completion, randomized missing entries change the behavior of the singular values and vectors of these matricizations and hence of the final approximation. In this chapter, we consider the class of Hierarchical Tucker (abbreviated HT) tensors as our low-rank tensors of interest. The set of all such tensors is a smooth, embedded submanifold of $\mathbb{R}^{n_1 \times n_2 \times \ldots \times n_d}$, first studied in [253], which we equip with a Riemannian metric. Using this Riemannian structure, we can construct optimization algorithms in order to solve (2.1) for d-dimensional tensors. We will also study some of the effects of higher dimensional sampling and extend ideas from compressive sensing and matrix completion to the HT tensor case for our specific seismic examples.

2.2 Previous Work

To provide the reader with some context on tensor representations, let us briefly detail some of the available structured tensor formats. We refer to [152, 157, 111] for a series of comprehensive overviews of structured tensor formats, and in particular to [116] for an algebraic and functional analytic point of view on the subject. In what follows, we let $N = \max_{i=1\ldots d} n_i$ be the maximum individual dimension size, $N^d := \prod_{i=1}^d n_i$ denote the dimension of the ambient space $\mathbb{R}^{n_1 \times n_2 \times \ldots \times n_d}$, and, for each tensor format discussed, K denotes the maximum of all of the rank parameters associated to that format.

The so-called Candecomp/Parafac (CP) decomposition is a very straightforward application of the separation of variables technique. Very much like the SVD of a matrix, one stipulates that, for a tensor $\mathbf{f} \in \mathbb{R}^{n_1 \times n_2 \times \ldots \times n_d}$, one can write it as

$$\mathbf{f} \approx \sum_{i=1}^{K} f_i^{(1)} \otimes f_i^{(2)} \otimes \cdots \otimes f_i^{(d)}$$

where $\otimes$ is the Kronecker product and $f_i^{(j)} \in \mathbb{R}^{n_j}$. In addition to its straightforward construction, the CP decomposition of rank K only requires dNK parameters versus the N^d of the full tensor and tensor-tensor operations can be performed efficiently

on the underlying factors rather than the full tensors themselves (see [19] for a comprehensive set of MATLAB tools).

Unfortunately, despite the parsimoniousness of the CP construction, the approximation of an arbitrary (full) tensor by CP tensors has both theoretical and numerical difficulties. In particular, the set of all CP tensors of rank at most K is not closed, and thus a best $K-$rank approximation is difficult to compute in many cases [85]. Despite this shortcoming, various authors have proposed iterative and non-iterative algorithms in the CP format for approximating full tensors [157] as well as interpolating tensors with missing data, such as the Alternating Least Squares approach (a block Gauss-Seidel type method) proposed alongside the CP format in [61] and [118], with convergence analysis in [252], and a nonlinear least-squares optimization scheme in [3]. The authors in [268] extended the Alternating Least Squares analysis to ensure that it converges globally to a stationary point of a block-convex model, which encompasses a variety of matrix and tensor completion models including the CP format.

The CP format is a specific case of the more general Tucker format, which aims to write a tensor $\mathbf{f}$ as a multilinear product

$$\mathbf{f} \approx U_1 \times_1 U_2 \times_2 \dots U_d \times_d \mathbf{C}$$

where $\mathbf{C} \in \mathbb{R}^{k_1 \times k_2 \times \dots \times k_d}$ is the so-called core tensor and the matrices $U_j \in \mathbb{R}^{n_j \times k_j}$, $j = 1, \dots, d$ are the factors of the decomposition. Here we use the notation of the multilinear product, that is, $U_i \times_i \mathbf{C}$ indicates that $\mathbf{C}$ is multiplied by U_i in dimension i, e.g., see [85, [84]]. We will elaborate on this construction in Section (2.4.2). The CP format follows from this formulation when the core tensor is diagonal, i.e., $\mathbf{C}_{i_1,i_2,i_3,\dots,i_d} = \mathbf{C}_{i_1,i_1,\dots,i_1}\delta_{i_1,i_2,\dots,i_d}$, where $\delta_{i_1,i_2,\dots,i_d} = 1$ when $i_1 = i_2 = \dots = i_d$ and 0 otherwise.

The Tucker format enjoys many benefits in terms of approximation properties over its CP counterpart. Namely, the set of all Tucker tensors of at most multilinear rank $\mathbf{k} = (k_1, k_2, \dots, k_d)$ is closed and, as a result, every tensor $\mathbf{f}$ has a best at most multilinear rank-$\mathbf{k}$ Tucker approximation. A near-optimal approximation can be computed efficiently by means of the Higher Order SVD [84]. For the tensor completion problem, the authors in [105] consider the problem of recovering a Tucker tensor with missing entries using the Douglas-Rachford splitting technique, which decouples interpolation and regularization by nuclear norm penalization of different matricizations of the tensor into subproblems that are then solved via a particular proximal mapping. An application of this approach to seismic data is detailed in [159] for the interpolation problem and [160] for denoising. Depending on the size and ranks of the tensor to be recovered, there are theoretical and numerical indications that this approach is no better than penalizing the nuclear norm in a single matricization (see [196] for a theoretical justification in the Gaussian measurement case, as well as [234] for an experimental demonstration of this effect). Some preliminary results on theoretical guarantees for recovering low-rank Tucker tensors from subsampled measurements are given in [140] for pointwise measurements and

a suitable, tensor-based incoherence condition and [185], which considers a nuclear norm penalty of the matricization of the first $d/2$ modes of $\mathbf{X}$ as opposed to a sum of nuclear norms of each of its d modes, as is typically considered.

Aside from convex relaxations of the tensor rank minimization problem, the authors in [161] develop an alternative manifold-based approach to Tucker Tensor optimization similar to our considerations for the Hierarchical Tucker case and subsequently complete such tensors with missing entries. In this case, each evaluation of the objective and Riemannian gradient requires $O(d(N+|\Omega|)K^d + dK^{d+1})$ operations, whereas our method only requires $O(dNK^2 + d|\Omega|K^3 + dK^4)$ operations. As a result of using the Hierarchical Tucker format instead of the Tucker format, our method scales much better as d, N, and K grow. The differential geometric considerations for the Tucker format were first analyzed in [174]. It is from here where the phrase Dirac Frenkel variational principle arises in the context of dynamical systems, which corresponds to the Riemannian gradient vanishing at the optimum value of the objective in an optimization context.

Hierarchical Tucker (HT) tensors were originally introduced in [115, 109], with the subclass of Tensor Train (TT) tensors developed independently in [194, 192]. These so-called tensor network decompositions have also been previously explored in the quantum physics community, see for instance [117]. TT tensors are often considered over HT tensors owing to their explicit, non-recursive formulation and relative ease of implementation for numerical methods, see for instance [193], although many of the ideas developed for TT tensors extend to the HT tensor case.

Previous work in completing tensors in the Tensor Train format includes [110, 136], wherein the authors use an alternating least-squares approach for the tensor completion problem. The derivations of the smooth manifold structure of the set of TT tensors can be found in [137]. This work builds upon the manifold structure of Hierarchical Tucker tensors studied in [253]. The authors in [97] have considered the manifold geometry of tensor networks in Banach spaces, but we will not employ such general machinery here.

Owing to its extremely efficient storage requirements (which are linear in the dimension d as opposed to exponential in d), the Hierarchical Tucker format has enjoyed a recent surge in popularity for parametrizing high-dimensional problems. The hTucker toolbox [162] contains a suite of MATLAB tools for working with tensors in the HT format, including efficient vector space operations, matrix-tensor and tensor-tensor products, and truncations of full arrays to HT format. This truncation, the so-called Hierarchical SVD developed in [109], allows one to approximate a full tensor in HT format with a near-optimal approximation error. Even though the authors in [22] develop a HT truncation method that does not need access to every entry of the tensor in order to form the HT approximation, their approach requires algorithm-driven access to the entries, which does not apply for the seismic examples we consider below. A HT approach for solving dynamical systems is outlined in [175], which considers similar manifold structure as in this article applied in a different context. The authors in [210] also consider the smooth manifold prop-

erties of HT tensors to construct tensor completion algorithms using a Hierarchical SVD-based approach. As we shall see, since their methods rely on computing SVDs of large matrices, they will have difficulty scaling to tensors with large mode sizes N, unlike the methods discussed below.

2.3 Contributions and Outline

In this chapter, we extend the primarily theoretical results of [253] to practical algorithms for solving optimization algorithms on the HT manifold. In Section (2.5.1), we introduce the Hierarchical Tucker format, wherein we restate some of the results of [253] to provide context for the Riemannian metric we introduce on the quotient manifold in Section (2.6). Equipped with this metric, we can now develop optimization methods on the HT manifold in Section (2.7) that are fast and SVD-free. For large-scale, high-dimensional problems, the computational costs of SVDs are prohibitive and affect the scalability of tensor completion methods such as [105]. Since we are using the HT manifold rather than the Tucker manifold, we avoid an exponential dependence on the internal rank parameters as in [161]. We initially proposed the idea for a Riemannian metric on the HT manifold in the conference proceedings [81] and in this chapter, we have subsequently improved upon these results to reduce the overall computational overhead and speed up the convergence of the algorithm by using our Gauss-Newton method. In Section (2.7.7), we exploit the structure of HT tensors to regularize different matricizations of the tensor without having to compute SVDs of these matricizations, lessening the effects of overfitting when there are very few samples available. We conclude by demonstrating the effectiveness of our techniques on interpolating various seismic data volumes with missing data points in all dimensions as well as missing receivers, which is more realistic. Our numerical results are similar to those presented previously in [81], but much more extensive and include our regularization and Gauss-Newton based methods. In this paper, we also compare our method to a reference implementation of [161] and achieve very reasonable results for our seismic data volumes.

We note that the algorithmic results here generalize readily to complex tensor completion $\mathbb{C}^{n_1 \times n_2 \times \cdots \times n_d}$ and more general subsampling operators P_Ω.

2.4 Notation

2.4.1 Matricization

We consider $d-$dimensional tensors $\mathbf{X}$ of size $n_1 \times n_2 \times \cdots \times n_d$. $t = (t_1, t_2, \ldots, t_k) \subset \{1, \ldots, d\}$ selects a subset of d dimensions and we denote $t^c := \{1, \ldots, d\} \setminus t$ its complement. We let the matricization of a tensor $\mathbf{X}$ along the modes $t \subset \{1, \ldots, d\}$ be the matrix $X^{(t)}$ such that the indices in t are vectorized along the rows and the

indices in t^c are vectorized along the columns, i.e., if we set $s = t^c$, then

$$X^{(t)} \in \mathbb{R}^{(n_{t_1} n_{t_2} \dots n_{t_k}) \times (n_{s_1} n_{s_2} \dots n_{s_{d-k}})}$$
$$(X^{(t)})_{(i_{t_1},\dots,i_{t_k}),(i_{s_1},\dots,i_{s_{d-k}})} := \mathbf{X}_{i_1,\dots,i_d}.$$

We also use the notation $(\cdot)_{(t)}$ for the dematricization operation, i.e., $(X^{(t)})_{(t)} = \mathbf{X}$, which reshapes the matricized version of $\mathbf{X}$ along modes t back to its full tensor form.

2.4.2 Multilinear product

A natural operation to consider on tensors is that of the multilinear product [84, 253, 109, 157].

Definition 2.1: Given a $d-$tensor $\mathbf{X}$ of size $n_1 \times n_2 \times \cdots \times n_d$ and matrices $A_i \in \mathbb{R}^{m_i \times n_i}$, the multilinear product of $\{A_i\}_{i=1}^d$ with $\mathbf{X}$, is the $m_1 \times m_2 \times ... \times m_d$ tensor $\mathbf{Y} = A_1 \times_1 A_2 \times_2 \dots A_d \times_d \mathbf{X}$, is defined in terms of the matricizations of $\mathbf{Y}$ as

$$Y^{(i)} = A_i X^{(i)} A_d^T \otimes A_{d-1}^T \otimes \dots A_{i+1}^T \otimes A_{i-1}^T \cdots \otimes A_1^T, \quad i = 1, 2, \dots, d.$$

In terms of indices, this definition is equivalent to

$$\mathbf{Y}_{i_1,\dots,i_d} = \sum_{j_1=1}^{n_1} (A_1)_{i_1,j_1} \sum_{j_2=1}^{n_2} (A_2)_{i_2,j_2} \cdots \sum_{j_{d-1}=1}^{n_{d-1}} (A_{d-1})_{i_{d-1},j_{d-1}} \sum_{j_d=1}^{n_d} (A_d)_{i_d,j_d} \mathbf{X}_{j_1,j_2,\dots,j_d}$$

Conceptually, we are applying each operator A_i to dimension i of the tensor $\mathbf{X}$, keeping all other coordinates fixed. For example, when A, X, B are matrices of appropriate sizes, the quantity AXB^T can be written as $AXB^T = A \times_1 B \times_2 X$. We remark in this instance that the ordering of the unfoldings matters and that this particular choice is compatible with the standard kronecker product. We refer to [157] for more details.

The standard Euclidean inner product between two $d-$dimensional tensors X and Y can be defined in terms of the standard Euclidean product for vectors, by letting

$$\langle \mathbf{X}, \mathbf{Y} \rangle := \mathrm{vec}(\mathbf{X})^T \mathrm{vec}(\mathbf{Y})$$

where $\mathrm{vec}(\mathbf{X}) := X^{(1,2,\dots,d)}$ is the usual vectorization operator. This inner product induces a norm $\|\mathbf{X}\|_2$ on the set of all $d-$dimensional tensors in the usual way, i.e., $\|\mathbf{X}\|_2 = \sqrt{\langle \mathbf{X}, \mathbf{X} \rangle}$.

Here we state several properties of the multilinear product, which are straightforward to prove.

Proposition 2.2: Let $\{A_i\}_{i=1}^d$, $\{B_i\}_{i=1}^d$ be collections of linear operators and $\mathbf{X}, \mathbf{Y}$ be tensors, all of appropriate sizes, so that the multilinear products below are well-defined. Then we have the following:

1. $(A_1 \times_1 \dots A_d \times_d) \circ (B_1 \times_1 \dots B_d \times_d \mathbf{X}) = (A_1 B_1) \times_1 \dots (A_d B_d) \times_d \mathbf{X}$ [85]
2. $\langle A_1 \times_1 \dots A_d \times_d \mathbf{X}, B_1 \times_1 \dots B_d \times_d \mathbf{Y} \rangle = \langle (B_1^T A_1) \times_1 \dots (B_d^T A_d) \times_d \mathbf{X}, \mathbf{Y} \rangle$

2.4.3 Tensor-tensor contraction

Another natural operation to consider between two tensors is tensor-tensor contraction, a generalization of matrix-matrix multiplication. We define tensor-tensor contraction in terms of tensors of the same dimension for ease of presentation [94].

Definition 2.3 : Given a $d-$tensor $\mathbf{X}$ of size $n_1 \times \cdots \times n_d$ and a $d-$tensor $\mathbf{Y}$ of size $m_1 \times \cdots \times m_d$, select $s, t \subset \{1, \dots, d\}$ such that $|s| = |t|$ and $n_{s_i} = m_{t_i}$ for $i = 1, \dots, |s|$. The *tensor-tensor contraction* of $\mathbf{X}$ and $\mathbf{Y}$ along modes s, t, denoted $\langle \mathbf{X}, \mathbf{Y} \rangle_{(s,t)}$, is defined as $(2d - (|s| + |t|))-$tensor Z of size (n_{s^c}, m_{t^c}), satisfying

$$\mathbf{Z} = \langle \mathbf{X}, \mathbf{Y} \rangle_{(s,t)} = (X^{(s^c)} Y^{(t)})_{(s^c),(t^c)}.$$

Tensor tensor contraction over modes s and t merely sums over the dimensions specified by s, t in $\mathbf{X}$ and $\mathbf{Y}$ respectively, leaving the dimensions s^c and t^c free.

The inner product $\langle \mathbf{X}, \mathbf{Y} \rangle$ is a special case of tensor-tensor contraction when $s = t = \{1, \dots, d\}$.

We also make use of the fact that when the index sets s, t are $s, t = [d] \setminus i$ with $\mathbf{X}$, $\mathbf{Y}$, and A_i are appropriately sized for $i = 1, \dots, d$, then

$$\begin{aligned} \langle A_1 \times_1 A_2 \times_2 \dots A_d \times_d \mathbf{X}, \mathbf{Y} \rangle_{[d]\setminus i,[d]\setminus i} = \\ A_i \langle A_1 \times_1 A_2 \times_2 \dots A_{i-1} \times_{i-1} A_{i+1} \times_{i+1} \dots A_d \times_d \mathbf{X}, \mathbf{Y} \rangle_{([d]\setminus i,[d]\setminus i)} \end{aligned} \tag{2.2}$$

i.e., applying A_i to dimension i commutes with contracting tensors over every dimension except the ith one.

2.5 Smooth Manifold Geometry of the Hierarchical Tucker Format

In this section, we review the definition of the Hierarchical Tucker format as well as previous results [253] in the smooth manifold geometry of this format. We extend these results in the next section by introducing a Riemannian metric on the space of HT parameters and subsequently derive the associated Riemannian gradient with respect to this metric. A reader familiar with the results in [253] can glance over this section quickly for a few instances of notation and move on to Section (2.6).

2.5.1 Hierarchical Tucker Format

The standard definition of the Hierarchical Tucker format relies on the notion of a dimension tree, chosen apriori, which specifies the format [109]. Intuitively, the dimension tree specifies which groups of dimensions are "separated" from other groups of dimensions, where "separation" is used in a similar sense to the SVD in two dimensions.

Definition 2.4 A dimension tree T is a non-trivial binary tree such that

- the root, $\mathrm{t}_{\mathrm{root}}$, has the label $\mathrm{t}_{\mathrm{root}} = \{1, 2, \ldots, d\}$
- for every $t \notin L$, where L is the set of leaves of T, the labels of its left and right children, t_l, t_r, form a partition of the label for t, i.e., $t_l \cup t_r = t$ and $t_l \cap t_r = \emptyset$.

We set $N(T) := T \setminus L$. An example of a dimension tree when $d = 6$ is given in Figure 2.1.

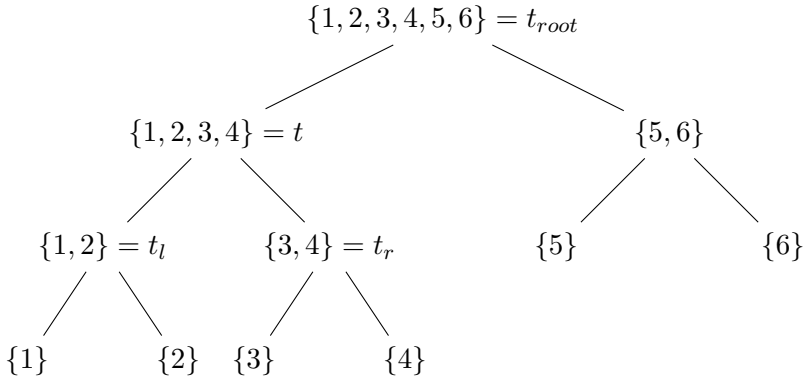


Figure 2.1 Complete dimension tree for $\{1, 2, 3, 4, 5, 6\}$.

Remark For the following derivations, we take the point of view that each quantity with a subscript $(\cdot)_t$ is associated to the node $t \in T$. By the definition of a dimension tree, for each $t \in T$, there is a corresponding subset of $\{1, \ldots, d\}$ associated to t. If our HT tensor has dimensions $n_1 \times n_2 \times ... \times n_d$, we let $n_t = \prod_{i \in t} n_i$ and, when $t \in N(T)$, n_t satisfies $n_t = n_{t_l} n_{t_r}$.

Definition 2.5 Given a dimension tree T and a vector of *hierarchical ranks* $(k_t)_{t \in T}$ with $k_t \in \mathbb{Z}^+$, $k_{\mathrm{t}_{\mathrm{root}}} = 1$, a tensor $\mathbf{X} \in \mathbb{R}^{n_1 \times n_2 \times \ldots \times n_d}$ can be written in the *Hierarchical Tucker format* if there exist parameters $x = ((U_t)_{t \in L}, (\mathbf{B}_t)_{t \in N(T)})$ such that $\phi(x) = \mathbf{X}$, where

$$\begin{aligned} \mathrm{vec}(\phi(x)) &= U_{t_l} \times_1 U_{t_r} \times_2 B_{\mathrm{t}_{\mathrm{root}}} && t = \mathrm{t}_{\mathrm{root}} \\ U_t &= (U_{t_l} \times_1 U_{t_r} \times_2 \mathbf{B}_t)^{(1,2)} && t \in N(T) \setminus \mathrm{t}_{\mathrm{root}} \end{aligned} \tag{2.3}$$

where $U_t \in \mathbb{R}_*^{n_t \times k_t}$, the set of full-rank $n_t \times k_t$ matrices, for $t \in L$ and $\mathbf{B}_t \in \mathbb{R}_*^{k_{t_l} \times k_{t_r} \times k_t}$, the set of 3-tensors of full *multilinear* rank, i.e.,

$$\operatorname{rank}(B_t^{(1)}) = k_{t_l}, \quad \operatorname{rank}(B_t^{(2)}) = k_{t_r}, \quad \operatorname{rank}(B_t^{(3)}) = k_t.$$

This technical definition can be visualized as shown in Figure 2.2. If we have a four-dimensional tensor $\mathbf{X}$, we first matricize the first two dimensions to form the matrix $X^{(1,2)}$, which can then be written in the form of a quasi-singular value decomposition. The insight of the Hierarchical Tucker format is that these quasi-singular vectors U_{12} contain information on the subspaces generated by dimensions 1 and dimensions 2 of the tensor. As such, we can reshape U_{12} in to a $n_1 \times n_2 \times k_{12}$ tensor that can further be decomposed in this multilinear fashion. We apply a similar splitting to U_{34}.

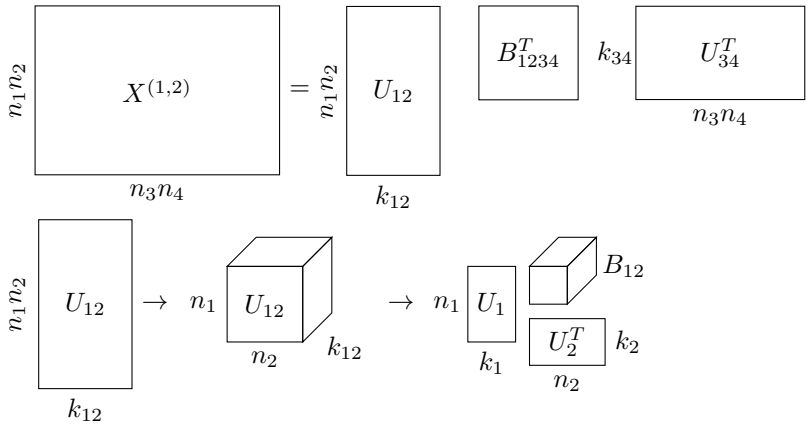


Figure 2.2 Visualizing the Hierarchical Tucker format for a 4D tensor $\mathbf{X}$ of size $n_1 \times n_2 \times n_3 \times n_4$

We say the parameters $x = (U_t, \mathbf{B}_t)$ are in *Orthogonal Hierarchical Tucker* (OHT) format if, in addition to the above construction, we also have

$$\begin{aligned} U_t^T U_t &= I_{k_t} \quad \text{for } t \in L \\ (B_t^{(1,2)})^T B_t^{(1,2)} &= I_{k_t} \quad \text{for } t \in N(T) \setminus \mathrm{t}_{\mathrm{root}} \end{aligned} \tag{2.4}$$

We have made a slight modification of the definition of the HT format compared to [253] for ease of presentation. When $d = 2$, our construction is the same as the subspace decomposition introduced in [183] for low-rank matrices, but our approach is not limited to this case.

Owing to the recursive construction in (2.3), the intermediate matrices U_t for $t \in N(T)$ do not need to be stored. Instead, specifying U_t for $t \in L$ and $\mathbf{B}_t$ for $t \in N(T)$ determines $\mathbf{X} = \phi(x)$ completely. Therefore, the overall number of parameters $x = ((U_t)_{t \in L}, (\mathbf{B}_t)_{t \in N(T)})$ is bounded above by $dNK + (d-2)K^3 + K^2$,

where $N = \max_{i=1,\dots,d} n_i$ and $K = \max_{t \in T} k_t$. When $d \geq 4$ and $K \ll N$, this quantity is much less than the N^d parameters typically needed to represent $\mathbf{X}$.

Definition 2.6 The *hierarchical rank* of a tensor $\mathbf{X} \in \mathbb{R}^{n_1 \times n_2 \times \dots \times n_d}$ corresponding to a dimension tree T is the vector $\mathbf{k} = (k_t)_{t \in T}$ where $k_{\mathrm{t_{root}}} = 1$ and for $t \in T \setminus \mathrm{t_{root}}$,

$$k_t = \operatorname{rank}(X^{(t)}).$$

We consider the set of *Hierarchical Tucker tensors of fixed rank* $\mathbf{k} = (k_t)_{t \in T}\}$, that is,

$$\mathcal{H} = \{\mathbf{X} \in \mathbb{R}^{n_1 \times n_2 \times \dots \times n_d} | \operatorname{rank}(X^{(t)}) = k_t \quad \text{for } t \in T \setminus \mathrm{t_{root}}\}.$$

We consider general HT tensors in the sequel, but we implement our algorithms with OHT parameters. In addition to significantly simplifying the resulting notation, this restriction allows us to avoid cumbersome and unnecessary matrix inversions, in particular for the resulting subspace projections in future sections. Moreover, the orthogonal format yields more accurate computations of inner products and subspace projections in finite arithmetic. This restriction does not reduce the expressibility of the HT format, however, since for any non orthogonalized parameters x such that $\mathbf{X} = \phi(x)$, there exists orthogonalized parameters x' with $\mathbf{X} = \phi(x')$ [Alg 3., 109].

We use the grouping $x = (U_t, \mathbf{B}_t)$ to denote $((U_t)_{t \in L}, (\mathbf{B}_t)_{t \in N(T)})$, as these are our independent variables of interest in this case. In order to avoid cumbersome notation, we also suppress the dependence on $(T, \mathbf{k})$ in the following, and presume a fixed dimension tree T and hierarchical ranks $\mathbf{k}$.

2.5.2 Quotient Manifold Geometry

In the interest of keeping this thesis self-contained, we briefly summarize the key results of [253] that we will use for the following sections.

Given the full-rank constraints above, the space of possible HT parameters, with $x = (U_t, \mathbf{B}_t)$, is written as

$$\mathcal{M} = \bigtimes_{t \in L} \mathbb{R}_*^{n_t \times k_t} \times \bigtimes_{t \in N(T)} \mathbb{R}_*^{k_{t_l} \times k_{t_r} \times k_t}.$$

$\mathcal{M}$ is an open submanifold of $\mathbb{R}^{\sum_{t \in L} n_t k_t + \sum_{t \in N(T)} k_{t_l} k_{t_r} k_t}$ with corresponding tangent space

$$\mathcal{T}_x \mathcal{M} = \bigtimes_{t \in L} \mathbb{R}^{n_t \times k_t} \times \bigtimes_{t \in N(T)} \mathbb{R}^{k_{t_l} \times k_{t_r} \times k_t}.$$

Let $\phi : \mathcal{M} \to \mathcal{H}$ be the parameter to tensor map in (2.3). Then for each $\mathbf{X} \in \mathcal{H}$, then there is an inherent ambiguity in its representation by parameters x, specifically if $\mathbf{X} = \phi(x) = \phi(y)$ for distinct parameters x and y, then these parameters are related in the following manner. Let $\mathcal{G}$ be the Lie group

$$\mathcal{G} = \{(A_t)_{t \in T} : A_t \in GL(k_t),\ t \neq \mathrm{t_{root}}, \quad A_{\mathrm{t_{root}}} = 1\}. \tag{2.5}$$

where $GL(p)$ is the matrix group of invertible $p \times p$ matrices and the group action of component-wise multiplication.

Let θ be the group action

$$\begin{aligned} &\theta : \mathcal{M} \times \mathcal{G} \to \mathcal{M} \\ &(x, \mathcal{A}) := ((U_t, B_t), (A_t)) \mapsto \theta_x(\mathcal{A}) := (U_t A_t, A_{t_l}^{-1} \times_1 A_{t_r}^{-1} \times_2 A_t^T \times_3 \mathbf{B}_t). \end{aligned} \tag{2.6}$$

Then $\phi(x) = \phi(y)$ if and only if there exists a unique $\mathcal{A} = (A_t)_{t \in T} \in \mathcal{G}$ such that $x = \theta_A(y)$ [Prop. 3, 253]. Therefore these are the only types of ambiguities we must consider in this format.

It follows that the orbit of x,

$$\mathcal{G}\, x = \{\theta_{\mathcal{A}}(x) : \mathcal{A} \in \mathcal{G}\},$$

is the set of all parameters that map to the same tensor $\mathbf{X} = \phi(x)$ under ϕ. This induces an equivalence relation on the set of parameters $\mathcal{M}$,

$$x \sim y \text{ if and only if } y \in \mathcal{G}\, x.$$

If we let $\mathcal{M}/\mathcal{G}$ be the corresponding quotient space of equivalence classes and $\pi : \mathcal{M} \to \mathcal{M}/\mathcal{G}$ denote the quotient map, then pushing ϕ down through π results in an injective function

$$\hat{\phi} : \mathcal{M}/\mathcal{G} \to \mathcal{H}$$

whose image is all of $\mathcal{H}$, and hence is an isomorphism (in fact, a diffeomorphism).

The vertical space, $\mathcal{V}_x\, \mathcal{M}$, is the subspace of $\mathcal{T}_x \mathcal{M}$ that is tangent to $\pi^{-1}(x)$. That is, $dx^v = (\delta U_t^v, \delta \mathbf{B}_t^v) \in \mathcal{V}_x\, \mathcal{M}$ when it is of the form [Eq. 26, 253]

$$\begin{aligned} \delta U_t^v &= U_t D_t && \text{for } t \in L \\ \delta \mathbf{B}_t^v &= D_t \times_3 \mathbf{B}_t - D_{t_l} \times_1 \mathbf{B}_t - D_{t_r} \times_2 \mathbf{B}_t && \text{for } t \in N(T) \cup \mathrm{t}_{\mathrm{root}} \\ \delta B_{\mathrm{t}_{\mathrm{root}}}^v &= -D_{t_l} B_{\mathrm{t}_{\mathrm{root}}} - B_{\mathrm{t}_{\mathrm{root}}} D_{t_r}^T && \text{for } t = \mathrm{t}_{\mathrm{root}} \end{aligned}$$

where $D_t \in \mathbb{R}^{k_t \times k_t}$. A straightforward computation shows that $D\phi(x)\big|_{\mathcal{V}_x \mathcal{M}} \equiv 0$, and therefore for every $dx^v \in \mathcal{V}_x\, \mathcal{M}$, $\phi(x) = \phi(x + dx^v)$ to first order in dx^v. From an optimization point of view, moving from the point x to $x + dx^v$, for small dx^v, will not change the current tensor $\phi(x)$ and therefore for any search direction p, we must filter out the corresponding component in $\mathcal{V}_x\, \mathcal{M}$ in order to compute the gradient correctly. We accomplish this by projecting on to a *horizontal space*, which is any complementary subspace to $\mathcal{V}_x\, \mathcal{M}$. One such choice is [Eq. 26, 253],

$$\begin{aligned} &\mathcal{H}_x\, \mathcal{M} = \{(\delta U_t^h, \delta \mathbf{B}_t^h) : \\ &\begin{cases} (\delta U_t^h)^T U_t = 0_{k_t} & \text{for } t \in L \\ (\delta B_t^{(1,2)})^T (U_{t_r}^T U_{t_r} \otimes U_{t_l}^T U_{t_l}) B_t^{(1,2)} = 0_{k_t} & \text{for } t \in N(T) \setminus \mathrm{t}_{\mathrm{root}} \end{cases} \}. \end{aligned} \tag{2.7}$$

Note that there is no restriction on $B^h_{\mathrm{t_{root}}}$, which is a matrix.

This choice has the convenient property that $\mathcal{H}_x\,\mathcal{M}$ is *invariant* under the action of θ, i.e., [Prop. 5, 253]

$$D\theta(x,\mathcal{A})[\mathcal{H}_x\,\mathcal{M},0] = \mathcal{H}_{\theta_x(\mathcal{A})}\,\mathcal{M}, \tag{2.8}$$

which we shall exploit for our upcoming discussion of a Riemannian metric.

The horizontal space $\mathcal{H}_x\,\mathcal{M}$ allows us to uniquely represent abstract tangent vectors in $T_{\pi(x)}\,\mathcal{M}/\mathcal{G}$ with concrete vectors in $\mathcal{H}_x\,\mathcal{M}$.

2.6 Riemannian Geometry of the HT Format

In this section, we introduce a Riemannian metric on the parameter space $\mathcal{M}$ that will allow us to use parameters x as representations for their equivalence class $\pi(x)$ in a well-defined manner when performing numerical optimization.

2.6.1 Riemannian metric

Since each distinct equivalence class $\pi(x)$ is uniquely identified with each distinct value of $\phi(x)$, the quotient manifold $\mathcal{M}\,/\,\mathcal{G}$ is really our manifold of interest for the purpose of computations—i.e, we would like to formulate our optimization problem over the equivalence classes $\pi(x)$. By introducing a Riemannian metric on $\mathcal{M}$ that respects its quotient structure, we can formulate concrete optimization algorithms in terms of the HT parameters without being affected by the non-uniqueness of the format—i.e., by optimizing over parameters x while implicitly performing optimization over equivalence classes $\pi(x)$. Below, we explain how to explicitly construct this Riemannian metric for the HT format.

Let $\eta_x = (\delta U_t, \delta \mathbf{B}_t), \zeta_x = (\delta V_t, \delta \mathbf{C}_t) \in \mathcal{T}_x\mathcal{M}$ be tangent vectors at the point $x = (U_t, \mathbf{B}_t) \in \mathcal{M}$. We let $M_t = U_t^T U_t$. Then we define the inner product $g_x(\cdot,\cdot)$ at x as

$$\begin{aligned} g_x(\eta_x,\zeta_x) := &\sum_{t\in L} \mathrm{tr}((M_t)^{-1}\delta U_t^T \delta V_t) \\ &+ \sum_{t\in M(T)\setminus \mathrm{t_{root}}} \langle \delta\mathbf{B}_t, (M_{t_l}) \times_1 (M_{t_r}) \times_2 (M_t)^{-1} \times_3 \delta\mathbf{C}_t\rangle \\ &+ \mathrm{tr}(M_{(\mathrm{t_{root}})_r}\delta B^T_{\mathrm{t_{root}}} M_{(\mathrm{t_{root}})_l}\delta C_{\mathrm{t_{root}}}). \end{aligned} \tag{2.9}$$

By the full-rank conditions on U_t and $\mathbf{B}_t$ at each node, by definition of the HT format, each M_t, for $t \in T$, is symmetric positive definite and varies smoothly with $x = (U_t, \mathbf{B}_t)$. As a result, g_x is a smooth, symmetric positive definite, bilinear form on $\mathcal{T}_x\mathcal{M}$, i.e., a Riemannian metric. Note that when x is in OHT, as in future sections, g_x reduces to the standard Euclidean product on the parameter space $\mathcal{T}_x\mathcal{M}$, making it straightforward to compute in this case.

Proposition 2.7 On the Riemannian manifold $(\mathcal{M}, g)$, θ defined in (2.6) acts isometrically on $\mathcal{M}$, i.e., for every $\mathcal{A} \in \mathcal{G}$, $\xi_x, \zeta_x \in \mathcal{H}_x \mathcal{M}$

$$g_x(\xi_x, \zeta_x) = g_{\theta_{\mathcal{A}}(x)}(\theta^* \xi_x, \theta^* \zeta_x)$$

where θ^* is the *push-forward* map, $\theta^* v = D\theta(x, \mathcal{A})[v]$.

Proof Let $x = (U_t, \mathbf{B}_t) \in \mathcal{M}$

$$\begin{aligned} y = (V_t, \mathbf{C}_t) &= \theta_{\mathcal{A}}(x) \\ &= (U_t A_t, A_{t_l}^{-1} \times_1 A_{t_r}^{-1} \times_2 A_t^T \times_3 \mathbf{B}_t) \end{aligned}$$

for $\mathcal{A} \in \mathcal{G}$.

If we write $\eta_x = (\delta U_t, \delta \mathbf{B}_t)$, $\zeta_x = (\delta V_t, \delta \mathbf{C}_t)$ for $\eta_x, \zeta_x \in \mathcal{H}_x \mathcal{M}$, then, by (2.8), it follows that

$$\eta_y = \theta^* \eta_x = (\delta U_t A_t, A_{t_l}^{-1} \times_1 A_{t_r}^{-1} \times_2 A_t^T \times_3 \delta \mathbf{B}_t)$$

and similarly for ζ_y.

We will compare each component of the sum of (2.9) term by term. For ease of presentation, we only consider interior nodes $t \in N(T) \setminus \mathrm{t}_{\mathrm{root}}$, as leaf nodes and the root node are handled in an analogous manner.

For $t \notin L \cup \mathrm{t}_{\mathrm{root}}$, let $\widetilde{\delta \mathbf{B}_t}$ be the component of η_y at the node t, i.e.,

$$\widetilde{\delta \mathbf{B}_t} = A_{t_l}^{-1} \times_1 A_{t_r}^{-1} \times_2 A_t^T \times_3 \delta \mathbf{B}_t$$

and similarly for $\widetilde{\delta \mathbf{C}_t}$.

The above inner product, evaluated at x, between $\delta \mathbf{B}_t$ and $\delta \mathbf{C}_t$, evaluated at x, can be written as

$$\mathrm{vec}(\delta \mathbf{B}_t)^T (M_t^{-1} \otimes M_{t_r} \otimes M_{t_l}) \mathrm{vec}(\delta \mathbf{C}_t),$$

and similarly for the inner product between $\delta \widetilde{\mathbf{B}}_t$ and $\delta \widetilde{\mathbf{C}}_t$ at y.

By setting $\tilde{M}_t := V_t^T V_t$, a quick computation shows that $\tilde{M}_t = A_t^T U_t^T U_t A_t = A_t^T M_t A_t$. Therefore, we have that the inner product between $\widetilde{\delta \mathbf{B}_t}$ and $\widetilde{\delta \mathbf{C}_t}$ at y is

$$\begin{aligned} &\mathrm{vec}(\widetilde{\delta \mathbf{B}_t})^T (\tilde{M}_t^{-1} \otimes \tilde{M}_{t_r} \otimes \tilde{M}_{t_l}) \mathrm{vec}(\widetilde{\delta \mathbf{C}_t}) \\ &= \mathrm{vec}(\delta \mathbf{B}_t)^T (A_t \otimes A_{t_r}^{-T} \otimes A_{t_l}^{-T})((A_t^{-1} M_t^{-1} A_t^{-T}) \otimes (A_{t_r}^T M_{t_r} A_{t_r}) \otimes (A_{t_l}^T M_{t_l} A_{t_l})) \\ &\quad (A_t^T \otimes A_{t_r}^{-1} \otimes A_{t_l}^{-1}) \mathrm{vec}(\delta \mathbf{C}_t) \\ &= \mathrm{vec}(\delta \mathbf{B}_t)^T (M_t^{-1} \otimes M_{t_r} \otimes M_{t_l}) \mathrm{vec}(\delta \mathbf{C}_t). \end{aligned}$$

Therefore, this term in the inner product is invariant under the group action θ and by adding the terms for each $t \in T$, we obtain that

$$g_x(\xi_x, \zeta_x) = g_{\theta_{\mathcal{A}}(x)}(\theta^* \xi_x, \theta^* \zeta_x).$$

As we are interested in carrying out our optimization using the HT parameters x as proxies for their equivalence classes $\pi(x)$, this proposition states that if we measure inner products between two tangent vectors at the point x, we obtain the same result as if we had measured the inner product between two tangent vectors transformed by $\theta_{\mathcal{A}}$ at the point $\theta_{\mathcal{A}}(x)$. In this sense, once we have a unique association of tangent vectors in $\mathcal{M}/\mathcal{G}$ with a subspace of $\mathcal{T}_x\mathcal{M}$, we can use the actual representatives, the parameters x, instead of the abstract equivalence class $\pi(x)$, in a well-defined way during our optimization.

This shows that $\mathcal{M}/\mathcal{G}$, endowed with the Riemannian metric

$$g_{\pi(x)}(\xi, \zeta) := g_x(\xi_x^h, \zeta_x^h)$$

where ξ^h, ζ_x^h are the horizontal lifts at x of ξ, ζ, respectively, is a Riemannian quotient manifold of $\mathcal{M}$ [Sec. 3.6.2, 1]

In summary, by using this Riemannian metric and restricting our optimization to only consider horizontal tangent vectors, we can implicitly formulate our algorithms on the abstract quotient space by working with the concrete HT parameters. Below, we will derive the Riemannian gradient in this context.

Remark It should be noted that although the horizontal space (2.7) is complementary to the vertical space (2.7), it is demonstrably *not* perpendicular to $\mathcal{V}_x\,\mathcal{M}$ under the Riemannian metric (2.9). Choosing a horizontal space which is perpendicular to $\mathcal{V}_x\,\mathcal{M}$ under the standard Euclidean product (i.e., (2.9) when x is orthogonalized) is beyond the scope of this work. Suffice to say, it can be done, as a generalization of the approach outlined in [183], resulting in a series of symmetry conditions on various multi-way combinations of parameters. The resulting projection operators involve solving a number of coupled Lyapunov equations, increasing with the depth of T. It remains to be seen whether such equations can be solved efficiently when d is large. We will not dwell on this point here, as we will not be needing orthogonal projections for our computations in the following. The authors in [148, 149] consider a similar approach for the Tucker tensor format.

We restrict ourselves to orthogonal parameters from this point forward, for the reasons stated previously. In order to not overburden ourselves with notation, we use the notation $\mathcal{M}$ to refer to orthogonal parameters and the corresponding group acting on $\mathcal{M}$ as $\mathcal{G}$ for the remainder of this chapter. In particular, when restricting to orthogonal HT parameters, the general linear group $GL(k_t)$ in (2.5) is replaced by $O(k_t)$, the group of orthogonal $k_t \times k_t$ matrices. The expression for the horizontal space (2.7) and the Riemannian metric are also simplified since, for orthogonal parameters, $U_t^T U_t = I_{k_t}$ for all $t \in T \setminus \mathrm{t}_{\mathrm{root}}$.

2.6.2 Riemannian gradient

The problem we are interested in solving is

$$\min_{x \in \mathcal{M}} f(\phi(x))$$

for a smooth objective function $f : \mathbb{R}^{n_1 \times n_2 \times \ldots \times n_d} \to \mathbb{R}$. We write $\hat{f} : \mathcal{M} \to \mathbb{R}$, where $\hat{f}(x) = f(\phi(x))$.

We need to derive expressions for the Riemannian gradient to update the HT parameters as part of local optimization procedures. Therefore, our primary quantity of interest is the Riemannian gradient of $\hat{f}$.

Definition 2.8 [Sec. 3.6, 1] Given a smooth scalar function $\hat{f}$ on a Riemannian manifold $\mathcal{N}$, the *Riemannian gradient* of $\hat{f}$ at $x \in \mathcal{N}$, denoted $\nabla^R \hat{f}(x)$, is the unique element of $\mathcal{T}_x\mathcal{N}$ which satisfies

$$g_x(\nabla^R \hat{f}(x), \xi) = D\hat{f}(x)\,[\xi] \quad \forall \xi \in \mathcal{T}_x\mathcal{N}$$

with respect to the Riemannian metric $g_x(\cdot, \cdot)$.

Our manifold of interest in this case is $\mathcal{N} = \mathcal{M}/\mathcal{G}$, with the corresponding horizontal space $\mathcal{H}_x\,\mathcal{M}$ in lieu of the abstract tangent space $T_{\pi(x)}\,\mathcal{M}/\mathcal{G}$. Therefore, in the above equation, we can consider the horizontal lift ξ^h of the tangent vector ξ and instead write

$$g_x(\nabla^R \hat{f}(x), \xi^h) = D\hat{f}(x)[\xi^h].$$

Our derivation is similar to that of [Sec 6.2.2, 253], except our derivations are more streamlined and cheaper computationally since we reduce the operations performed at the interior nodes $t \in N(T)$. By a slight abuse of notation in this section, we denote variational quantities associated to node t as $\delta Z_t \in \mathbb{R}^{n_{t_l} n_{t_r} \times k_t}$ and let $\delta \mathbf{Z}_t \in \mathbb{R}^{n_{t_l} \times n_{t_r} \times k_t}$ where $(\delta Z_t)_{(1,2)} = \delta \mathbf{Z}_t$ is the reshaping of δZ_t in to a $3-$tensor. The Riemannian gradient will be denoted $(\delta U_t, \delta \mathbf{B}_t)$ and a general horizontal vector will be denoted by $(\delta V_t, \delta \mathbf{C}_t)$.

When $x = (U_t, \mathbf{B}_t)$ is orthogonalized, we use $\langle \cdot, \cdot \rangle$ to denote the Euclidean inner product. By the chain rule, we have that, for any $\xi = (\delta V_t, \delta \mathbf{C}_t) \in \mathcal{H}_x\,\mathcal{M}$,

$$\begin{aligned} D\hat{f}(x)[\xi] &= Df(\phi(x))[D\phi(x)[\xi]] \\ &= \langle \nabla_{\phi(x)} f(\phi(x)), D\phi(x)[\xi] \rangle. \end{aligned}$$

Then each tensor $\delta \mathbf{V}_t \in \mathbb{R}^{n_{t_l} \times n_{t_r} \times k_t}$, with $\delta V_{t_{\text{root}}} = D\phi(x)[\xi]$, satisfies the recursion

$$\delta \mathbf{V}_t = \delta V_{t_l} \times_1 U_{t_r} \times_2 \mathbf{B}_t + U_{t_l} \times_1 \delta V_{t_r} \times_2 \mathbf{B}_t + U_{t_l} \times_1 U_{t_r} \times_2 \delta \mathbf{C}_t, \tag{2.10}$$

for matrices $\delta V_{t_l} \in \mathbb{R}^{n_{t_l} \times k_{t_l}}$, $\delta V_{t_r} \in \mathbb{R}^{n_{t_r} \times k_{t_r}}$ and tensor $\delta \mathbf{C}_t \in \mathbb{R}^{k_{t_l} \times k_{t_r} \times k_t}$ satisfying [Lemma 2, 253]

$$\delta V_{t_l}^T U_{t_l} = 0 \quad \delta V_{t_r}^T U_{t_r} = 0 \quad (\delta C_t^{(1,2)})^T B_t^{(1,2)} = 0. \tag{2.11}$$

The third orthogonality condition is omitted when $t = \mathrm{t}_{\mathrm{root}}$.

Owing to this recursive structure, we compute $\langle \delta\mathbf{U}_t, \delta\mathbf{V}_t \rangle$, where $\delta\mathbf{U}_t$ is the component of the Riemannian gradient at the current node and recursively extract the components of the Riemannian gradient associated to the children, i.e., $\delta U_{t_l}, \delta U_{t_r}$, and $\delta\mathbf{B}_t$. Here we let $\delta U_{\mathrm{t}_{\mathrm{root}}} = \nabla_{\phi(x)} f(\phi(x))$ be the Euclidean gradient of $f(\phi(x))$ at $\phi(x)$, reshaped into a matrix of size $n_{(\mathrm{t}_{\mathrm{root}})_l} \times n_{(\mathrm{t}_{\mathrm{root}})_r}$.

We set

$$\begin{aligned}
\delta U_{t_l} &= P^{\perp}_{U_{t_l}} \langle U_{t_r}^T \times_2 \delta\mathbf{U}_t, \mathbf{B}_t \rangle_{(2,3),(2,3)} \\
\delta U_{t_r} &= P^{\perp}_{U_{t_r}} \langle U_{t_l}^T \times_1 \delta\mathbf{U}_t, \mathbf{B}_t \rangle_{(1,3),(1,3)} \\
\delta\mathbf{B}_t &= U_{t_l}^T \times_1 U_{t_r}^T \times_2 \delta\mathbf{U}_t
\end{aligned}$$

followed by a projection of $\delta\mathbf{B}_t^{(1,2)}$ on to $\mathrm{span}(B_t^{(1,2)})^{\perp}$ if $t \in N(T) \setminus \mathrm{t}_{\mathrm{root}}$. Here $P_{U_t} = (I_{k_t} - U_t U_t^T)$ is the usual projection on to $\mathrm{span}(U_t)^{\perp}$. After a straightforward calculation, it follows that $\langle \delta\mathbf{U}_t, \delta\mathbf{V}_t \rangle$ is equal to

$$\langle \delta U_{t_l}, \delta V_{t_l} \rangle + \langle \delta U_{t_r}, \delta V_{t_r} \rangle + \langle \delta\mathbf{B}_t, \delta\mathbf{C}_t \rangle$$

and $\delta U_{t_l}, \delta U_{t_r}$, and $\delta\mathbf{B}_t$ satisfy (2.11). Their recursively decomposed factors will therefore be in the horizontal space $\mathcal{H}_x \mathcal{M}$.

$\delta\mathbf{B}_t$ is the component of the Riemannian gradient at node t. If t_l is a leaf node, then we have extracted the component of the Riemannian gradient associated to t_l, namely δU_{t_l}. Otherwise, we set $\delta\mathbf{U}_{t_l} = (\delta U_{t_l})_{(1,2)}$ and apply the above recursion. We make the same considerations for the right children.

In the above computations, the multilinear product operators are never formed explicitly and instead each operator is applied to various reshapings of the matrix or tensor of interest, see [93] for a reference Matlab implementation.

We make the following observations in order to minimize the number of computations performed on intermediate tensors, which can be much larger than $\dim(\mathcal{M})$. In computing the terms

$$P^{\perp}_{U_{t_l}} \langle U_{t_r}^T \times_2 \delta\mathbf{U}_t, \mathbf{B}_t \rangle_{(2,3),(2,3)},$$

we have that $\delta\mathbf{U}_t = (P^{\perp}_{U_t} \delta\tilde{U}_t)_{(1,2)}$ for a matrix $\delta\tilde{U}_t \in \mathbb{R}^{n_{t_l} n_{t_r} \times k_t}$. Using (2.2), the above expression can be written as

$$\langle P^{\perp}_{U_{t_l}} \times_1 U_{t_r}^T \times_2 (P^{\perp}_{U_t} \delta\tilde{U}_t)_{(1,2)}, \mathbf{B}_t \rangle_{(2,3),(2,3)}. \tag{2.12}$$

We note that in the above, $P^{\perp}_{U_{t_l}} \times_1 U_{t_r}^T \times_2 (P^{\perp}_{U_t} \delta\tilde{U}_t)_{(1,2)} = (U_{t_r}^T \otimes P^{\perp}_{U_{t_l}} P^{\perp}_{U_t} \delta\tilde{U}_t)_{(1,2)}$, and the operator applied to $\delta\tilde{U}_t$ satisfies

$$\begin{aligned} U_{t_r}^T \otimes P^{\perp}_{U_{t_l}} P^{\perp}_{U_t} &= U_{t_r}^T \otimes P^{\perp}_{U_{t_l}} (I_{n_t} - U_t U_t^T) \\ &= U_{t_r}^T \otimes P^{\perp}_{U_{t_l}} (I_{n_t} - U_{t_r} \otimes U_{t_l} B_t^{(1,2)} (B_t^{(1,2)})^T U_{t_r}^T \otimes U_{t_l}^T) \\ &= U_{t_r}^T \otimes P^{\perp}_{U_{t_l}}. \end{aligned}$$

This means that, using (2.2), we can write (2.12) as

$$P^{\perp}_{U_{t_l}} \langle U_{t_r}^T \times_2 (\delta\tilde{U}_t)_{(1,2)}, \mathbf{B}_t \rangle_{(2,3),(2,3)}$$

i.e., we do not have to apply $P^{\perp}_{U_t}$ to the matrix $\delta\tilde{U}_t$ at the parent node of t. Applying this observation recursively and to the other terms in the Riemannian gradient, we merely need to orthogonally project the resulting extracted parameters $(\delta U_t, \delta\mathbf{B}_t)$ on to $\mathcal{H}_x \mathcal{M}$ after applying the formula (2.12) without applying the intermediate operators $P^{\perp}_{U_t}$, reducing the overall computational costs. We summarize our algorithm for computing the Riemannian gradient in Algorithm (2.1).

Algorithm 2.1 The Riemannian gradient $\nabla^R f$ at a point $x = (U_t, \mathbf{B}_t) \in \mathcal{M}$

Input $x = (U_t, \mathbf{B}_t)$ parameter representation of the current point.
Compute $\mathbf{X} = \phi(x)$ and $\nabla_{\mathbf{X}} f(\mathbf{X})$, the Euclidean gradient of f, a $n_1 \times \dots n_d$ tensor.
$\delta U_{\mathrm{t_{root}}} \leftarrow (\nabla_{\mathbf{X}} f(\mathbf{X}))_{(1,2)}$
for $t \in N(T)$, visiting parents before their children **do**
  $\delta\mathbf{U}_t \leftarrow (\delta U_t)_{(1,2)}$
  $\delta U_{t_l} \leftarrow \langle U_{t_r}^T \times_2 \delta\mathbf{U}_t, \mathbf{B}_t \rangle_{(2,3),(2,3)}$, $\delta U_{t_r} \leftarrow \langle U_{t_l}^T \times_1 \delta\mathbf{U}_t, \mathbf{B}_t \rangle_{(1,3),(1,3)}$
  $\delta\mathbf{B}_t \leftarrow U_{t_l}^T \times_1 U_{t_r}^T \times_2 \delta\mathbf{U}_t$
  if $t \neq \mathrm{t_{root}}$ **then**
    $\delta\mathbf{B_t} \leftarrow (P^{\perp}_{B_t^{(1,2)}} (\delta B_t)^{(1,2)})_{(1,2)}$
for $t \in L$
  $\delta U_t \leftarrow P^{\perp}_{U_t} \delta U_t$
Output $\nabla^R f \leftarrow (\delta U_t, \delta\mathbf{B}_t)$

This algorithm is computing the operator $D\phi(x)^* : \mathbb{R}^{n_1 \times n_2 \times \dots \times n_d} \to \mathcal{H}_x \mathcal{M}$ applied to the Euclidean gradient $\nabla_{\phi(x)} f(\phi(x))$. The forward operator $D\phi(x) : \mathcal{H}_x \mathcal{M} \to \mathcal{T}_{\phi(x)} \mathcal{H} \subset \mathbb{R}^{n_1 \times n_2 \times \dots \times n_d}$ can be computed using a component-wise orthogonal projection $P_x^H : \mathcal{T}_x \mathcal{M} \to \mathcal{H}_x \mathcal{M}$ followed by applying (2.10) recursively.

2.6.3 Tensor Completion Objective and Gradient

In this section, we specialize the computation of the objective and Riemannian gradient in the HT format to the case where the Euclidean gradient of the objective function is sparse, in particular for tensor completion. This will allow us to scale our method to high dimensions in a straightforward fashion as opposed to the inherently

dense considerations in Algorithm (2.1). Here for simplicity, we suppose that our dimension tree T is complete, that is a full binary tree up to level $\text{depth}(T) - 1$ and all of the leaves at level $\text{depth}(T)$ are on the leftmost side of T, as in Figure 2.1. This will ease the exposition as well as allow for a more efficient implementation compared to a noncomplete tree. In what follows below, we denote $\mathbf{i} = (i_1, i_2, \ldots, i_d)$, with $1 \leq i_j \leq n_j$ for all $1 \leq j \leq d$, and let $\mathbf{i}_t$ be the subindices of $\mathbf{i}$ indexed by $t \in T$.

We consider a separable, smooth objective function on the HT manifold,

$$\hat{f}(x) = f(\phi(x)) = \sum_{\mathbf{i} \in \Omega} f_{\mathbf{i}}(\phi(x)_{\mathbf{i}}), \tag{2.13}$$

where $f_{\mathbf{i}} : \mathbb{R} \to \mathbb{R}$ is a smooth, single variable function. For the least-squares tensor completion problem, $f_{\mathbf{i}}(a) = \frac{1}{2}(a - b_{\mathbf{i}})^2$.

In this section, we also use the Matlab notation for indexing in to matrices, i.e., $A(m, n)$ is the (m, n)th entry of A, and similarly for tensors. Let $K = \max_{t \in T} k_t$.

2.6.4 Objective function

With this notation in mind, we write each entry of $P_\Omega \phi(x)$, indexed by $\mathbf{i} \in \Omega$, as

$$(P_\Omega \phi(x))(\mathbf{i}) = \sum_{r_l=1}^{k_{t_l}} \sum_{r_r=1}^{k_{t_r}} (U_{t_l})(\mathbf{i}_{t_l}, r_l) \cdot (U_{t_r})(\mathbf{i}_{t_r}, r_r) \cdot B_{\mathrm{t}_{\text{root}}}(r_l, r_r), \quad \text{where } t = \mathrm{t}_{\text{root}}\,.$$

Each entry of U_{t_l}, U_{t_r} can be computed by applying the recursive formula (2.3), i.e.,

$$U_t(\mathbf{i}_t, r) = \sum_{r_l=1}^{k_{t_l}} \sum_{r_r=1}^{k_{t_r}} (U_{t_l})(\mathbf{i}_{t_l}, r_l) \cdot (U_{t_r})(\mathbf{i}_{t_r}, r_r) \cdot \mathbf{B}_t(r_l, r_r, r)$$

with the substitutions of $t \to t_l, t_r$ as appropriate.

At each node $t \in T$, we perform at most K^3 operations and therefore the computation of $P_\Omega \phi(x)$ requires at most $2|\Omega| d K^3$ operations. The least squares objective, $\frac{1}{2}\|P_\Omega \phi(x) - b\|_2^2$, can be computed in $|\Omega|$ operations.

2.6.5 Riemannian gradient

The Riemannian gradient is more involved, notation-wise, to derive explicitly compared to the objective, so in the interest of brevity we only concentrate on the recursion for computing δU_1 below.

We let $\mathbf{Z} = \nabla_{\phi(x)} f(\phi(x))$ denote the Euclidean gradient of $f(\mathbf{X})$ evaluated at $\mathbf{X} = \phi(x)$, which has nonzero entries $\mathbf{Z}(\mathbf{i})$ indexed by $\mathbf{i} \in \Omega$. By expanding out (2.12), for each $\mathbf{i} \in \Omega$, δU_{t_l} evaluated at the root node with coordinates $\mathbf{i}_{t_l}, r_l$ for $r_l =$

$1, \ldots, k_{t_l}$ is

$$\delta U_{t_l}(\mathbf{i}_{t_l}, r_l) = \sum_{\mathbf{i}=(\mathbf{i}_{t_l}, \mathbf{i}_{t_r}) \in \Omega} \mathbf{Z}(\mathbf{i}) \sum_{r_r=1}^{k_{t_r}} U_{t_r}(\mathbf{i}_{t_r}, r_r) B_{\mathrm{t_{root}}}(r_l, r_r), \quad \text{where } t = \mathrm{t_{root}}\,.$$

For each $t \in N(T) \cup \mathrm{t_{root}}$, we let $\widetilde{\delta U_{t_l}}$ denote the length k_{t_l} vector, which depends on $\mathbf{i}_t$, satisfying, for each $\mathbf{i} \in \Omega, r_l = 1, ..., k_{t_l}$,

$$(\widetilde{\delta U_{t_l}})(\mathbf{i}_t, r_{t_l}) = \sum_{r_r=1}^{k_{t_r}} \sum_{r_t=1}^{k_t} U_{t_r}(\mathbf{i}_{t_r}, r_r) \mathbf{B}_t(r_l, r_r, r_t) \widetilde{\delta U_t}(\mathbf{i}_t, r_t).$$

This recursive construction above, as well as similar considerations for the right children for each node, yields Algorithm (2.2). For each node $t \in T \backslash \mathrm{t_{root}}$, we perform $3|\Omega|K^3$ operations and at the root where we perform $3|\Omega|K^2$ operations. The overall computation of the Riemannian gradient requires at most $6d|\Omega|K^3$ operations, when T is complete, and a negligible $O(dK)$ additional storage to store the vectors $\widetilde{\delta U_t}$ for each fixed $\mathbf{i} \in \Omega$. The computations above are followed by componentwise orthogonal projection on to $\mathcal{H}_x \mathcal{M}$, which requires $O(d(NK^2 + K^4))$ operations and are dominated by the $O(d|\Omega|K^3)$ time complexity when $|\Omega|$ is large.

Therefore for large $|\Omega|$, each evaluation of the objective, with or without the Riemannian gradient, requires $O(d|\Omega|K^3)$ operations. Since $f(\mathbf{X})$ can be written as a sum of component functions $f_{\mathbf{i}}$, each depending only on the entry $\mathbf{X}(\mathbf{i})$, we can parallelize the computation of $f(\mathbf{X})$ and the Riemannian gradient in the following way. For a system setup with p processors, we first partition the sampling set Ω in to p disjoint subsets, with the subset, Ω_j, being assigned to the jth processor. The objective (and possibly the Riemannian gradient) are computed at each processor independently using the set Ω_j and the results are added together afterwards. This "embarrassingly parallel" structure, as referred to in the parallel computing literature, allows us to scale these algorithms to large problems in a distributed environment.

In certain situations, when say $|\Omega| = pN^d$ for some $p \in [10^{-3}, 1]$ and d is sufficiently small, say $d = 4, 5$, it may be more efficient from a computer hardware point of view to use the dense linear algebra formulation in Algorithm (2.1) together with an efficient dense linear algebra library such as BLAS, rather than Algorithm (2.2). The dense formulation requires $O(N^d K)$ operations when T is a balanced tree, which may be smaller than the $O(d|\Omega|K^3)$ operations needed in this case.

Remark: By comparison, the gradient in the Tucker tensor completion case [161] requires\ $O(d(|\Omega| + N)K^d + K^{d+1})$ operations, which scales much more poorly when $d \geq 4$ compared to using Algorithm (2.2). This discrepancy is a result of the structural differences between Tucker and Hierarchical Tucker tensors, the latter of which allows one to exploit additional low-rank behaviour of the core tensor compared to the Tucker format.

Algorithm 2.2 Objective and Riemannian gradient for separable objectives

Input: $x = (U_t, \mathbf{B}_t)$ parameter representation of the current point
$f_x \leftarrow 0$, $\delta U_t, \delta \mathbf{B}_t \leftarrow 0$, $\widetilde{\delta U_t} \leftarrow 0$
for $\mathbf{i} \in \Omega$
  for $t \in N(T)$, visiting children before their parents
    for $z = 1, 2, \ldots, k_t$
      $U_t(\mathbf{i}_t, z) \leftarrow \sum_{w=1}^{k_l} \sum_{y=1}^{k_r} (U_{t_l})(\mathbf{i}_{t_l}, w) \cdot (U_{t_r})(\mathbf{i}_{t_r}, y) \cdot \mathbf{B}_t(w, y, z)$
    $f_x \leftarrow f_x + f_{\mathbf{i}}(U_{\mathrm{t_{root}}}(\mathbf{i}))$
    $\widetilde{\delta U_{\mathrm{t_{root}}}} \leftarrow \nabla f_{\mathbf{i}}(U_{\mathrm{t_{root}}}(\mathbf{i}))$
  for $t \in N(T)$, visiting parents before their children
    for $w = 1, \ldots, k_{t_l}$, $y = 1, \ldots, k_{t_r}$, $z = 1, \ldots, k_t$
      $\delta \mathbf{B}_t(w, y, z) \leftarrow \delta \mathbf{B}_t(w, y, z) + \widetilde{\delta U_t}(z) \cdot (U_{t_l})(\mathbf{i}_{t_l}, w) \cdot (U_{t_r})(\mathbf{i}_{t_r}, y)$
    for $w = 1, \ldots, k_{t_l}$
      $\widetilde{\delta U_{t_l}}(w) \leftarrow \sum_{y=1}^{k_{t_r}} \sum_{z=1}^{k_t} (U_{t_r})(\mathbf{i}_{t_r}, y) \cdot \mathbf{B}_t(w, y, z) \cdot \widetilde{\delta U_t}(z)$
    for $y = 1, \ldots, k_{t_r}$
      $\widetilde{\delta U_{t_r}}(y) \leftarrow \sum_{w=1}^{k_{t_l}} \sum_{z=1}^{k_t} (U_{t_l})(\mathbf{i}_{t_l}, w) \cdot \mathbf{B}_t(w, y, z) \cdot \widetilde{\delta U_t}(z)$
  for $t \in L$
    for $z = 1, 2, \ldots, k_t$
      $\delta U_t(\mathbf{i}_t, z) \leftarrow \delta U_t(\mathbf{i}_t, z) + \widetilde{\delta U_t}(z)$
Project $(\delta U_t, \delta \mathbf{B}_t)$ componentwise on to $\mathcal{H}_x \mathcal{M}$
Output: $f(x) \leftarrow f_x$, $\nabla^R f(x) \leftarrow (\delta U_t, \delta \mathbf{B}_t)$

2.7 Optimization

2.7.1 Reorthogonalization as a retraction

As is standard in manifold optimization, we employ a *retraction* on the tangent bundle in lieu of the exponential mapping, the former being much less computationally demanding compared to the latter.

Definition 2.9: A retraction on a manifold $\mathcal{N}$ is a smooth mapping $\mathcal{R}$ from the tangent bundle $\mathcal{T}\mathcal{N}$ onto $\mathcal{N}$ with the following properties: Let $\mathcal{R}_x$ denote the restriction of $\mathcal{R}$ to $\mathcal{T}_x\mathcal{N}$.

- $\mathcal{R}_x(0_x) = x$, where 0_x denotes the zero element of $\mathcal{T}_x\mathcal{N}$
- With the canonical identification $\mathcal{T}_{0_x}\mathcal{T}_x\mathcal{N} \simeq \mathcal{T}_x\mathcal{N}$, $\mathcal{R}_x$ satisfies

$$D\mathcal{R}_x(0_x) = \mathrm{id}_{\mathcal{T}_x\mathcal{N}}$$

where $\mathrm{id}_{\mathcal{T}_x\mathcal{N}}$ denotes the identity mapping on $\mathcal{T}_x\mathcal{N}$ (Local rigidity condition).

A retraction approximates the action of the exponential mapping to first order and hence much of the analysis for algorithms utilizing the exponential mapping

can also be carried over by the usual modifications to those using retractions. For more details, we refer the reader to [1].

A computationally feasible retraction on the HT parameters is given by QR-based reorthogonalization. The QR-based orthogonalization of (potentially nonorthogonal) parameters $x = (U_t, \mathbf{B}_t)$, denoted $QR(x)$, is given in Algorithm (2.3). Alternative retractions, such as those based on the SVD, can also be considered in a similar manner but we do not explore those options here. Since we are implicitly performing optimization on the quotient space $\mathcal{M}/\mathcal{G}$, the choice of retraction should not significantly affect algorithmic performance.

Proposition 2.10 Given $x \in \mathcal{M}$, $\eta \in \mathcal{T}_x \mathcal{M}$, let $\mathrm{QR}(x)$ be the QR-based orthogonalization defined in Algorithm (2.3). Then $\mathcal{R}_x(\eta) := \mathrm{QR}(x + \eta)$ is a retraction on $\mathcal{M}$.

Proof We first explicitly define the orthogonal parameter space and its corresponding tangent space.

Below, let $W_{k_{t_l}, k_{t_r}, k_t}$ be the closed submanifold of $\mathbb{R}_*^{k_{t_l} \times k_{t_r} \times k_t}$, the set of $3-$tensors with full multilinear rank, such that $\mathbf{W} \in W_{k_{t_l}, k_{t_r}, k_t}$ is orthonormal along modes 1 and 2, i.e., $(W^{(1,2)})^T W^{(1,2)} = I_{k_t}$ and let $\mathrm{St}(n_t, k_t)$ be the $n_t \times k_t$ Stiefel manifold of $n \times k_t$ matrices with orthonormal columns.

Our orthogonal parameter space $\mathcal{M}$ is then

$$\mathcal{M} = \bigtimes_{t \in L} \mathrm{St}(n_t, k_t) \times \bigtimes_{t \notin L \cup \mathrm{t}_{\mathrm{root}}} W_{k_{t_l}, k_{t_r}, k_t} \times \mathbb{R}_*^{k_{(\mathrm{t}_{\mathrm{root}})_l} \times k_{(\mathrm{t}_{\mathrm{root}})_r}}$$

with corresponding tangent space at $x = (U_t, \mathbf{B}_t) \in \mathcal{M}$

$$\mathcal{T}_x \mathcal{M} = \bigtimes_{t \in L} \mathcal{T}_{U_t} \mathrm{St}(n_t, k_t) \times \bigtimes_{t \notin L \cup \mathrm{t}_{\mathrm{root}}} \mathcal{T}_{\mathbf{B}_t} W_{k_{t_l}, k_{t_r}, k_t} \times \mathbb{R}^{k_{(\mathrm{t}_{\mathrm{root}})_l} \times k_{(\mathrm{t}_{\mathrm{root}})_r}}.$$

Note that $\mathcal{T}_Y \mathrm{St}(n, p) = \{Y\Omega + Y^\perp K : \Omega^T = -\Omega \in \mathbb{R}^{p \times p}, K \in \mathbb{R}^{(n-p) \times p}\}$. We omit an explicit description of $\mathcal{T}_{B_t} W_{k_{t_l}, k_{t_r}, k_t}$ for brevity.

It is easy to see that the first point in the definition of a retraction is satisfied, since for $X \in \mathrm{St}(n, p)$, $\mathrm{qf}(X) = X$

Let $x = (U_t, \mathbf{B}_t) \in \mathcal{M}$ and $\eta = (\delta U_t, \delta \mathbf{B_t}) \in \mathcal{T}_x \mathcal{M}$. To avoid notational overload, we use the slight abuse of notation that $B_t := B_t^{(1,2)}$ for $t \neq \mathrm{t}_{\mathrm{root}}$.

Let $s \in [0, t) \mapsto x(s)$ be a curve in the parameter space $\mathcal{M}$ with $x(0) = x$ and $x'(0) = \eta$ and $x(s) = (U_t(s), B_t(s))$ and $x'(s) = (\delta U_t(s), \delta B_t(s))$.

Then we have that, in Kronecker form,

$$D\mathcal{R}_x(0_x)[\eta] = \begin{cases} \frac{d}{ds} \mathrm{qf}(x(s)_t)\big|_{s=0} & \text{if } t \in L \\ \frac{d}{ds} \mathrm{qf}((R_{t_r}(s) \otimes R_{t_l}(s))(x(s)_t)\big|_{s=0} & \text{if } t \notin \mathrm{t}_{\mathrm{root}} \cup L \\ \frac{d}{ds} (R_{t_r}(s) \otimes R_{t_l}(s))(x(s)_t)\big|_{s=0} & \text{if } t = \mathrm{t}_{\mathrm{root}} \end{cases}$$

The fact that $D\mathcal{R}_x(0_x)[\eta]_t = \delta U_t$ for $t \in L$ follows from Example 8.1.5 in [1].

To compute $D\mathcal{R}_x(0_x)[\eta]_t$ for $t \notin L \cup \mathrm{t_{root}}$, we first note the formula from [1]

$$D\,\mathrm{qf}(Y)[U] = \mathrm{qf}(Y)\rho_{\mathrm{skew}}(\mathrm{qf}(Y)^T U(\mathrm{qf}(Y)^T Y)^{-1}) + (I - \mathrm{qf}(Y)\,\mathrm{qf}(Y)^T)U(\mathrm{qf}(Y)^T Y)^{-1} \tag{2.14}$$

where $Y \in \mathbb{R}_*^{n\times k}$, $U \in T_Y\mathbb{R}_*^{n\times k} \simeq \mathbb{R}^{n\times k}$ and $\mathrm{qf}(Y)$ is the Q-factor of the QR-decomposition of Y.

Therefore, if we set $Z(s) = (R_{t_r}(s) \otimes R_{t_l}(s))(x(s)_t)$, where $R_t(s)$ is the R-factor of the QR-decomposition of the matrix associated to node t, we have

$$Z'(0) = [(R'_{t_r}(0) \otimes I_{k_l}) + (I_{k_r} \otimes R'_{t_l}(0)]B_t + \delta B_t$$

As a result of the discussion in Example 8.1.5 in [1], since $R_t(0) = I_{k_t}$ we have that

$$R'_t(0) = \begin{cases} \rho_{UT}(U_t^T \delta U_t) & \text{for } t \in L \\ \rho_{UT}(B_t^T \delta B_t) & \text{for } t \notin L \cup \mathrm{t_{root}} \end{cases}$$

where $\rho_{UT}(A)$ is the projection onto the upper triangular term of the unique decomposition of a matrix into the sum of a skew-symmetric term and an upper triangular term.

Since $U_t \in \mathrm{St}(n_t, k_t)$ and $B_t \in \mathrm{St}(k_{t_l}k_{t_r}, k_t)$, since when $X \in St(n,k)$,

$$T_X St(n,k) = \{X\Omega + X^\perp K : \Omega = -\Omega^T\},$$

then $X^T\delta X$ is skew symmetric, for any tangent vector δX, which implies that $\rho_{UT}(X^T\delta X)$ is zero.

It follows that $R'_t(0) = 0$ for all $t \in T \setminus \mathrm{t_{root}}$, and therefore

$$Z'(0) = \delta B_t$$

from which we immediately obtain

$$D\mathcal{R}_x(0_x)[\eta]_t = \delta B_t \quad \text{for } t \notin L \cup \mathrm{t_{root}}$$

A similar approach holds when $t = \mathrm{t_{root}}$, and therefore, $\mathcal{R}_x(\eta)$ is a retraction on $\mathcal{M}$.

As before for the Riemannian metric, we can treat the retractions on the HT parameter space as implicitly being retractions on the quotient space as outlined below.

Since $\mathcal{R}_x(\eta)$ is a retraction on the parameter space $\mathcal{M}$, and our horizontal space is invariant under the Lie group action, by the discussion in [4.1.2 1], we have the following

Proposition 2.11 The mapping

$$\widetilde{\mathcal{R}}_{\pi(X)}(z) = \pi(\mathcal{R}_X(Z))$$

is a retraction on $\mathcal{M}/\mathcal{G}$, where $\mathcal{R}_X(Z)$ is the QR retraction previously defined on $\mathcal{M}$, $\pi : \mathcal{M} \to \mathcal{M}/\mathcal{G}$ is the projection operator, and Z is the horizontal lift at X of the tangent vector z at $\pi(X)$.

Algorithm 2.3 QR-based orthogonalization [Alg. 3 109]

Input HT parameters $x = (U_t, \mathbf{B}_t)$
Output $y = (V_t, \mathbf{C}_t)$ orthogonalized parameters such that $\phi(x) = \phi(y)$
for $t \in L$
  $Q_t R_t = U_t$, where Q_t is orthogonal and R_t is upper triangular with positive diagonal elements
  $V_t \leftarrow Q_t$
for $t \in T \setminus (L \cup \mathrm{t_{root}})$, visiting children before their parents
  $Z_t \leftarrow R_{t_l} \times_1 R_{t_r} \times_2 \mathbf{B}_t$
  $Q_t R_t = Z_t^{(1,2)}$, where Q_t is orthogonal and R_t is upper triangular
  $\mathbf{C}_t \leftarrow (Q_t)_{(1,2)}$
$C_{\mathrm{t_{root}}} \leftarrow R_{(\mathrm{t_{root}})_l} B_{\mathrm{t_{root}}} R^T_{(\mathrm{t_{root}})_r}$

2.7.2 Vector transport

The notion of vector transport allows us to relax the isometry constraints of parallel transport between tangent spaces at differing points on a manifold and decrease computational complexity without sacrificing the convergence quality of our CG-based method. Since our parameter space $\mathcal{M}$ is a subset of Euclidean space, given a point $x \in \mathcal{M}$ and a horizontal vector $\eta_x \in \mathcal{H}_x \mathcal{M}$, we take our vector transport $\mathcal{T}_{x,\eta_x} : \mathcal{H}_x \mathcal{M} \to \mathcal{H}_{R_x(\eta_x)} \mathcal{M}$ of the vector $\xi_x \in \mathcal{H}_x \mathcal{M}$ to be

$$\mathcal{T}_{x,\eta_x} \xi_x := P^h_{\mathcal{R}_x(\eta_x)} \xi_x$$

where P^h_x is the component-wise projection onto the horizontal space at x, given in (2.7). This mapping is well defined on $\mathcal{M}/\mathcal{G}$ since $\mathcal{H}_x \mathcal{M}$ is invariant under θ, and induces a vector transport on the quotient space [Sec. 8.1.4, 1].

2.7.3 Smooth optimization methods

Now that we have established the necessary components for manifold optimization, we present a number of concrete optimization algorithms for solving

$$\min_{x \in \mathcal{M}} f(\phi(x))$$

that are suitable for large-scale problems.

2.7.4 First order methods

Given the expressions for the Riemannian gradient and retraction, it is straightforward to implement the classical Steepest Descent algorithm with an Armijo line search on this Riemannian manifold, specialized from the general Riemannian manifold case [1] to the HT manifold in Algorithm (2.4). This algorithm consists of computing the Riemannian gradient, followed by a line search, HT parameter update, and a reorthogonalization. We describe the various components of Algorithm (2.4) below.

Here g_i denotes the Riemannian gradient at iteration i of the algorithm, p_i is the search direction for the optimization method, and α_i is the step length.

We choose the Polak-Ribiere approach

$$\beta_i = \frac{\langle g_i, g_i - \mathcal{T}_{x_{i-1},\alpha_{i-1}p_{i-1}}(g_{i-1})\rangle}{\langle g_{i-1}, g_{i-1}\rangle}$$

to compute the CG-parameter β_i, so that the search direction p_i satisfies

$$p_i = -g_i + \beta_i \mathcal{T}_{x_{i-1},\alpha_{i-1}p_{i-1}} p_{i-1}$$

and $p_1 = -g_1$.

2.7.5 Line search

As for any gradient based optimization scheme, we need a suitable initial step size and a computationally efficient line search. Following [230], we use a variation of the limited-minimization line search approach to set the initial step length based on the previous search direction and gradient that are vector transported to the current point—i.e, we have

$$\begin{aligned} s_i &= \mathcal{T}_{x_{i-1},\alpha_{i-1}p_{i-1}} \alpha_{i-1} p_{i-1} \\ y_i &= g_i - \mathcal{T}_{x_{i-1},\alpha_{i-1}p_{i-1}} g_{i-1}. \end{aligned}$$

In this context, s_i is the manifold analogue for the Euclidean difference between iterates, $x_i - x_{i-1}$ and y_i is the manifold analogue for the difference of gradients between iterates, $g_i - g_{i-1}$, which are standard optimization quantities in optimization algorithms set in $\mathbb{R}^n$.

Our initial step size for the direction p_i is given as

$$\alpha_0 = -g_i^T p_i / (L_i \|p_i\|_2^2)$$

where $L_i = y_i^T s_i / \|s_i\|_2^2$ is the estimate of the Lipschitz constant for the gradient [Eq. 16, 231].

In this context, computing the gradient is much more expensive than evaluating the objective. For this reason, we use a simple Armijo-type back-/forward-tracking approach that only involves function evaluations and seeks to minimize the $1D$

Algorithm 2.4 General Nonlinear Conjugate Gradient method for minimizing a function f over $\mathcal{H}$

Input Initial guess $x_0 = (U_t, \mathbf{B}_t)$, $0 < \sigma \leq 1$ sufficient decrease parameter for the Armijo line search, $0 < \theta < 1$ step size decrease parameter, $\gamma > 0$ CG restart parameter.

$p_{-1} \leftarrow 0$
$i \leftarrow 0$
for $i = 0, 1, 2, \ldots$ until convergence
  $\mathbf{X}_i \leftarrow \phi(x_i)$
  $f_i \leftarrow f(\mathbf{X}_i)$
  $g_i \leftarrow \nabla^R \hat{f}(x_i)$ (Riemannian gradient of $\hat{f}(x)$ at x_i)
  $s_i \leftarrow \mathcal{T}_{x_{i-1}, \alpha_{i-1} p_{i-1}} \alpha_{i-1} p_{i-1}$ (Vector transport the previous search direction)
  $y_i \leftarrow g_i - \mathcal{T}_{x_{i-1}, \alpha_{i-1} p_{i-1}} g_{i-1}$
  $L_i \leftarrow y_i^T s_i / \|s_i\|^2$ (Lipschitz constant estimate)
  $p_i \leftarrow -g_i + \beta_i \mathcal{T}_{x_{i-1}, \alpha_{i-1} p_{i-1}} p_{i-1}$
  if $\langle p_i, g_i \rangle > -\gamma$ (Restart CG direction)
    $p_i = -g_i$
  if $y_i^T s_i > 0$
    $\alpha \leftarrow -g_i^T p_i / (L_i \|p_i\|_2^2)$
  else
    $\alpha \leftarrow \alpha_{i-1}$
  Find $m \in \mathbb{Z}$ such that $\alpha_i = \alpha \theta^m$ and
    $f(x_i + \alpha_i p_i) - f_i \leq \sigma \alpha_i g_i^T p_i$
    $f(x_i + \alpha_i p_i) \leq \min\{f(x_i + \alpha_i \theta p_i), f(x_k + \alpha_i \theta^{-1} p_i)\}$
    (Find a quasi-optimal minimizer)
  $x_{i+1} \leftarrow \mathcal{R}_{x_i}(\alpha_i p_i)$ (Reorthogonalize)
  $i \leftarrow i + 1$

function $f(x + \alpha p_i)$ quasi-optimally, i.e., to find $m \in \mathbb{Z}$ such that $\alpha = \theta^m \alpha_0$ for $\sigma > 0$

$$\begin{aligned} f(x_i + \alpha p_i) - f(x_i) &\leq \sigma \alpha g_i^T p_i \\ f(x_i + \alpha p_i) &\leq \min\{f(x_i + \theta \alpha p_i), f(x_i + \theta^{-1} \alpha p_i)\} \end{aligned} \tag{2.15}$$

so $\alpha \approx \alpha^* = \operatorname{argmin}_\alpha f(x_i + \alpha p_i)$ in the sense that increasing or decreasing α by a factor of θ will increase $f(x_i + \alpha p_i)$. After the first few iterations of our optimization procedure, we observe empirically that our line search only involves two or three additional function evaluations to verify the second inequality in (2.15), i.e., our initial step length α_0 is quasi-optimal.

Because $\phi(\mathcal{R}_x(\alpha\eta)) = \phi(x + \alpha\eta)$ for any $x \in \mathcal{M}$ and horizontal vector η, where $\mathcal{R}_x$ is the QR retraction, Armijo linesearches do not require reorthogonalization at the intermediate steps, which further reduces computational costs. The vector $x + \alpha\eta$ is only orthogonalized once an appropriate step length α is found.

2.7.6 Gauss-Newton Method

Because of the least-squares structure of our tensor completion problem (2.1), we can approximate the Hessian by the Gauss-Newton Hessian

$$H_{GN} := D\phi^*(x)D\phi(x) : \mathcal{H}_x\,\mathcal{M} \to \mathcal{H}_x\,\mathcal{M},$$

which arises from linearizing the objective function

$$\min_{x\in\mathcal{M}} \frac{1}{2}\|\phi(x) - b\|_2^2$$

around the current point x.

Note that we do not use the "true" Gauss-Newton Hessian for the tensor completion objective (2.1), which is $D\phi^*(x)P_\Omega^*P_\Omega D\phi(x)$, since for even moderate subsampling ratios, $P_\Omega^*P_\Omega$ is close to the zero operator and this Hessian is very poorly conditioned as a result. Moreover, as we shall see, this formulation will allow us to simplify the application of H_{GN} and its inverse.

Since $D\phi(x) : \mathcal{H}_x\,\mathcal{M} \to \mathcal{T}_{\phi(x)}\mathcal{H}$ is an isomorphism, it is easy to see that H_{GN} is symmetric and positive definite on $\mathcal{H}_x\,\mathcal{M}$. The solution to the Gauss-Newton equation is then

$$H_{GN}\xi = -\nabla^R f(x)$$

for $\xi \in \mathcal{H}_x\,\mathcal{M}$.

We can simplify the computation of H_{GN} by exploiting the recursive structure of $D\phi^*(x)$ and $D\phi(x)$, thereby avoiding intermediate vectors of size $\mathbb{R}^{n_1\times n_2\times\ldots\times n_d}$ in the process. At the root note, by (2.12), we have that

$$\begin{aligned}
\delta U_{t_l}' &= P_{U_{t_l}}^{\perp}\langle U_{t_r}^T \times_2 D\phi(x)[\xi], \mathbf{B}_t\rangle_{(2,3),(2,3)},\\
\delta U_{t_r}' &= P_{U_{t_r}}^{\perp}\langle U_{t_l}^T \times_1 D\phi(x)[\xi], \mathbf{B}_t\rangle_{(1,3),(1,3)},\\
\delta B_t' &= U_{t_l}^T D\phi(x)[\xi] U_{t_r}
\end{aligned}$$

where

$$D\phi(x)[\xi] = \delta U_{t_l} \times_1 U_{t_r} \times_2 B_t + U_{t_l} \times_1 \delta U_{t_r} \times_2 B_t + U_{t_l} \times_1 U_{t_r} \times_2 \delta B_t, \quad t = \mathrm{t}_{\mathrm{root}}\,.$$

In the above expression, $D\phi(x)$ is horizontal, so that for each $t \in T \setminus \mathrm{t}_{\mathrm{root}}$, δU_t is perpendicular to U_t (2.11). A straightforward computation simplifies the above expression to

$$\delta U_{t_l}' = \delta U_{t_l} G_{t_l}, \quad \delta U_{t_r}' = \delta U_{t_r} G_{t_r}, \quad \delta B_t' = \delta B_{\mathrm{t}_{\mathrm{root}}}.$$

where $G_{t_l} = B_{\mathrm{t}_{\mathrm{root}}} B_{\mathrm{t}_{\mathrm{root}}}^T$ and $G_{t_r} = B_{\mathrm{t}_{\mathrm{root}}}^T B_{\mathrm{t}_{\mathrm{root}}}$.

This expression gives us the components of the horizontal vector $\delta U_{t_l}', \delta U_{t_r}'$ sent to the left and right children, respectively, as well as the horizontal vector $\delta B_t'$.

We proceed recursively by considering a node $t \in N(T) \cup \mathrm{t}_{\mathrm{root}}$ and let $\delta U_t G_t$ be the contribution from the parent node of t. By applying the adjoint partial derivatives, followed by an orthogonal projection on to $\mathcal{H}_x \mathcal{M}$, we arrive at a simplified form for the Gauss-Newton Hessian

$$\begin{aligned} P^{\perp}_{U_{t_l}} \langle U_{t_r}^T \times_2 \delta U_t G_t, \mathbf{B}_t \rangle_{(2,3),(2,3)} &= \langle \delta U_{t_l} \times_1 G_t \times_3 \mathbf{B}_t, \mathbf{B}_t \rangle_{(2,3),(2,3)} \\ &:= \delta U_{t_l} G_{t_l}, \\ P^{\perp}_{U_{t_r}} \langle U_{t_l}^T \times_1 \delta U_t G_t, \mathbf{B}_t \rangle_{(1,3),(1,3)} &= \delta U_{t_r} G_{t_r}, \\ P^{\perp}_{B_t^{(1,2)}} (U_{t_l}^T \times_1 U_{t_r}^T \times_2 G_t \times_3 \delta U_t) &= \delta G_t \times_3 \mathbf{B}_t. \end{aligned}$$

In these expressions, the matrices G_t are the Gramian matrices associated to the HT format, initially introduced in [109] and used for truncation of a general tensor to the HT format as in [248]. They satisfy, for $x = (U_t, \mathbf{B}_t) \in \mathcal{M}$, $G_{\mathrm{t}_{\mathrm{root}}} = 1$ and

$$G_{t_l} = \langle G_t \times_3 \mathbf{B}_t, \mathbf{B}_t \rangle_{(2,3),(2,3)}, \quad G_{t_r} = \langle G_t \times_3 \mathbf{B}_t, \mathbf{B}_t \rangle_{(1,3),(1,3)}, \tag{2.16}$$

i.e., the same recursion as G_t in the above derivations. Each G_t is a $k_t \times k_t$ symmetric positive definite matrix (owing to the full rank constraints of the HT format) and also satisfies

$$\lambda_j(G_t) = \sigma_j(X^{(t)})^2 \tag{2.17}$$

where $\lambda_j(A)$ is the jth eigenvalue of the matrix A and $\sigma_j(A)$ is the jth singular value of A.

Assuming that each G_t is well conditioned, applying the inverse of H_{GN} follows directly, summarized in Algorithm (2.5).

Algorithm 2.5 The inverse Gauss-Newton Hessian applied to a vector

Input Current point $x = (U_t, \mathbf{B}_t)$, horizontal vector $\zeta = (\delta U_t, \delta \mathbf{B}_t)$
Compute $(G_t)_{t \in T}$ using (2.16)
for $t \in T \setminus \mathrm{t}_{\mathrm{root}}$
  if $t \in L$
    $\widetilde{\delta U_t} \leftarrow \delta U_t G_t^{-1}$
  else
    $\widetilde{\delta \mathbf{B}_t} \leftarrow G_t^{-1} \times_3 \delta \mathbf{B}_t$
$H_{GN}^{-1} \zeta \leftarrow (\widetilde{\delta U_t}, \widetilde{\delta \mathbf{B}_t})$

For the case where our solution HT tensor exhibits quickly-decaying singular values of the matricizations, as is typically the assumption on the underlying tensor, the Gauss-Newton Hessian becomes poorly conditioned as the iterates converge to the solution, owing to (2.17). This can be remedied byintroducing a small $\epsilon > 0$ and applying $(G_t + \epsilon I)^{-1}$ instead of G_t^{-1} in Algorithm 2.5 or by applying H_{GN}^{-1} by applying H_{GN} in a truncated PCG method. For efficiency purposes, we find the former option preferable. Alternatively, we can also avoid ill-conditioning via regularization, as we will see in the next section.

Remark: We note that applying the inverse Gauss-Newton Hessian to a tangent vector is akin to ensuring that the projection on to the horizontal space is orthogonal, as in [6.2.2, 253]. Using this method, however, is much faster than the previously proposed method, because applying Algorithm 2.5 only involves matrix-matrix operations on the small parameters, as opposed to operations on much larger intermediate matrices that live in the spaces between the full tensor space and the parameter space.

2.7.7 Regularization

In the tensor completion case, when there is little data available, interpolating on the HT manifold is susceptible to overfitting if one chooses the ranks $(k_t)_{t\in T}$ for the interpolated tensor too high. In that case, one can converge to solutions in $\text{null}(P_\Omega)$ that try leave the current manifold, associated to the ranks $(k_t)_{t\in T}$, to another nearby manifold corresponding to higher ranks. This can lead to degraded results in practice, as the actual ranks for the solution tensor are almost always unknown. One can use cross-validation techniques to estimate the proper internal ranks of the tensor, but we still need to ensure that the solution tensor has the predicted ranks for this approach to be successful – i.e., the iterates x must stay away from the boundary of $\mathcal{H}$.

To avoid our HT iterates converging to the manifold boundary, we introduce a regularization term on the singular values of the HT tensor $\phi(x) = \mathbf{X}$. To accomplish this, we exploit the hierarchical structure of $\mathbf{X}$ and specifically the property of the Gramian matrices G_t in 2.17 to ensure that *all* matricizations of $\mathbf{X}$ remain well-conditioned *without* having to perform SVDs on each matricization $X^{(t)}$. The latter approach would be prohibitively expensive when d or N are even moderately large.

Instead, we penalize the growth of the Frobenius norm of $X^{(t)}$ and $(X^{(t)})^\dagger$, which indirectly controls the largest and smallest singular values of $X^{(t)}$. We implement this regularization via the Gramian matrices in the following way.

From (2.17), it follows that $\text{tr}(G_t) = \|G_t\|_* = \|X^{(t)}\|_F^2$ and likewise $\text{tr}(G_t^{-1}) = \|G_t^{-1}\|_* = \|(X^{(t)})^\dagger\|_F^2$.

Our regularizer is then

$$H((\mathbf{B}_{t'})_{t'\in T}) = \sum_{t\in T} \text{tr}(G_t) + \text{tr}(G_t^{-1}).$$

A straightforward calculation shows that for $\mathcal{A} \in \mathcal{G}$

$$(G_t)_{t\in T,x} = (A_t^T G_t A_t)_{t\in T,\theta_{\mathcal{A}}(x)}$$

for A_t orthogonal. Therefore, our regularizer R is well-defined on the quotient manifold in the sense that it is $\theta-$invariant on the parameter space $\mathcal{M}$. This is the same regularization term considered in [161], used for (theoretically) preventing the iterate from approaching the boundary of $\mathcal{H}$. In our case, we can leverage the

structure of the Gramian matrices to implement this regularizer in a computationally feasible way.

Since in the definition of the Gramian matrices (2.16), G_t is computed recursively via tensor-tensor contractions (which are smooth operations), it follows that the mapping $g : (\mathbf{B}_t)_{t\in N(T)} \to (G_t)_{t\in T}$ is smooth. In order to compute its derivatives, we consider a node $t \in T \setminus \mathrm{t}_{\mathrm{root}}$ and consider the variations of its left and right children, i.e.,

$$\delta G_{t_r} = \frac{\partial G_{t_r}}{\partial \mathbf{B}_t}\delta\mathbf{B}_t + \frac{\partial G_{t_r}}{\partial G_t}\delta G_t, \qquad \delta G_{t_l} = \frac{\partial G_{t_l}}{\partial \mathbf{B}_t}\delta\mathbf{B}_t + \frac{\partial G_{t_l}}{\partial G_t}\delta G_t. \tag{2.18}$$

We can take the adjoint of this recursive formulation, and thus obtain the gradient of g, if we compute the adjoint partial derivatives in (2.18) as well as taking the adjoint of the recursion itself. To visualize this process, we consider the relationship between input variables and output variables in the recursion as a series of small directed graphs, shown in Figure 2.3.

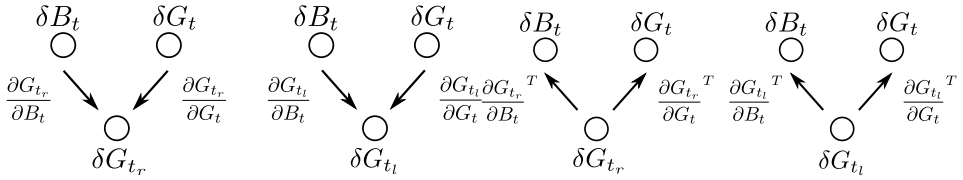


Figure 2.3 Forward and adjoint depictions of the Gramian mapping differential

These graphs can be understood in the context of Algorithmic Differentiation, whereby the forward mode of this derivative map propagates variables up the tree and the adjoint mode propagates variables down the tree and adds (accumulates) the contributions of the relevant variables.

Since we only consider tangent vectors δB_t that are in the horizontal space at x, each extracted component is projected on to $(B_t^{(1,2)})^{\perp}$. We summarize our results in the following algorithms.

Algorithm 2.6 $Dg[\delta\mathbf{B}_t]$

Input Current point $x = (U_t, \mathbf{B}_t)$, horizontal vector $dx = (\delta U_t, \delta\mathbf{B}_t)$.
Compute $(G_t)_{t\in T}$ using (2.16)
$\delta G_{\mathrm{t}_{\mathrm{root}}} \leftarrow 0$
for $t \in N(T)$, visiting parents before children
$\delta G_{t_l} \leftarrow \langle \delta G_t \times_3 \mathbf{B}_t, \mathbf{B}_t\rangle_{(2,3),(2,3)} + 2\langle G_t \times_3 \delta\mathbf{B}_t, \mathbf{B}_t\rangle_{(2,3),(2,3)}$
$\delta G_{t_r} \leftarrow \langle \delta G_t \times_3 \mathbf{B}_t, \mathbf{B}_t\rangle_{(1,3),(1,3)} + 2\langle G_t \times_3 \delta\mathbf{B}_\mathbf{t}, \mathbf{B}_t\rangle_{(1,3),(1,3)}$
Output $Dg[\delta\mathbf{B}_t] \leftarrow (\delta G_t)_{t\in T}$

Algorithm 2.7 $Dg^*[\delta G_t]$

Input Current point $x = (U_t, \mathbf{B}_t)$, Gramian variations $(\delta G_t)_{t\in T}$, $\delta G_{\mathrm{t_{root}}} = 0$.
Compute $(G_t)_{t\in T}$ using (2.16)
for $t \in T$
  $\widetilde{\delta G_t} \leftarrow \delta G_t$
for $t \in N(T)$, visiting children before parents
  $\delta \mathbf{B}_t \leftarrow (\widetilde{\delta G_{t_l}} + \widetilde{\delta G_{t_l}}^T) \times_1 G_t \times_3 \mathbf{B}_t + (\widetilde{\delta G_{t_r}} + \widetilde{\delta G_{t_r}}^T) \times_2 G_t \times_3 \mathbf{B}_t$
  if $t \neq \mathrm{t_{root}}$
    $\delta \mathbf{B}_t \leftarrow (P^{\perp}_{B^{(1,2)}} \delta \mathbf{B}_t^{(1,2)})_{(1,2)}$
    $\widetilde{\delta G_t} \leftarrow \delta G_t + \langle \widetilde{G_{t_l}} \times_1 \mathbf{B}_t, \mathbf{B}_t \rangle_{(1,2),(1,2)} + \langle \widetilde{G_{t_r}} \times_2 \mathbf{B}_t, \mathbf{B}_t \rangle_{(1,2),(1,2)}$
Output $Dg^*[\delta G_t] \leftarrow (\delta \mathbf{B}_t)_{t\in T}$

Applying Algorithm (2.7) to the gradient of $H(\mathbf{B}_t)$,

$$\nabla H(\mathbf{B}_t) = (V_t(I_{k_t} - S_t^{-2})V_t^T),$$

where $G_t = V_t S_t V_t^T$ is the eigenvalue decomposition of G_t, yields the Riemannian gradient of the regularizer. Note that here, we avoid having to compute SVDs of *any* matricizations of the full data $\phi(x)$, resulting in a method which is much faster than other tensor completion methods that require the SVDs on tensors in $\mathbb{R}^{n_1 \times n_2 \times \ldots \times n_d}$ [105]. Note that the cost of computing this regularizer $H(\mathbf{B}_t)$ and its gradient are almost negligible compared to the cost of computing the objective and its Riemannian gradient.

Finally, we should also note that the use of this regularizer is not designed to improve the recovery quality of problem instances with a relatively large amount of data and is useful primarily in the case where there is very little data so as to prevent overfitting, as we shall see in the numerical results section.

2.7.8 Convergence analysis

Our analysis here follows from similar considerations in [Sec. 3.6 , 161].

Theorem 2.12 Let $\{x_i\}$ be an infinite sequence of iterates, with x_i generated at iteration i, generated from (2.4) for the Gramian-regularized objective with $\lambda > 0$

$$f(x) = \frac{1}{2}\|P_\Omega \phi(x) - b\|_2^2 + \lambda^2 \sum_{t \in T \setminus \mathrm{t_{root}}} \mathrm{tr}(G_t(x)) + \mathrm{tr}(G_t^{-1}(x)).$$

Then $\lim_{i\to\infty} \|\nabla^R f(x_i)\| = 0$.

Proof To show convergence, we merely need to show that the iterates remain in a sequentially compact set, since any accumulation point of $\{x_i\}$ is a critical point of f, by [Theorem 4.3.1, 1]. But this follows because by construction, since

$f(x_i) \leq f(x_0) := C^2$ for all i. Letting $\mathbf{X}_i := \phi(x_i)$

$$\frac{1}{2}\|P_\Omega \phi(x_i) - b\|_2^2 + \lambda^2 \sum_{t \in T \setminus \mathrm{t}_{\mathrm{root}}} \mathrm{tr}(G_t(x_i)) + \mathrm{tr}(G_t^{-1}(x_i)) =$$
$$\frac{1}{2}\|P_\Omega \mathbf{X}_i - b\|_2^2 + \lambda^2 \sum_{t \in T \setminus \mathrm{t}_{\mathrm{root}}} \|X_i^{(t)}\|_F^2 + \|(X_i^{(t)})^\dagger\|_F^2 \leq C^2$$

This shows, in particular, that

$$\lambda^2 \sum_{t \in T \setminus \mathrm{t}_{\mathrm{root}}} \|X_i^{(t)}\|_F^2 \leq C^2 \quad \lambda^2 \sum_{t \in T \setminus \mathrm{t}_{\mathrm{root}}} \|(X_i^{(t)})^\dagger\|_F^2 \leq C^2$$

and therefore we have upper and lower bounds on the maximum and minimum singular values of $X_i^{(t)}$

$$\sigma_{\max}(X_i^{(t)}) \leq \|X_i^{(t)}\|_F \leq C/\lambda \quad \sigma_{\min}^{-1}(X_i^{(t)}) \leq \|(X_i^{(t)})^\dagger\|_F \leq C/\lambda$$

and therefore the iterates X_i stay within the compact set

$$\mathcal{C} = \{\mathbf{X} \in \mathcal{H} : \sigma_{\min}(X_k^{(t)}) \geq \lambda/C, \sigma_{\max}(X_k^{(t)}) \leq C/\lambda, t \in T \setminus \mathrm{t}_{\mathrm{root}}\}.$$

One can show, as a modification of the proof in [253], that $\hat{\phi} : \mathcal{M}/\mathcal{G} \to \mathcal{H}$ is a homeomorphism on to its image, so that $\hat{\phi}^{-1}(C)$ is compact in $\mathcal{M}/\mathcal{G}$.

We let $\|\cdot\|_T$ be the natural metric on $\mathcal{M}$, which is to say

$$d(x, y) = \|x - y\|_T = \sum_{t \in L} \|U_t - V_t\|_F + \sum_{t \in N(T)} \|\mathbf{B}_t - \mathbf{C}_t\|_F.$$

where $x = (U_t, \mathbf{B}_t)$, $y = (V_t, \mathbf{C}_t) \in \mathcal{M}$.

A metric on $\mathcal{M}/\mathcal{G}$ that generates the topology on the quotient space, is

$$\tilde{d}(\pi(x), \pi(y)) = \inf_{\mathcal{A}, \mathcal{B} \in \mathcal{G}} d(\theta_{\mathcal{A}}(x), \theta_{\mathcal{B}}(y)). \tag{2.19}$$

Note that this pseudo-metric is a metric which generates the topology on $\mathcal{M}/\mathcal{G}$ by [Theorem 2.1, 45] since $\{\theta_{\mathcal{A}}\}_{\mathcal{A} \in \mathcal{G}}$ is a group of isometries acting on $\mathcal{M}$ and the orbits of the action are closed by [Lemma 1, 253]. Note that this metric is equivalent to

$$\tilde{d}(\pi(x), \pi(y)) = \inf_{\mathcal{A} \in \mathcal{G}} d(x, \theta_{\mathcal{A}}(y)), \tag{2.20}$$

which is well-defined and equal to (2.19) since $\|\theta_{\mathcal{A}}(x) - \theta_{\mathcal{B}}(y)\|_T = \|x - \theta_{\mathcal{A}^{-1}\mathcal{B}}(y)\|_T$ and $\mathcal{A}, \mathcal{B}$ vary over $\mathcal{G}$.

Let $\{x_i\}$ be a sequence in $\pi^{-1}(\hat{\phi}^{-1}(C))$. By the compactness of $\phi^{-1}(C)$ in $\mathcal{M}/\mathcal{G}$, we have that there is some set of HT parameters $y \in \phi^{-1}(C)$ such that, without loss

of generality,

$$\tilde{d}(\pi(x_i), \pi(y)) \to 0 \text{ as } i \to \infty.$$

Then, by the characterization (2.20), since $\mathcal{G}$ is compact, there exists a sequence $\mathcal{A}_i \subset \mathcal{G}$ such that

$$d(x_i, \theta_{\mathcal{A}_i}(y)) \to 0$$

Also by the compactness of $\mathcal{G}$, there exists a subsequence $\{\mathcal{A}_{i_j}\}$ that converges to $\mathcal{A} \in \mathcal{G}$, so $d(\theta_{\mathcal{A}_{i_j}}(y), \theta_{\mathcal{A}}(y))$ converges to 0 by continuity of the Lie group action θ. It then follows that

$$d(x_{i_j}, \theta_{\mathcal{A}}(y)) \leq d(x_{i_j}, \theta_{\mathcal{A}_{i_j}}(y)) + d(\theta_{\mathcal{A}_{i_j}}(y), \theta_{\mathcal{A}}(y)) \to 0 \quad \text{as } j \to \infty$$

And so any sequence $\{x_i\} \in \pi^{-1}(\hat{\phi}^{-1}(C))$ has a convergent subsequence and so $\pi^{-1}(\hat{\phi}^{-1}(C))$ is sequentially compact in $\mathcal{M}$. Therefore since the sequence x_k generated by Algorithm (2.4) remains in $\pi^{-1}(\hat{\phi}^{-1}(C))$ for all i, a subsequence of x_i converges to some $x \in \pi^{-1}(\hat{\phi}^{-1}(C))$, and so x is a critical point of f.

Although we have only shown convergence to a limit point where the gradient vanishes, the authors in [225] consider the convergence of general line search methods that operate on the affine variety of rank at most k matrices. It would be an interesting extension of these results to the tensor case, but we leave this for future work. We also remark that this proof also holds for the Gauss-Newton method as well, since the results in [Theorem 4.3.1, 1] simply require that the search direction has a negative inner product with the gradient in the limit, which can be shown in this case.

2.8 Numerical Examples

To address the challenges of large-scale tensor completion problems, as encountered in exploration seismology, we implemented the approach outlined in this chapter in a highly optimized parallel Matlab toolbox entitled HTOpt (available at http://www.math.ubc.ca/~curtd/software.html for academic use). Contrary to the HT toolbox [162], whose primary function is performing operations on known HT tensors, our toolbox is designed to solve optimization problems in the HT format such as the seismic tensor completion problem. Our package includes the general optimization on HT manifolds detailed in Algorithm (2.1) as well as sparsity-exploiting objective and Riemannian gradient in Algorithm (2.2), implemented in Matlab. We also include a parallel implementation using the Parallel Matlab toolbox for both of these algorithms. All of the following experiments were run on a single IBM x3550 workstation with 2 quad-core Intel 2.6Ghz processors with 16GB of RAM running Linux 2.6.18.

In this section, we compare our Gauss-Newton method with the interpolation scheme detailed in [161], denoted geomCG, for interpolating tensors with missing entries on the Tucker manifold. We have implemented a completely Matlab-based

version of geomCG, which does not take advantage of the sparsity of the residual when computing the objective and Riemannian gradient, but uses Matlab's internal calls to LAPACK libraries to compute matrix-matrix products and is much more efficient for this problem. To demonstrate this speedup, we compare our method to the reference mex implementation for geomCG from [161] on a randomly generated $200 \times 200 \times 200$ with multilinear rank 40 in each dimension, a training set comprised of 10% of the data points chosen randomly, and run for 20 iterations. The results of this comparison are shown in Table 2.1. We restrict our comparison to 3D tensors since the implementation of [161] available on the author's website is strictly for 3D tensors.

Method	Time (s)	Training error	Test error	Difference
Original [161]	4701s	$1.091 \cdot 10^{-3}$	$2.792 \cdot 10^{-3}$	N/A
Ours	96.1s	$1.129 \cdot 10^{-3}$	$2.384 \cdot 10^{-3}$	$7.956 \cdot 10^{-4}$

Table 2.1 Comparison between geomCG implementations on a $200 \times 200 \times 200$ random Gaussian tensor with multilinear rank 40 and subsampling factor $= 0.1$, both run for 20 iterations. Quantities are relative to the respective underlying solution tensor.

Since we take advantage of dense, multithreaded linear algebra routines, we find that our Matlab implementation is significantly faster than the mex code of [161] when $K \geq 20$ and $|\Omega|$ is a significant fraction of N^d, as is the case in the examples below.

In the examples considered below, we consider fixed multilinear ranks for each problem. There have been rank increasing strategies proposed for matrix completion in, among other places, [254, 184, 32] and as previously mentioned in [161], but we leave the question as to how to incorporate these heuristics in to the HT optimization framework for future research.

2.8.1 Seismic data

We briefly summarize the structure of seismic data in this section. Seismic data is typically collected via a boat equipped with an airgun and, for our purposes, a 2D array of receivers positioned on the ocean floor. The boat periodically fires a pressure wave in to the earth, which reflects off of subterranean discontinuities and produces a returning wave that is measured at the receiver array. The resulting data volume is five-dimensional, with two spatial source coordinates, denoted x_{src}, y_{src}, two receiver coordinates, denoted x_{rec}, y_{rec}, and time. For these experiments, we take a Fourier transform along the time axis and extract a single 4D volume by fixing a frequency and let $\mathbf{D}$ denote the resulting frequency slice with dimensions $n_{src} \times n_{src} \times n_{rec} \times n_{rec}$.

From a practical point of view, the acquisition of seismic data from a physical system only allows us to subsample receiver coordinates, i.e., $\Omega = [n_{src}]\times[n_{src}]\times\mathcal{I}$ for some $\mathcal{I} \subset [n_{rec}] \times [n_{rec}]$ with $|\mathcal{I}| < n_{rec}^2$, rather than the standard tensor completion approach, which assumes that $\Omega \subset [n_{src}] \times [n_{src}] \times [n_{rec}] \times [n_{rec}]$ is random and unstructured. As a result, we use the dimension tree in Figure 2.4 for completing seismic data.

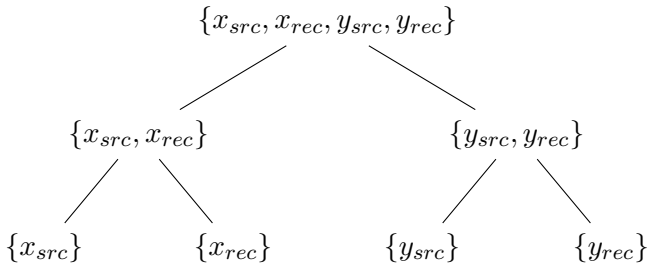


Figure 2.4 Dimension tree for seismic data

With this choice, the fully sampled data $\mathbf{D}$ has quickly decaying singular values in each matricization $D^{(t)}$ and is therefore represented well in the HT format. Additionally, the subsampled data $P_\Omega\mathbf{D}$ has increased singular values in all matricizations, and is poorly represented as a HT tensor with fixed ranks $\mathbf{k}$ as a result. We examine this effect empirically in [81] and note that this data organization is used in [87] to promote low rank of the solution operators of the wave equation with respect to source and receiver coordinates. In a noise-free environment, seismic data volumes are modelled as the restriction of Green's functions of the wave equation to the acquisition surface, which explains why this particular coordinate grouping decreases the rank of seismic data. Moreover, since we restrict our attention to relatively low frequency data, these volumes are relatively smooth, resulting in quickly decaying singular values [224].

Although this approach is limited to considerations of seismic data, for larger dimensions/different domains, potentially the method of [21] can choose an appropriate dimension tree automatically. In the next section, we also include the case when $\Omega \subset [n_{src}] \times [n_{src}] \times [n_{rec}] \times [n_{rec}]$, i.e. the "missing points" scenario, to demonstrate the added difficulty of the "missing receivers" case described above.

2.8.2 Single reflector data

For this data set, we generate data from a very simple seismic model consisting of two horizontal layers with a moderate difference in wavespeed and density between them. We generate this data with $n_{src} = n_{rec} = 50$ and extract a frequency slice at 4.21Hz, rescaled to have unit norm.

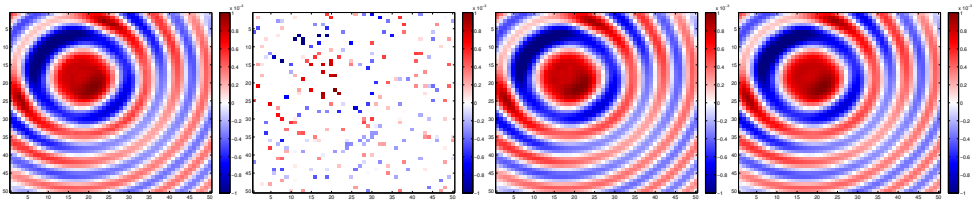
We consider the two sampling scenarios discussed in the previous section: we remove random points from the tensor, with results shown in Figure 2.5, and we

remove random receivers from the tensor, with results shown in Figure 2.6. Here geomCG(r_{leaf}) - w denote the Tucker interpolation algorithm with rank r_{leaf} in each mode and w rank continuation steps, i.e., the approach proposed in [161]. We also let HT($r_{\text{leaf}}, r_{x_{\text{src}}x_{\text{rec}}}$) denote the HT interpolation method with rank r_{leaf} as in the Tucker interpolation and rank $r_{x_{\text{src}}x_{\text{rec}}}$ as the internal rank for the dimension tree. As is customary in the seismic literature, we measure recovery quality in terms of SNR, namely

$$\text{SNR}(\mathbf{X}, \mathbf{D}) = -20 \log_{10} \left(\frac{\|\mathbf{X}_{\Omega^c} - \mathbf{D}_{\Omega^c}\|}{\|\mathbf{D}_{\Omega^c}\|} \right) \text{dB},$$

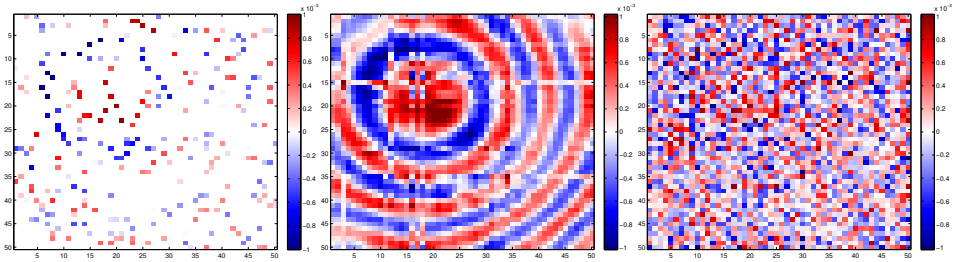
where $\mathbf{X}$ is our interpolated signal, $\mathbf{D}$ is our reference solution, and $\Omega^c = [n_{src}] \times [n_{src}] \times [n_{rec}] \times [n_{rec}] \setminus \Omega$. As we can see in Figure 2.5, the HT formulation is able to take advantage of low-rank separability of the seismic volume to produce a much higher quality solution than that of the Tucker tensor completion. The rank continuation scheme does not seem to be improving the recovery quality of the Tucker solution to the same degree as in [161], although it does seem to mitigate some of the overfitting errors for geomCG(30). We display slices for fixed source coordinates and varying receiver coordinates in Figure 2.5 for randomly missing points and Figure 2.6 for randomly missing receivers. We summarize our results in Tables (2.2) and (2.3). By exploiting the low-rank structure of the HT format compared to the Tucker format, we are able to achieve much better results than Tucker tensor completion, especially for the realistic case of missing receiver samples.

In all instances for these experiments, the HT tensor completion outperforms the conventional Tucker approach both in terms of recovery quality and recovery speed. We note that geomCG does not scale as well computationally as our HT algorithm for $d > 3$, as the complexity analysis in [161] predicts. As such, we only consider the HT interpolation for the next sections, where we will solve the tensor completion problem for much larger data volumes.

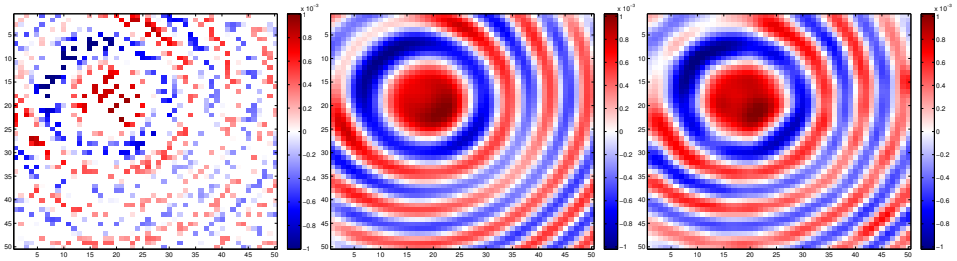


True data | Input data $b = P_\Omega \mathbf{X}^*$ | HT(30,80) - SNR 30.5 dB | geomCG(20)-0 - SNR 29.4 dB

Figure 2.5 Reconstruction results for 90% missing points, best results for geomCG and HTOpt.



90% missing receivers | HT(20,20) - SNR 7.04 dB | geomCG(20)-0 - SNR −1.92 dB



70% missing receivers | HT(20,20) - SNR 20.4 dB | geomCG(30)-5 - SNR 16.8 dB

Figure 2.6 Reconstruction results for sampled receiver coordinates, best results for geomCG and HTOpt. Top row: 90% missing receivers. Bottom row: 70% missing receivers.

Percentage Training Data	10 %	30 %	50 %
geomCG(20) - 0	28.5 (1023)	30.5 (397)	30.7 (340)
geomCG(30) - 0	-6.7 (1848)	21.8 (3621)	31.5 (2321)
geomCG(30) - 5	16.1 (492)	13.8 (397)	15.5 (269)
HTOpt(20,60)	30.1 (83)	30.4 (59)	30.4 (57)
HTOpt(20,80)	30.3 (121)	30.8 (75)	30.8 (53)
HTOpt(30,80)	**31.6** (196)	**32.9** (133)	**33.1** (114)

Table 2.2 Reconstruction results for single reflector data - missing points - mean SNR over 5 random training sets. Values are SNR (dB) and time (in seconds) in parentheses.

2.8.3 Convergence speed

We demonstrate the superior convergence of the Gauss-Newton method compared to the Steepest Descent and Conjugate Gradient methods when used to complete the simple, single reflector data volume with 50% missing receivers and all ranks set to 20. We start each method with the same initial point and allow each to run for

Percentage Training Data	10 %	30 %	50 %
geomCG(20) - 0	-5.1 (899)	9.9 (898)	18.5 (891)
geomCG(30) - 5	-3.6 (1796)	-4.7 (1834)	6.1 (1802)
HTOpt(20,20)	**6.1** (111)	**19.8** (101)	20.1 (66)
HTOpt(30,20)	2.8 (117)	18.1 (109)	19.8 (94)
HTOpt(30,40)	0.0 (130)	13.4 (126)	**21.6** (108)

Table 2.3 Reconstruction results for single reflector data - missing receivers - mean SNR over 5 random training test sets. Values are SNR (dB) and time (in seconds) in parentheses.

at most 1000 iterations. As we can see in Figure 2.7, the Gauss-Newton method converges much faster than the other two while simultaneously having per-iteration computational costs that are comparable to the other methods.

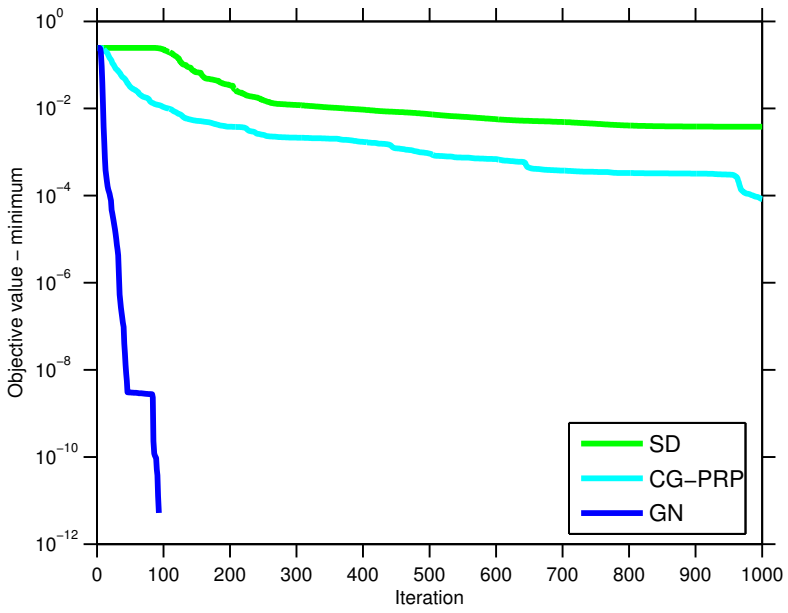
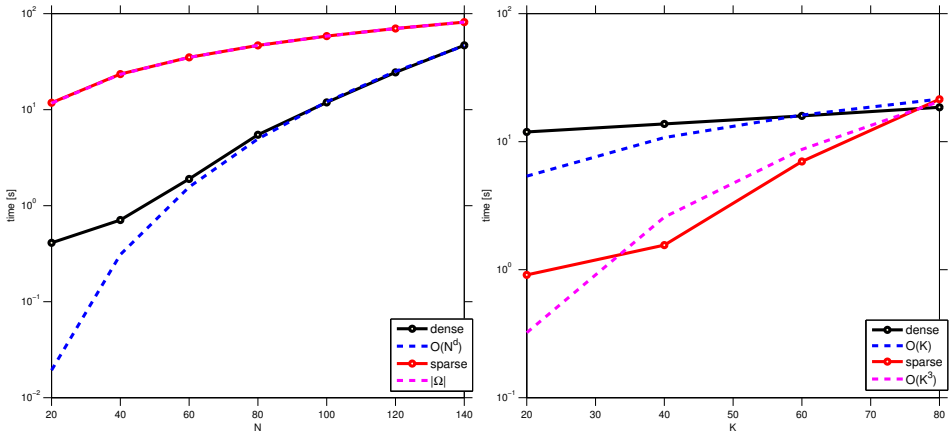


Figure 2.7 Convergence speed of various optimization methods

2.8.4 Performance

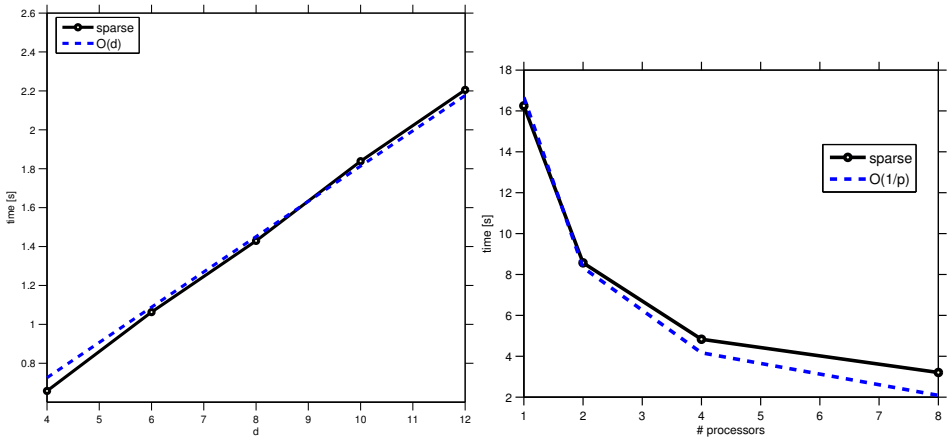
We investigate the empirical performance scaling of our approach as N, d, K, and $|\Omega|$ increase, as well as the number of processors for the parallel case, in Figure 2.8. Here we denote the use of Algorithm (2.1) as the "dense" case and Algorithm (2.2) as the "sparse" case. We run our optimization code in Steepest Descent mode with a single iteration for the line search, and average the running time over 10 iterations and 5 random problem instances. Our empirical performance results agree very

closely with the theoretical complexity estimates, which are $O(N^d K)$ for the dense case and $O(|\Omega| d K^3)$ for the sparse case. Our parallel implementation for the sparse case scales very close to the theoretical time $O(1/\#\text{ processors})$.



Fixed K, d, varying N, $|\Omega| = 1000N$

Fixed $N, d, |\Omega|$, varying K



Fixed $N, K, |\Omega|$, varying d

Fixed N, K, d, Ω, varying # of processors

Figure 2.8 Dense & sparse objective, gradient performance.

2.8.5 Synthetic BG Compass data

This data set was provided to us by the BG Group company and consists of 5D data generated from an unknown synthetic model. Here $n_{src} = 68$ and $n_{rec} = 401$ and we extract frequency slices at 4.86 Hz, 7.34 Hz, and 12.3 Hz. On physical grounds, we expect a slower decay of the singular values at higher frequencies and thus the problem is much more difficult at 12.3 Hz compared to 4.86 Hz.

At these frequencies, the data has relatively low spatial frequency content in the receiver coordinates, and thus we subsample the receivers by a factor of 2 to $n_{rec} = 201$, for the purposes of speeding up the overall computation and ensuring that the intermediate vectors in the optimization are able to fit in memory. Our overall data volume has dimensions $\mathbf{D} \in \mathbb{R}^{68\times68\times201\times201}$.

We randomly remove varying amounts of receivers from this reduced data volume and interpolate using 50 iterations of the GN method discussed earlier. We display several recovered slices for fixed source coordinates and varying receiver coordinates (so-called common source gathers in seismic terminology) in Figures [2.9, 2.10, 2.11].

We summarize our recovery results for tensor completion on these data sets from missing receivers in Table 2.4 and the various recovery parameters we use in Table 2.5. When the subsampling rate is extremely high (90% missing receivers in these examples), the recovery can suffer from overfitting issues, which leads to spurious artifacts in the recovered volume and lower SNRs overall. Using the Gramian-based regularization method discussed earlier, we can mitigate some of those artifacts and boost recovered SNRs, as seen in Figure 2.12.

Frequency	% Missing	Train SNR (dB)	Test SNR (dB)	Runtime (s)
4.86 Hz	25%	21.2	21	4033
	50%	21.3	20.9	4169
	75%	21.5	19.9	4333
	90%	19.9	10.4	4679
	90%*	20.8*	13.0*	5043
7.34 Hz	25%	17.3	17.0	4875
	50%	17.4	16.9	4860
	75%	17.7	16.5	5422
	90%	16.6	9.82	4582
	90%*	16.6*	10.5*	4947
12.3 Hz	25%	14.9	14.2	5950
	50%	15.2	13.8	7083
	75%	15.8	9.9	7387
	90%	13.9	5.39	4578
	90%*	14*	6.5*	4966

Table 2.4 HT Recovery results on the BG data set - randomly missing receivers. Starred quantities are computed with regularization.

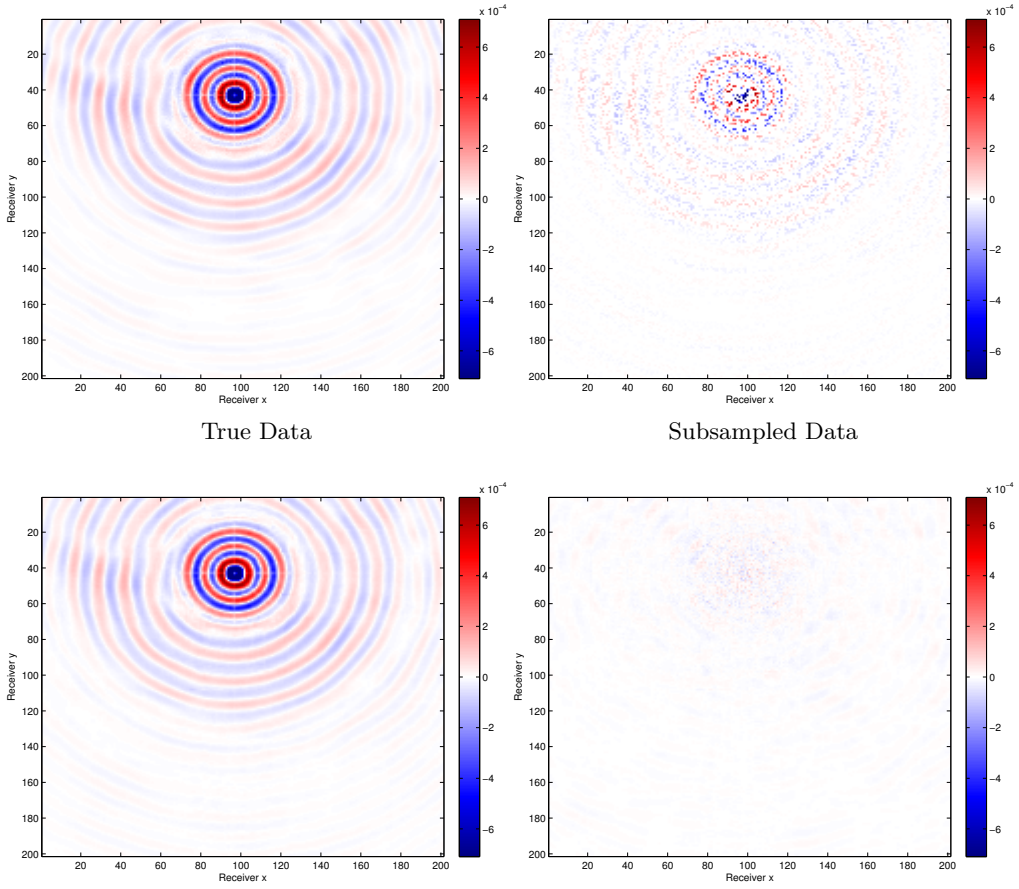


Figure 2.9 HT interpolation results on the BG data set with 75% missing receivers at 4.68 Hz, figures are shown for fixed source coordinates.

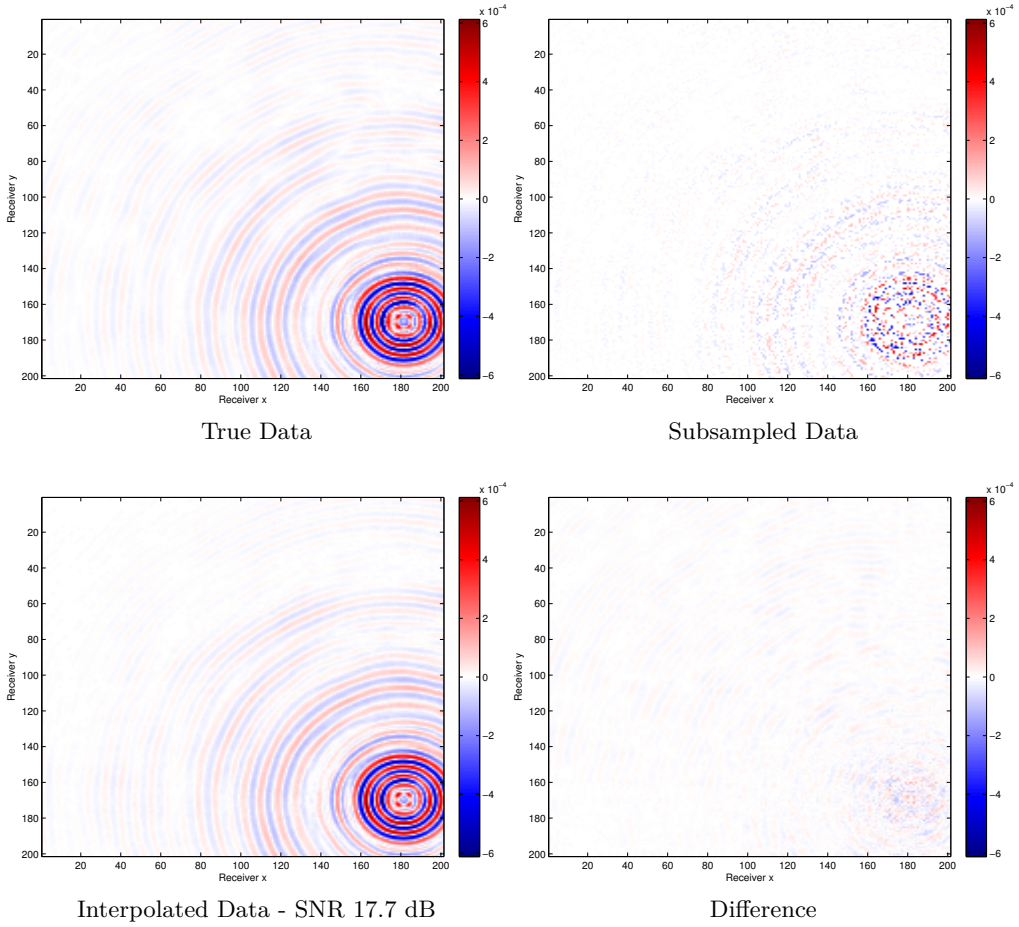


Figure 2.10 HT interpolation results on the BG data set with 75% missing receivers at 7.34 Hz, figures are shown for fixed source coordinates.

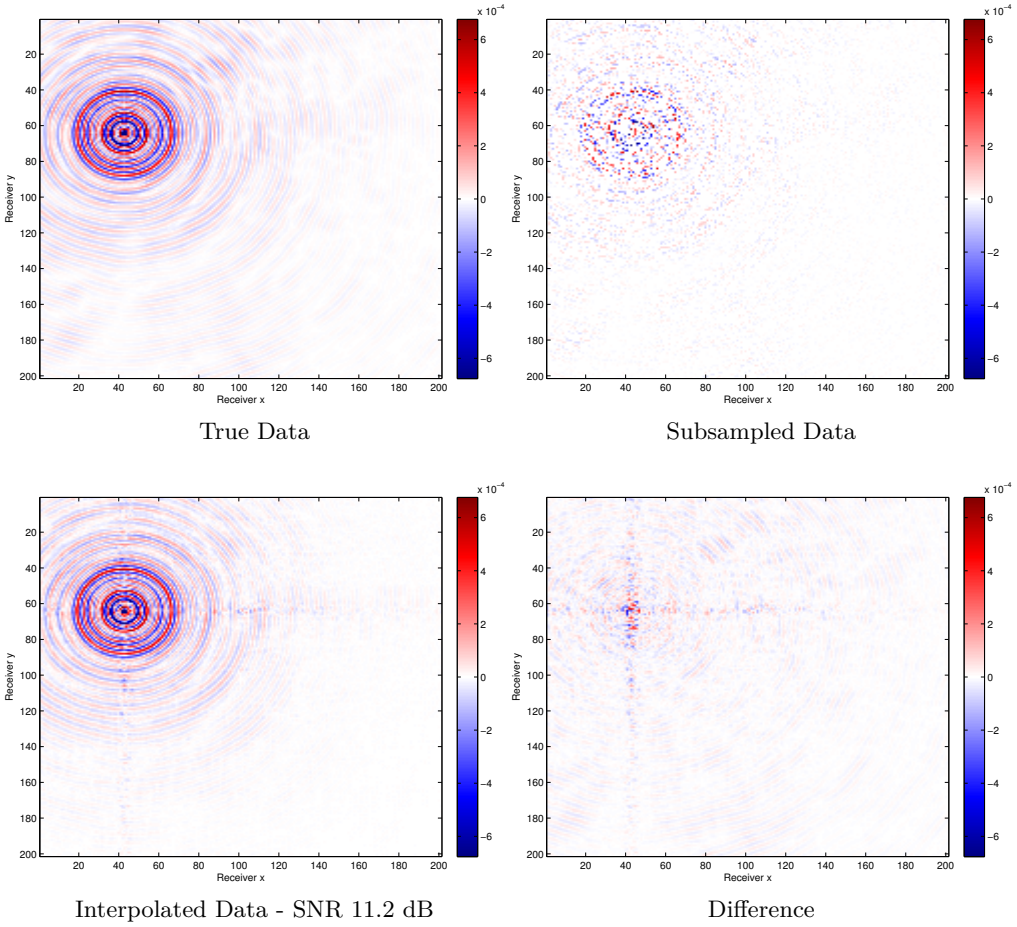


Figure 2.11 HT interpolation results on the BG data set with 75% missing receivers at 12.3 Hz, figures are shown for fixed source coordinates.

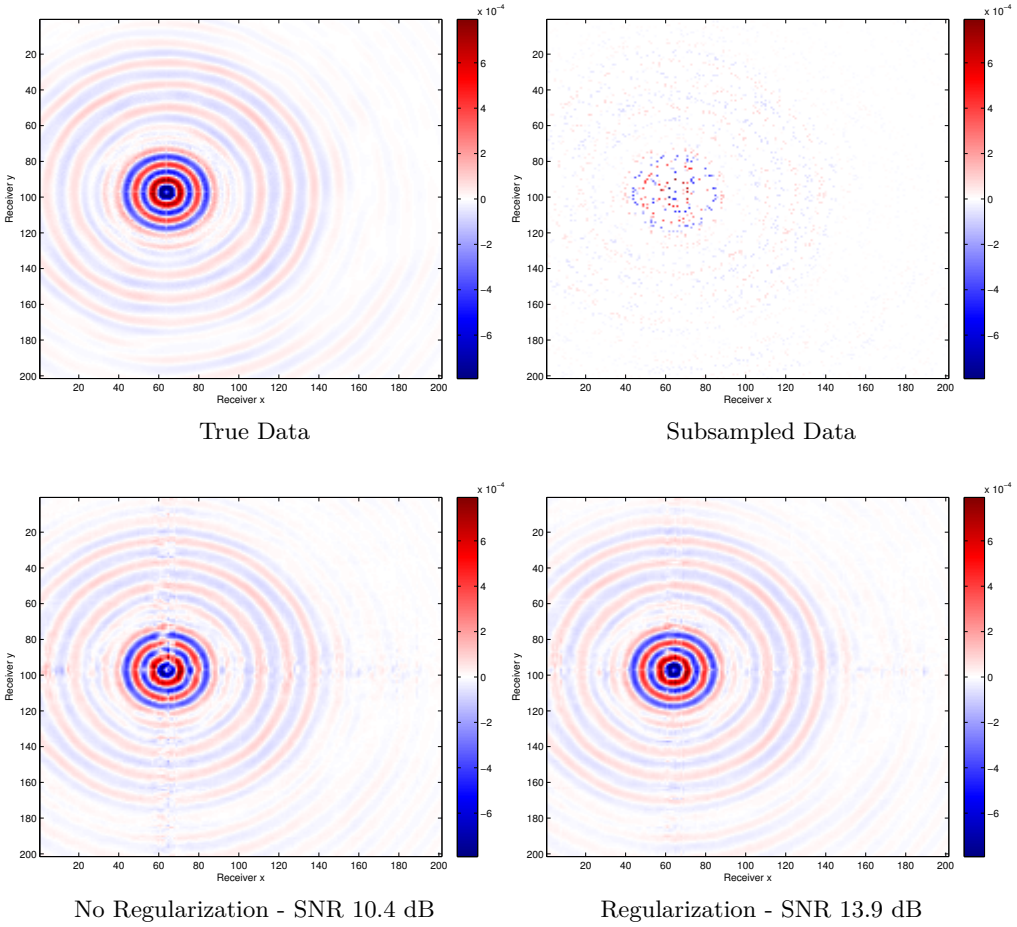


Figure 2.12 Regularization reduces some of the spurious artifacts and reduces overfitting in the case where there is very little data. 4.86 Hz data, 90% missing receivers.

Frequency	$r_{x_{src}x_{rec}}$	$r_{x_{src}}$	$r_{x_{rec}}$	HT-SVD SNR (dB)
4.86 Hz	150	68	120	21.1
7.34 Hz	200	68	120	17.0
12.3 Hz	250	68	150	13.9

Table 2.5 HT parameters for each data set and the corresponding SNR of the HT-SVD approximation of each data set. The 12.3 Hz data is of much higher rank than the other two data sets and thus is much more difficult to recover.

2.8.6 Effect of varying regularizer strength

In this section, we examine the effect of varying the λ parameter based on the scenario described in Figure 2.12. That is to say, we use the same data volume $\mathbf{D}$ at as in the previous section with 90% of the receiver points removed and recover this volume using 50 iterations of the Gauss-Newton method for $\lambda = 0, 10^{-15}, \ldots, 10^{-10}$. We plot the results of the test SNRs in Figure 2.13 and indicate 0 on the log-scale graph by -16, since 10^{-16} is effectively at machine precision. For this problem with very little data, choosing an appropriate λ parameter can mitigate some of the overfitting errors and in our experiments, we found a parameter range from $10^{-12} - 10^{-10}$ improved our recovery SNRs, i.e., small but nonzero values. Higher values of λ than those shown here resulted in underfitting of the data and significantly worse SNRs and were omitted. In practice, one would employ a cross-validation scheme to choose an appropriate λ.

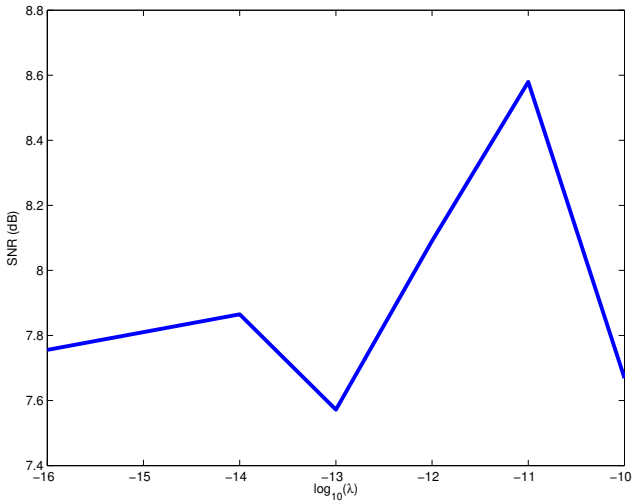


Figure 2.13 Recovery SNR versus $log_{10}(\lambda)$ for 4.68Hz data. 90% missing receivers.

2.9 Discussion

For the seismic data examples considered in this chapter, we have assumed a particular sampling scheme, i.e., receiver-only sampling, that is much more feasible acquisition scenario than the traditional pointwise sampling in all coordinates simultaneously. Other application areas besides seismology may have further insights in to appropriate and feasible sampling schemes for these data volumes.

It is no accident that the frequencies of the data in this section are quite low: low frequency seismic data has a much faster decay of singular values in multiple dimensions compared to higher frequency data [16]. As the frequency of the data increases, potentially a multidimensional extension of the hierarchical partitioning

scheme introduced in [163] could be employed to reconstruct the data. Given that we are ultimately interested in using this completed data volume for solving an inverse problem, a multi-frequency inversion scheme modeled by the Helmholtz equation, for example, is typically only performed in a frequency band from, say, 3-15Hz. If we are only interested in frequencies in this range, the HT method can perform well.

One important concern for utilizing these methods in practice is how to adequately estimate the HT rank parameters in each dimension, given data with missing entries. Cross-validation is one viable, albeit expensive, strategy for exploring the space of HT ranks, given the computational costs associated to interpolating the large data volume. Alternatively, one may simply stipulate that the ranks cannot exceed a fixed fraction of the spatial sampling of their corresponding dimensions, arising from the need to keep the computational costs under control.

2.10 Conclusion

In this chapter we have developed the algorithmic components to solve optimization problems on the manifold of fixed-rank Hierarchical Tucker tensors. By exploiting this manifold structure, we solve the tensor completion problem where the tensors of interest exhibit low-rank behavior. Our algorithm is computationally efficient because we mostly rely on operations on the small HT parameter space. The manifold optimization itself guarantees that we do not run into convergence issues, which arise when we ignore the quotient structure of the HT format. Our application of this framework to seismic examples confirms the validity of our new approach and outperforms existing Tucker-based approaches for large data volumes. To stabilize the recovery for high subsampling ratios, we introduced an additional regularization term that exploits properties of the Gramian matrices without the need to compute SVDs in the ambient space.

While the method clearly performs well on large-scale problems, there are still a number of theoretical questions regarding the performance of this approach. In particular, the generalization of matrix completion recovery guarantees to the HT format remains an open problem. As in many alternative approaches to matrix/tensor completion, the selection of the rank parameters and regularization parameters remain challenging both theoretically and from a practical point of view. However, the chapter clearly illustrates that the HT format is a viable option to represent and complete high-dimensional data volumes in a computationally feasible manner.

Chapter 3

The Moreau Envelope and the Polyak-Lojasiewicz Inequality

3.1 Introduction

In this chapter, we study $\mathcal{C}^1$ convex functions that satisfy the Polyak-Lojasiewicz (PL) inequality, that is to say, functions $f : \mathbb{R}^n \mapsto \mathbb{R}$ satisfying

$$\frac{1}{2}\|\nabla f(x)\|_2^2 \geq \mu(f(x) - \min f) \quad \forall x \in C \tag{3.1}$$

where $\mu > 0$ is a constant and $C \subset \mathbb{R}^n$. We denote the set of all $\mathcal{C}^1$ convex functions satisfying (3.1) as $PL(\mu, C)$ and simply write $PL(\mu)$ when $C = \mathbb{R}^n$. As shown in [147], the PL-inequality is in some sense the weakest condition one can impose on a function in order to ensure that gradient descent converges at a linear rate to a global minimizer. This condition also implies that stationary points are global minimizers, although it does not imply that there are unique minimizers as is the case with a strongly convex function. Strongly convex functions satisfy (3.1), but it is possible for convex but not strongly convex, or even some non-convex functions, to satisfy this inequality. In this chapter, we consider the behaviour of functions satisfying (3.1) under various transformations, including composition with smooth maps and under the Moreau envelope. Lemma 3.4 proved in this chapter will be used in Chapter 4, while the other lemmas are more general, and all of the stated results are novel.

3.2 Notation

The notation introduced in this section will encompass this chapter as well as Chapter 4.

We use $\langle x, y\rangle$ to denote the Euclidean inner product between vectors x and y. ℓ_p norms are denoted as $\|x\|_p = (\sum_{i=1}^n |x_i|^p)^{1/p}$ for $1 \leq p < \infty$ and $\|x\|_\infty = \max_{i=1,\dots,n} |x_i|$. The ℓ_0 pseudonorm counts the number of nonzero elements of a vector, i.e., $\|x\|_0 = |\{i : x_i \neq 0\}|$. Mixed $\ell_{p,q}$ norms operate on matrices and satisfy

$$\|X\|_{p,q} = \left(\sum_i \left(\sum_j (|X_{i,j}|^q)^{1/q}\right)^p\right)^{1/p}$$

when $p, q \in [1, \infty)$ and similarly for p or $q = \infty$.

The minimum value of a real-valued function $f : \mathbb{R}^n \mapsto \mathbb{R}$ is denoted either as $\min_x f(x)$ or $\min f$ when convenient. The set of minimizers is denoted $\operatorname{argmin} f = \operatorname{argmin}_x f(x) = \{x : f(x) = \min f\}$. When f is convex, $\operatorname{argmin} f$ is a convex set. The domain of an extended-real valued function $f : \mathbb{R}^n \mapsto \mathbb{R} \cup \{\infty\}$ is the set $\operatorname{dom} f = \{x : f(x) < \infty\}$. A proper convex function is one such that $\operatorname{dom} f \neq \emptyset$. The convex hull of a set C is the set of all convex combinations of elements of C, $\operatorname{conv} C := \{\sum_{i=1}^p \lambda_i x_i : \sum_{i=1}^p \lambda_i = 1, \lambda_i \geq 0, x_i \in C$. The interior of a set C is defined as $\operatorname{int} C := \bigcup_{D \subset C, D \text{ is open}} D$.

A mapping $G : \mathbb{R}^n \mapsto \mathbb{R}^m$ is *Lipschitz-continuous* with constant L when it satisfies

$$\|G(x) - G(y)\|_2 \leq L\|x - y\|_2 \quad \forall x, y.$$

The indicator function of a set C, denoted $\delta_C(x)$ or $\delta(x|C)$, is defined as

$$\delta_C(x) = \begin{cases} 0 & \text{if } x \in C \\ \infty & \text{otherwise} \end{cases}$$

The epigraph of a function $f(x)$, denoted $\operatorname{epi} f$, is the set $\operatorname{epi} f = \{(x, \tau) : f(x) \leq \tau\}$. For a closed, convex set $C \subset \mathbb{R}^n$, the distance function,

$$\begin{aligned} d_C(x) = \min_z \|z - x\|_2 \\ \text{s.t. } z \in C, \end{aligned}$$

which measures the distance from the vector x to the set C, is a convex function. We have that $d_C(x) = 0$ if and only if $x \in C$. For $x \notin C$, there is always a unique closest point in C to x that we denote as $P_C(x)$, which satisfies

$$\begin{aligned} P_C(x) = \operatorname*{argmin}_z \|z - x\|_2 \\ \text{s.t. } z \in C. \end{aligned}$$

Moreover, the squared distance function $\frac{1}{2} d_C^2(x)$ is convex and differentiable with

$$\nabla \frac{1}{2} d_C^2(x) = x - P_C(x).$$

The Moreau envelope of a convex function $f(x)$ with parameter λ is defined as

$$M_f^\lambda(z) = \min_x \frac{1}{2\lambda}\|x - z\|_2^2 + f(x).$$

We let $P_f^\lambda(z) = \text{argmin}_x \frac{1}{2\lambda}\|x - z\|_2^2 + f(x)$ be the corresponding proximal operator. Note that $\frac{1}{2}d_C^2(x) = M_{\delta_C}^1$. Moreau envelopes are well studied in the literature, and we recall some basic facts here

Facts 3.1

1. M_f^λ is convex and $\mathcal{C}^1$, whenever f is convex [Theorem 2.26, 216]
2. $\nabla M^\lambda f(x) = \frac{1}{\lambda}(x - P_f^\lambda(x))$
3. The gradient of $M^\lambda f$ is λ-Lipschitz continuous
4. $\text{argmin}\, M^\lambda f = \text{argmin}\, f$ and $\min M^\lambda f = \min f$
5. $x \in \text{argmin}\, f$ if and only if $P_f^\lambda x = x$
6. M_f^λ is strongly convex if and only if f is strongly convex [Lemma 2.19, 203]

From [12.24, 25], we have that

$$M_f^\lambda(x) = f(P_f^\lambda x) + \frac{1}{2\lambda}\|x - P_f^\lambda x\|_2^2 \tag{3.2}$$

A convex function $h(x)$ is a gauge if it satisfies $h(x) = \gamma(x|C) = \inf\{\lambda : x \in \lambda C\}$, where C is a convex set containing the origin, and so $h(0) = 0$. Examples of such functions are norms, in particular atomic norms [67]

$$\|x\|_\mathcal{A} := \inf\{\lambda \geq 0 | x \in \lambda \text{conv}(\mathcal{A})\} \tag{3.3}$$

where $\mathcal{A}$ is a set of atoms that induce a low-complexity structure of interest. (3.3) induces a norm when $\mathcal{A}$ is a symmetric set, i.e., $\mathcal{A} = -\mathcal{A}$ and $0 \in \text{int (conv } \mathcal{A})$. For instance, if $\mathcal{A}$ consists of $n-$length vectors with a single nonzero element, then $\|x\|_\mathcal{A}$ is the ℓ_1 norm, the convex relaxation of the ℓ_0 norm.

The corresponding polar gauge $h^\circ(z)$ is defined as

$$h^\circ(z) := \inf\{t > 0 : th(x) \geq \langle x, z\rangle \ \forall x\}.$$

When $h(z) = \|z\|_p$ is a norm, the polar gauge is the correponding dual norm

$$h^\circ(z) = \|z\|_q,$$

where $\frac{1}{p} + \frac{1}{q} = 1$.

3.3 Lemmas

Lemma 3.2: If $f \in PL(\mu, K)$ for a compact set K with $\operatorname{argmin} f \subset K$, $g : \mathbb{R}^m \mapsto \mathbb{R}^n$ is a $\mathcal{C}^1$ mapping satisfying

$$\min_{x \in K} \sigma_{\min}(\nabla g(x)) = \lambda > 0,$$

then the composition $k(x) = f(g(x))$ satisfies the PL-inequality with constant $\lambda^2 \mu$ on K. When $g(x) = Ax - b$ is affine, for some matrix A and constant vector b, $\lambda = \sigma_{\min}(A)$.

Proof We have that

$$\begin{aligned}\frac{1}{2}\|\nabla k(x)\|_2^2 &= \frac{1}{2}\|\nabla g(x)^* \nabla f(g(x))\|_2^2 \\ &\geq \frac{\lambda^2}{2}\|\nabla f(g(x))\|_2^2 \\ &\geq \lambda^2 \mu (f(g(x)) - \min f) = \lambda^2 \mu (k(x) - \min f)\end{aligned}$$

whenever $x \in K$. Note that $\min f \leq \min k = \min_x f(g(x))$ by definition, so $-\min f \geq -\min k$ and it follows that

$$\frac{1}{2}\|\nabla k(x)\|_2^2 \geq \lambda^2 \mu (k(x) - \min k)$$

as desired.

The PL-constant of f is important for the analysis of steepest descent methods used to compute $\min f$, as analyzed in [147]. The larger the μ, the faster the (worst-case) rate of convergence. Thus, it is interesting to study whether certain transformations of f that preserve the set $\operatorname{argmin} f$ either improve or degrade this constant. For the Moreau-envelope, the following lemmas demonstrate that there is an upper limit to the PL-constant of the Moreau-envelope of a function and the convex functions that realize this upper limit are indicator functions. This will imply that the distance function $\frac{1}{2\lambda} d_C^2(x)$ has a PL-constant of $\frac{1}{\lambda}$.

Lemma 3.3 If $M_f^\lambda \in PL(\mu)$, then $\mu \leq \dfrac{1}{\lambda}$.

Proof We rewrite (3.2) as

$$\begin{aligned}M_f^\lambda(x) - \min f &= f(P_f^\lambda x) - \min f + \frac{\lambda}{2}\|\nabla M_f^\lambda x\|_2^2 \\ &\geq \frac{\lambda}{2}\|\nabla M_f^\lambda x\|_2^2\end{aligned}$$

since $f(P_f^\lambda x) - \min f \geq 0$. Since $M_f^\lambda \in PL(\mu)$, then

$$\frac{\lambda}{2}\|\nabla M_f^\lambda x\|_2^2 \leq M_f^\lambda(x) - \min f \leq \frac{1}{2\mu}\|\nabla M_f^\lambda x\|_2^2$$

which implies that $\mu \leq \frac{1}{\lambda}$.

Lemma 3.4 Given the Moreau envelope of a closed, proper convex function f, $M^\lambda f$, $M^\lambda f \in PL(\frac{1}{\lambda})$ if and only if $f = \delta_C + d$ for some convex set C and some constant d.

Proof The "only if" direction follows by computing, supposing $d = 0$ without loss of generality,

$$\begin{aligned}\frac{1}{2}\|\nabla M^\lambda_{\delta_C}\|_2^2 &= \frac{1}{2}\|\lambda^{-1}(x - P_C(x))\|_2^2 \\ &= \frac{1}{\lambda}(M^\lambda_{\delta_C}).\end{aligned}$$

Suppose that $M^\lambda f$ satisfies the PL-inequality with $\mu = \frac{1}{\lambda}$. Then for any $x \in \mathrm{dom} f$,

$$\frac{1}{2}\|\nabla M^\lambda f\|_2^2 \geq \frac{1}{\lambda}(M^\lambda_f(x) - \min f).$$

Here we note that $\min M^\lambda_f = \min f$.

So by (3.2)

$$\begin{aligned}\frac{1}{2}\|\nabla M^\lambda f\|_2^2 &= \frac{1}{2\lambda^2}\|x - P^\lambda_f x\|_2^2 \\ &= \frac{1}{\lambda}(M^\lambda_f(x) - f(P^\lambda_f x))\end{aligned}$$

And therefore

$$\frac{1}{\lambda}(M^\lambda_f(x) - f(P^\lambda_f x)) \geq \frac{1}{\lambda}(M^\lambda_f(x) - \min f)$$

Rearranging the above inequality, it follows that $f(P^\lambda_f x) \leq \min f$, and therefore $f(P^\lambda_f x) = \min f$, which implies that $P^\lambda_f x \in \mathrm{argmin}\, f$.

Since $f(P^\lambda_f x) = \min f$, we must have that $P^\lambda_f x = x$, which is equivalent to $x \in \mathrm{argmin}\, f$. Since x was arbitrary, it follows that $\mathrm{dom} f = \mathrm{argmin}\, f$, and so f is constant on its domain. As f is convex, this set, which we denote C, is convex, and it follows that $f = \delta_C + d$, where $d = \min f$.

Lemma 3.5 If $f \in PL(\mu)$, then its Moreau envelope satisfies $M^\lambda_f \in PL(\frac{\mu}{1+\mu\lambda})$.

Proof Note that since the proximal point $P^\lambda_f(x) = \mathrm{argmin}_y \frac{1}{2\lambda}\|x - y\|_2^2 + f(y)$, the optimal conditions of this program imply that $x - P^\lambda_f x = \lambda \nabla f(P^\lambda_f x)$. Therefore we have that

$$\begin{aligned}\frac{1}{2}\|\nabla M^\lambda_f(x)\|_2^2 &= \frac{1}{2}\|\lambda^{-1}(x - P^\lambda_f x)\|_2^2 \\ &= \frac{1}{2}\|\nabla f(P^\lambda_f x)\|_2^2 \\ &\geq \mu(f(P^\lambda_f x) - \min f) \\ &= \mu(M^\lambda_f(x) - \frac{1}{2\lambda}\|x - P^\lambda_f x\|_2^2 - \min f)\end{aligned}$$

where the second line is the PL-inequality for f and the last line follows from (3.2).

Rearranging yields

$$\begin{aligned}\frac{1}{2}\|\nabla M_f^\lambda(x)\|_2^2 + \frac{\mu}{2\lambda}\|x - P_f^\lambda x\|_2^2 &= (1+\mu\lambda)\frac{1}{2}\|\nabla M_f^\lambda(x)\|_2^2 \\ &\geq \mu(M_f^\lambda(x) - \min f)\end{aligned}$$

and the theorem follows from noting that $\min f = \min M_f^\lambda$.

In the goal to have as large of a μ as possible, the previous lemma demonstrates that the Moreau envelope can only decrease the PL constant of the original function, resulting in slower convergence of gradient descent in the worse-case scenario. There exist functions that realize this worst-case scenario, showing that the aforementioned bounds are tight.

Example For the simple quadratic function $f(x) = \dfrac{\sigma}{2}\|x\|_2^2$, it is easy to show that $f \in PL(\sigma)$ and that this is the largest PL-constant this function can have. The corresponding Moreau envelope M_f^λ is

$$M_f^\lambda(x) = \frac{1}{2}\frac{\sigma}{1+\lambda\sigma}\|x\|_2^2$$

which has a maximal PL-constant of $\dfrac{\sigma}{1+\lambda\sigma}$, which is strictly less than $\dfrac{1}{\lambda}$, as per Lemma 3.3.

In this context, one can ask how much one can "improve" a function, imbuing f with the PL-inequality through its Moreau envelope, if f does not satisfy the PL-inequality originally. In general, this might be difficult to analyze for general convex functions. Our previous analysis deals with the case when f is an indicator function, and so we consider specific computable examples below.

The Huber function, $\kappa_\lambda(x)$, is the Moreau envelope of the absolute value function

$$\begin{aligned}\kappa_\lambda(x) &= \min_y \frac{1}{2\lambda}(x-y)^2 + |y| \\ &= \begin{cases} \dfrac{1}{2\lambda}x^2 & \text{if } |x| \leq \lambda \\ |x| - \dfrac{1}{2}\lambda & \text{if } |x| > \lambda \end{cases}.\end{aligned} \tag{3.4}$$

The vector Huber norm, by abuse of notation is also written $\kappa(x)$, is defined as $\kappa(x) = \sum_{i=1}^n \kappa(x_i)$, and satisfies $\kappa(x) = \min_y \dfrac{1}{2\lambda}\|y-x\|_2^2 + \|y\|_1$. It is often used in regression contexts [114, 180, 179], to smooth out the singularity associated to the ℓ_1 norm yet retain the linear growth for large residuals, which is related to a smaller sensitivity to large outlying values. The Huber norm is not strongly convex, since $\|x\|_1$ is not strongly convex.

Lemma 3.6 For any $0 < \mu \leq \frac{1}{\lambda}$, $\kappa_\lambda(x) \in PL(\mu, C)$ where

$$C = \left\{ x : \|x\|_\infty \leq \frac{1}{2\mu} + \frac{1}{2\lambda} \right\}$$

Proof We consider the one-dimensional Huber norm, as the multi-dimensional case follows a similar argument. Note that

$$\kappa'_\lambda(x) = \begin{cases} \frac{1}{\lambda} x & \text{if } |x| \leq \lambda \\ \text{sign}(x) & \text{if } x > \lambda \end{cases}$$

where $\text{sign}(x) = \begin{cases} x/|x| & \text{if } x \neq 0 \\ 0 & \text{if } x = 0 \end{cases}$.

Fix any $0 < \mu \leq \frac{1}{\lambda}$. In the region $|x| \leq \lambda$, it follows that

$$\begin{aligned} \frac{1}{2}(\kappa'_\lambda(x))^2 &= \frac{1}{\lambda}\kappa_\lambda(x) \\ &\geq \mu\kappa_\lambda(x) \end{aligned}$$

When $\lambda \leq |x| \leq \frac{1}{2\mu} + \frac{1}{2}\lambda$, we have that

$$\frac{1}{2}(\kappa'_\lambda(x))^2 = \frac{1}{2},$$

and since $|x| \leq \frac{1}{2\mu} + \frac{1}{2}\lambda$,

$$\mu\kappa_\lambda(x) = \mu(|x| - \frac{1}{2}\lambda) \leq \frac{1}{2}.$$

To ensure that $\lambda \leq \frac{1}{2\mu} + \frac{1}{2}\lambda$, it must hold that $\mu \leq \frac{1}{\lambda}$. Note that we have $\mu = \frac{1}{\lambda}$ in this case without contradicting Lemma 3.4 since the set of points on which the PL-inequality holds for $\kappa_\lambda(x)$ is not the entirety of $\text{dom}(\kappa_\lambda)$. Decreasing the parameter μ increases the region of validity of this bound. The interested reader can verify that this is the largest set on which (3.1) holds for $\kappa_\lambda(x)$.

Chapter 4

A level set, variable projection approach for convex composite optimization

4.1 Introduction

Although elegant in their mathematical simplicity and computational efficiency, real-world noise often does not follow a Gaussian distribution. Instead, measurements acquired from a physical experiment can be contaminated with large outliers. Consider the *robust tensor completion* problem, where our signal, which is a low-rank tensor, has been subsampled and a percentage of the remaining samples has been corrupted by high amplitude noise. As mentioned previously in Section 1.2, the maximum likelihood estimator is the solution to the following optimization problem

$$\min_{\mathbf{X}\in\mathcal{H}} \|\mathcal{A}(\mathbf{X}) - b\|_1.$$

Here $\mathcal{H}$ is our chosen class of low-rank tensors, $\mathcal{A}$ is the subsampling operator, . We consider the class of Hierarchical Tucker tensors, as developed in Chapter 2, which allows us to write this program as

$$\min_{x\in\mathcal{M}} \|\mathcal{A}(\phi(x)) - b\|_1.$$

This problem instance is a particular case of the more general class of convex composite problems, an important class of non-convex optimization programs of the form

$$\min_{x\in C} h(c(x)). \tag{4.1}$$

Here $h : \mathbb{R}^m \mapsto [-\infty, \infty]$ is a (typically) non-smooth, proper function with closed, bounded level sets, $c : \mathbb{R}^n \mapsto \mathbb{R}^m$ is a $\mathcal{C}^2$ mapping, and C is a closed convex set, often $\mathbb{R}^n$. Despite being non-smooth, h is often quite structured, typically convex, and often polyhedral. Here we assume that we can project on to the level sets of h easily, i.e., we can efficiently compute

$$P_{h\leq\tau}(y) = \operatorname*{argmin}_x \|x - y\|_2 \quad \text{s.t. } h(x) \leq \tau$$

for a given pair (y, τ). For theoretical reasons, we also assume that c is coercive, i.e., $\lim_{x\to\infty} \|c(x)\| = \infty$. This problem formulation encompasses many important applications such as hinge loss support vector machines [121, 267, 227] and nonlinear least squares [107, 151, 88]. Problems of the form

$$\min_x f(x) + g(x)$$

where f is smooth and g is non-smooth but convex and "prox-friendly", i.e., $\min_z \frac{1}{2\lambda}\|z-x\|_2^2+g(z)$ is easily computable, are often referred to as convex-composite problems in the literature [43], but we use the nomenclature *additive composite* problems here to distinguish this particular case. Problems of this form can be written in the convex-composite form (4.1) with $c(x) = [\ f(x); x\ ]$ and $h(f, x) = f + g(x)$. See [167] and the references contained therein for more examples.

Despite the ubiquity of this problem formulation, there have been relatively few numerical examples tackling problems in the form (4.1) directly. Many approaches have been developed that sidestep the issue of the non-smoothness of h through the use of the Alternative Direction of Multipliers Method (ADMM) [59, 229], which decouples the problem in to a series of simpler subproblems. The convergence of ADMM is not well understood for general nonlinear operators and one must solve a large number of nonlinear least-squares problems as a result. From a practical point of view, there are a number of auxiliary parameters that affect the convergence of such methods and their dependence on these parameters is not well understood in general, although some effort has been made to understand specific cases in [106]. One could also smooth the function h with a Moreau envelope or similar mapping, but the optimal smoothness parameter can depend on constants that are unknown in general [91].

One technique for solving (4.1) is the Gauss-Newton method [43, 44, 169, 265]. At every iteration, the Gauss-Newton method linearizes the function c around the current point x_k and repeatedly chooses a step direction satisfying

$$d_k \in \operatorname*{argmin}_{\|d\|\leq\Delta} h(c(x_k) + \nabla c(x_k)d)$$

for some constant $\Delta > 0$. Practically speaking, solving these subproblems is extremely challenging for large scale problems, owing to the norm constraint on d and the non-smooth objective h. Another approach, first proposed in [167] and studied

further in [168, 139], is to use a proximal method to solve the linearized problem, i.e., by choosing a search direction d_k satisfying

$$d_k = \operatorname*{argmin}_d h(c(x_k) + \nabla c(x_k)d) + \frac{\mu}{2}\|d\|_2^2. \tag{4.2}$$

Although this subproblem is strongly convex in d, it is still nonsmooth and an explicit representation for d_k does not usually exist, unlike for proximal methods for additive problems. For instance, using the mirror descent method to solve (4.2) yields a convergence rate of $O(1/T)$ after T iterations, which is far too slow for large problems [144].

The aforementioned approaches tackle the original optimization problem directly, which can be computationally challenging for non-smooth h. One alternative method to deal with such optimization programs is the SPGL1 algorithm introduced in [255]. The SPGL1 approach solves the Basis Pursuit problem

$$\begin{aligned} &\min_x \|x\|_1 \\ &\text{s.t. } Ax = b \end{aligned}$$

using a level-set method. Instead of solving the BP problem directly, which has a non-smooth objective and difficult to satisfy constraint for general linear operators A, SPGL1 inverts the objective and constraints, yielding a series of related problems

$$\begin{aligned} v(\tau) = \min_x \ &\frac{1}{2}\|Ax - b\|_2^2 \\ \text{s.t. } &\|x\|_1 \leq \tau \end{aligned} \tag{4.3}$$

for a specifically chosen sequence of scalar parameters τ. Here $v(\tau)$ is referred to as the value function of the original problem. In this case, the smallest $\tau > 0$ such that $v(\tau) = 0$ yields a solution x that solves the original BP problem. The value function is much easier to evaluate than the original problem, as one solves an optimization problem with a smooth objective and a simple to project on constraint. The authors use a Newton root-finding method to update τ, exploiting the relationship between $v'(\tau)$ and the dual of (4.3). This idea of "flipping" the optimization problem to convert a non-smooth objective in to a simple constraint, along with much of the same machinery as SPGL1, will allow us to solve the more general problem of (4.1). These ideas are further explored in [17] for a particular class of convex programs.

4.2 Technique

We reformulate (4.1) by introducing the variable $z = c(x)$ in order to decouple the non-smooth function h from the smooth mapping c, which gives us the equivalent

problem

$$\begin{aligned} \min_{x,z} \quad & h(z) \\ \text{s.t.} \quad & c(x) - z = 0. \end{aligned} \tag{4.4}$$

Using the same SPGL1-type approach as previously, we flip the objective and the constraints to consider the associated value function of this problem, which is

$$\begin{aligned} \phi(\tau) = \min_{x,z} \ & \frac{1}{2}\|c(x) - z\|_2^2 \\ \text{such that} \ & h(z) \leq \tau. \end{aligned} \tag{4.5}$$

As previously, the smallest τ that results in $\phi(\tau^*) = 0$, denoted τ^*, is the optimal value of (4.4) and the resulting solution x of (4.5) solves the original problem as well. If our current τ is such that $\phi(\tau) > 0$, then we are super-optimal and infeasible for the original problem, i.e., $\|c(x) - z\|_2 > 0$ and $h(z) < \min_x h(c(x))$.

To ease notation, we let $L_\tau := \{z : h(z) \leq \tau\}$.

In order to evaluate (4.5), we could compute $\phi(\tau)$ by means of a projected gradient method in the joint variables (x, z), but this may converge slowly. On the other hand, we can exploit the fact that the constraints act only on the z variable to speed up convergence with the variable projection (VP) approach [150, 107, 14]. The VP method considers the program $\phi(\tau)$, at a fixed parameter x, as a function of z only. In this case, given our assumptions on h, minimizing the program with respect to z

$$\begin{aligned} P_{L_\tau}(x) = \operatorname*{argmin}_z g(x, z) = & \frac{1}{2}\|c(x) - z\|_2^2 \\ \text{such that} \ & z \in L_\tau \end{aligned}$$

which is easily computable, by assumption.

Plugging this expression in to $g(x, z)$ yields the reduced objective

$$\begin{aligned} \tilde{g}(x) = g(x, z(x)) &= \frac{1}{2}\|c(x) - P_{L_\tau}(x)\|_2^2 \\ &= \frac{1}{2} d^2(c(x), L_\tau). \end{aligned} \tag{4.6}$$

Since L_τ is closed and convex by assumption, $\frac{1}{2}d^2(y, L_\tau)$ is differentiable and therefore so is $\tilde{g}(x)$, with $\nabla\tilde{g}(x) = \nabla c(x)^T(c(x) - z(x))$. This agrees with the variable projection interpretation, as in [13]. Minimizing the reduced objective $\tilde{g}(x)$, has been numerically shown to converge much faster and towards a much higher quality local minimum (in the non-convex case) compared to minimizing the joint objective $g(x, z)$ [13]. Another interpretation for solving (4.5) is that we are looking for the point of minimal distance between the level set L_τ and the image of the mapping c, i.e., $c(\mathbb{R}^n) = \{c(x) : c(x) \in \text{dom}(h)\}$. The τ^* that solves the original problem is the scalar such that $d(c(\mathbb{R}^n), L_\tau) = 0$, i.e., we enlarge the level set L_τ by increasing

the τ parameter until the point where $c(\mathbb{R}^n) \cap L_\tau \neq \emptyset$, but $c(\mathbb{R}^n) \cap L_{\tau'} = \emptyset$ for all $\tau' < \tau$. A pictorial representation of this situation is shown in Figure 4.1.

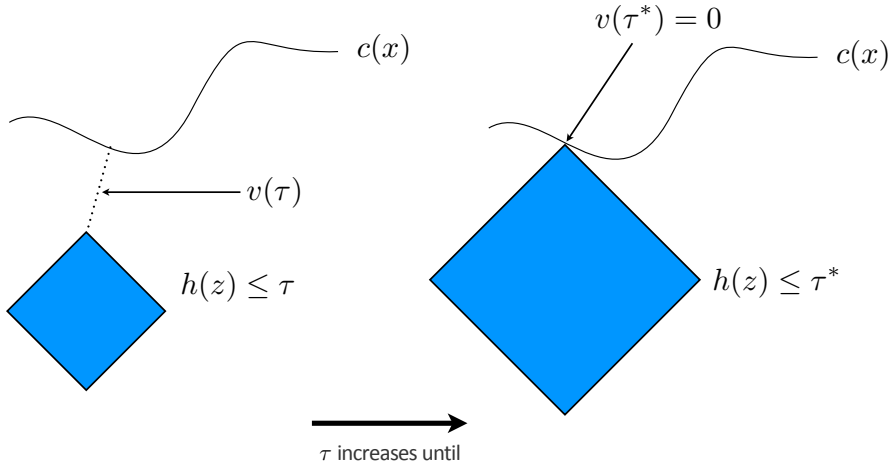


Figure 4.1 A cartoon depiction of the level set method for convex-composite optimization

We can also handle convex constraints on the variable x, such as $\|x\|_1 \leq \beta$ by adding them to the constraints in (4.6). The resulting variable projected problem is simply a nonlinear least squares problem with constraints.

In order to update the parameter τ in the most general case, we use the secant method for finding the root of $\phi(\tau) = 0$, i.e.,

$$\tau_{k+1} = \tau_k - \phi(\tau_k) \frac{\tau_k - \tau_{k-1}}{\phi(\tau_k) - \phi(\tau_{k-1})}. \tag{4.7}$$

The secant method is known to converge superlinearly for sufficiently smooth functions with simple roots [89].

We can also employ Newton's method, which converges quadratically, when we have a simple expression for $v'(\tau)$, which we will see in the following section.

If h is non-convex but has a simple projection, i.e., the ℓ_0 pseudonorm, then we simply use the secant method to update τ. In the case of ℓ_0, since τ is integer-valued, we perform rounding after each secant update. Algorithm (4.1) outlines the full method, which we call the VELVET algorithm (conVex compositE LeVel sEt wiTh variable projection).

Algorithm 4.1 The VELVET algorithm for solving problem (4.1)

Input: Initial point $\tilde{x}$, number of allowed τ updates T
Let $Q(\tau) = \min_x \frac{1}{2} d^2_{h()\leq\tau}(c(x))$
  $q(\tau) = \operatorname{argmin}_x \frac{1}{2} d^2_{h()\leq\tau}(c(x))$
If $h(z)$ is a gauge,
  Newton $\leftarrow$ true
  $\tau_0 \leftarrow 0$
  $f_1 \leftarrow Q(\tau_0)$
  $x_1 \leftarrow q(\tau_0)$
Otherwise
  Newton $\leftarrow$ false
  $\tau_0 \leftarrow 0$, $\tau_1 \leftarrow c$
  $f_0 \leftarrow Q(\tau_0)$
  $x_0 \leftarrow q(\tau_0)$
  $f_1 \leftarrow Q(\tau_1)$
  $x_1 \leftarrow q(\tau_1)$
Endif
for $k = 1, 2, \ldots, T$,
  If Newton
    $\tau_{k+1} = \tau_k - f_k/f'_k$ (4.11)
  Otherwise
    $\tau_{k+1} = \tau_k - (f_k - f_{k-1})/(\tau_k - \tau_{k-1})$
  Endif
  $f_{k+1} = Q(\tau_{k+1})$
  $x_{k+1} = q(\tau_{k+1})$
Return x_{T+1}

4.3 Theory

We consider the convergence estimates when evaluating the subproblems

$$\min_x \frac{1}{2} d^2_{L_\tau}(c(x)). \tag{4.8}$$

To ease the notation, let $z(x) = P_{L_\tau}(x)$ and $D(x) = \frac{1}{2} d^2_{L_\tau}(x)$, so the subproblem objective can be written as $p = D \circ c$. We consider algorithms defined on the set $C = \{x : p(x) \leq p(x_0)\}$ for some fixed point x_0.

Lemma 4.1 If g is a continuous, coercive mapping and the level set L_τ is bounded, then C is compact.

Proof

Suppose C is not bounded. Then there is a sequence $x_n \in C$ with $\|x_n\|_2 \to \infty$. Then $\|c(x_n)\|_2 \to \infty$, since c is coercive. Then we have that there is a k such that $\|c(x_k)\|_2 > \sup_k \|z(x_k)\|_2 + \sqrt{2p(x_0)}$, since L_τ is bounded by assumption and

$z(x_k) \in L_\tau$. This would imply that

$$\begin{aligned}\|c(x_k) - z(x_k)\|_2 &\geq |\|c(x_k)\| - \|z(x_k)\|| \\ &= \|c(x_k)\| - \|z(x_k)\| \\ &> \sqrt{2p(x_0)}\end{aligned}$$

for some k, contradicting the fact that $x_n \in C$ for all n. Therefore C is bounded.

To see that C is closed, consider $x_n \in C$ with $x_n \to x$. Then $c(x_n) \to c(x)$, since g is continuous, and since $c(x_n) \in \{z : D(z) \leq D(c(x_0))$ and D is continuous, and hence lower semi-continuous, this set is closed. Therefore

$$c(x) \in \{z : D(z) \leq D(c(x_0))\},$$

which is a restatement of $x \in C$.

Recently, the authors in [147] proposed a unified convergence analysis framework for minimizing smooth convex functions $f(x)$ with L-Lipschitz continuous gradient that satisfy the Polyak-Lojasiewicz (PL) inequality

$$\frac{1}{2}\|\nabla f(x)\|_2^2 \geq \mu(f(x) - \min f) \quad \forall x$$

where $\mu > 0$. Strongly convex functions satisfy the PL-inequality with the same constant, but the reverse implication does not necessarily hold. In our case, the distance function $\frac{1}{2}d_C^2(x)$ is **not** strongly convex, since $\frac{1}{2}d_C^2 = M_{\delta_C}^1$ and δ_C is not strongly convex, neither is $\frac{1}{2}d_C^2(x)$ by [Lemma 2.19, 203]. We will still manage to obtain worst-case linear convergence for gradient descent due to the PL-inequality, as we shall see.

The benefit of considering the PL-inequality is that the convergence proof for gradient descent becomes drastically simplified, which we reproduce here for the reader's convenience.

Proposition 4.2 [147] If $f(x)$ is a $\mathcal{C}^1$ function with $L-$Lipschitz continuous gradient and f satisfies the PL-inequality with constant μ and $X^* = \operatorname{argmin}_x f(x) \neq \emptyset$, then applying the gradient descent method with step size $1/L$,

$$x_{k+1} = x_k - \frac{1}{L}\nabla f(x)$$

has a global linear convergence rate

$$f(x_k) - f^* \leq \left(1 - \frac{\mu}{L}\right)^k (f(x_0) - f^*).$$

Proof An function $f(x)$ with $L-$Lipschitz continuous gradient satisfies

$$f(y) \leq f(x) + \nabla f(x)^T(y - x) + \frac{L}{2}\|y - x\|_2^2 \quad \forall y, x.$$

Plugging in $y = x_{k+1} = x_k - \frac{1}{L}\nabla f(x_k)$, $x = x_k$ gives the estimate

$$\begin{aligned} f(x_{k+1}) - f(x_k) &\leq \nabla f(x_k)^T(x_{k+1} - x_k) + \frac{L}{2}\|x_{k+1} - x_k\|_2^2 \\ &\leq -\frac{1}{2L}\|\nabla f(x_k)\|_2^2 \end{aligned}$$

and the PL-inequality yields

$$f(x_{k+1}) - f(x_k) \leq -\frac{\mu}{L}(f(x_k) - \min f).$$

Rearranging and subtracting $\min f$ from both sides gives

$$f(x_{k+1}) - \min f \leq \left(1 - \frac{\mu}{L}\right)(f(x_k) - \min f)$$

and applying this result recursively gives the linear convergence rate.

Lemma 4.3 Suppose that f is a $\mathcal{C}^1$ function with $L-$ Lipchitz continuous gradient, g is a $\mathcal{C}^1$ $\beta-$Lipschitz continuous mapping with $\gamma-$Lipschitz continuous gradient and let

$$\begin{aligned} \alpha &= \max_{x \in K} \|\nabla f(g(x))\|_2 \\ \kappa &= \max_{x \in K} \|\nabla g(x)\|_2 \end{aligned}$$

for a compact set K. Note that $\alpha, \kappa < \infty$, since K is compact and the mappings are continuous. Then $k(x) = f(g(x))$ has a $\alpha\gamma + \kappa\beta L$ Lipschitz gradient. If $g(x) = Ax - b$ is affine, then $k(x) = f(g(x))$ is $L\|A\|_2^2$-Lipschitz continuous.

Proof

$$\begin{aligned} \|\nabla(f \circ g(x)) - \nabla(f \circ g(y))\|_2 &= \|Dg(x)^* \nabla f(g(x)) - Dg(y)^* \nabla f(g(y))\|_2 \\ &= \|(Dg(x) - Dg(y))^* \nabla f(g(x)) + \\ &\quad Dg(y)^*(\nabla f(g(x)) - \nabla f(g(y)))\|_2 \\ &\leq (\alpha\gamma + \kappa\beta L)\|x - y\|_2 \end{aligned}$$

The affine case follows a similar derivation.

Proposition 4.4 Let x_0 be the initial point which belongs to a neighbourhood of $\operatorname{argmin} p$ which does not contain any local but non-global minimizers. If c is a $\mathcal{C}^1$ $\beta-$Lipschitz continuous mapping with $\gamma-$Lipschitz continuous gradient, we let

$$\begin{aligned} \alpha &:= \max_{x \in C} \|c(x) - P_{L_\tau}(c(x))\|_2 \\ \kappa &:= \max_{x \in C} \|\nabla c(x)\|_2 \end{aligned}$$

for $C = \{x : p(x) \leq p(x_0)\}$, which is compact. Suppose that

$$\lambda := \min_{x \in C} \sigma_{\min}(\nabla c(x)) > 0$$

then the gradient descent iteration with constant step size $(1/(\alpha\gamma + \kappa\beta))$ convergences linearly, i.e.,

$$p(x_k) - \min p \leq \left(1 - \frac{\lambda^2}{(\alpha\gamma + \kappa\beta)}\right)^k (p(x_0) - \min p)$$

If $c(x) = Ax - b$ is affine, we can remove the restriction on the initial point x_0 and this convergence rate reduces to

$$p(x_k) - \min p \leq \left(1 - \frac{1}{\kappa^2(A)}\right)^k (p(x_0) - \min p).$$

Proof The squared distance function $D(x)$ satisfies the PL-inequality with $\mu = 1$ by Lemma 3.4. Also note that $D(x)$ has a $1-$Lipschitz gradient, since $D(x) = M^1_{\delta_{L_\tau}}$. For the composition $\nabla D(c(x)) = c(x) - P_{L_\tau}(c(x))$, we have that the gradient is bounded by $\|\nabla D(c(x))\|_2 \leq \alpha < \infty$, where the latter inequality arises from the fact that C is compact and c is continuous. Applying Lemma 4.3, the composite function $p(x) = D(c(x))$ is $(\alpha\gamma+\kappa\beta)$-Lipchitz continuous and satisfies the PL-inequality with constant λ^2, by Lemma 3.2, and the theorem follows.

In order to compute $v'(\tau)$, we consider the case when h is a gauge, and so the variational machinery of [15] can be applied in the following considerations. If $h(z)$ is a gauge, we can upgrade the secant method to Newton's method as follows. First note that, for a gauge $\gamma(z|C)$ and the coordinate projection operator $\pi : (x, z) \mapsto z$, the composition $\Gamma(x, z) = \gamma(z|C) \circ \pi(x, z)$ is a gauge

$$\Gamma(x, z) = \gamma((x, z)|\mathbb{R}^n \times C)$$

with corresponding polar gauge

$$\Gamma(x, z)^\circ = \gamma((x, z)|\mathbb{R}^n \times C^\circ)$$

where $C^\circ = \{v : \langle v, x\rangle \leq 1 \ \forall x \in C\}$ is the anti-polar of C [Theorem 14.5, 217]. This can also be written as $\Gamma(x, z)^\circ = h^\circ(z)$.

Theorem 4.5 Suppose that h is a gauge, c satsfies the hypotheses of Proposition 4.4, then $v_{lin}(\tau) = v(\tau)$, where

$$v_{lin}(\tau) = \min_x \frac{1}{2} d^2_{L_\tau}(c(x^*) + \nabla c(x^*)(x - x^*)) \tag{4.9}$$

and $x^* \in \operatorname{argmin}_x \frac{1}{2} d^2_{L_\tau}(c(x))$. If also it holds that

$$v_{lin}(\tau + \Delta\tau) + O(|\Delta\tau|^2) = v(\tau + \Delta\tau) \tag{4.10}$$

for all $\Delta\tau$ sufficiently small, then

$$v'(\tau) = -h^\circ(z - c(x)) \tag{4.11}$$

where (x, z) are such that $(x, z) = \arg\min \frac{1}{2}\|c(x) - z\|_2^2 + i_{L_\tau}(z)$. Condition (4.10) always holds when $c(x) = Ax - b$ is affine.

Proof First note that if $x^* \in \operatorname{argmin}_x \frac{1}{2} d^2_{L_\tau}(c(x))$, then x^* satisfies

$$\nabla c(x^*)^T (c(x^*) - P_{L_\tau}(c(x^*))) = 0.$$

Computing the gradient of the objective function in the linearized problem in (4.9) yields

$$\nabla c(x^*)^T (c(x^*) + \nabla c(x^*)(x - x^*) - P_{L_\tau}(c(x^*) + \nabla c(x^*)(x - x^*)))$$

and this gradient vanishes when $x = x^*$. Since the objective satisfies the PL-inequality, by [147] x^* is a global minimizer of (4.9) and therefore $v_{lin}(\tau) = v(\tau)$.

If also we have that $v_{lin}(\tau + \Delta\tau) + O(|\Delta\tau|^2) = v(\tau + \Delta\tau)$, then the difference quotient

$$\begin{aligned} \frac{v(\tau + \Delta\tau) - v(\tau)}{\Delta\tau} &= \frac{v_{lin}(\tau + \Delta\tau) + O(|\Delta\tau|^2) - v_{lin}(\tau)}{\Delta\tau} \\ &\to v'_{lin}(\tau) \end{aligned}$$

as $\Delta\tau \to 0$.

We can write

$$f(x, z, \tau) = \frac{1}{2} \left\| \begin{bmatrix} \nabla c(x^*) & -I \end{bmatrix} \begin{bmatrix} x \\ z \end{bmatrix} - (c(x^*) + \nabla c(x^*)x^*) \right\|_2^2 + \delta((x, z, \tau) | \operatorname{epi} \Gamma(x, z))$$

where $v_{lin}(\tau) = \min_{x,z} f(x, z, \tau)$. It follows from [Theorem 6.2, 15] that if

$$(\hat{x}, \hat{z}) \in \operatorname*{argmin}_{x,z} \frac{1}{2}\|c(x^*) + \nabla c(x^*)(x - x^*) - z\|_2^2 + i_{L_\tau}(z)$$

we must have that

$$\begin{aligned} v'_{lin}(\tau) &= h^\circ(\hat{z} - (c(x^*) + \nabla c(x^*)(\hat{x} - x^*))) \\ &= h^\circ(\hat{z} - c(x^*)) \end{aligned}$$

where the second line follows from the fact that

$$x^* \in \operatorname*{argmin}_x \frac{1}{2} d^2_{L_\tau}(c(x^*) + \nabla c(x^*)(x - x^*)).$$

Note that we have $\hat{z} = P_{L_\tau}(c(x^*))$ in this case, completing the proof.

4.4 Numerical Examples

4.4.1 Cosparsity

The classical setup for compressed sensing aims to recover a signal $x \in \mathbb{R}^n$ from a measurement vector $b = Ax \in \mathbb{R}^{m \times n}$, where $m \ll n$. In order to ensure successful recovery, the signal x is assumed to be sparse with s unknown entries, although the location of these entries is unknown. The strongest known theoretical guarantees for recovery are for random matrices, and in particular when A has Gaussian i.i.d. entries, $m = O(s \log(n/s))$ samples are sufficient to recover x with high probability [58, 53]. If x is $s-$sparse in some dictionary $D \in \mathbb{R}^{n \times p}$, i.e., $x = D\alpha$ where $\alpha \in \mathbb{R}^p$ has $s-$ nonzeros, with A being our measurement operator, we can recover x by through solving the optimization program

$$\begin{aligned} &\min_{\alpha} \|\alpha\|_p \\ &\text{such that } \; AD\alpha = b \end{aligned} \tag{4.12}$$

where $p = 0$ is the non-convex ℓ_0 pseudnorm and $p = 1$ is its convex relaxation. This is the synthesis model of compressed sensing studied in [90, 178, 57]. An alternative signal model is the cosparsity model [186], which stipulates that Ωx is sparse for some dictionary $\Omega \in \mathbb{R}^{p \times n}$. A zero value in Ωx stipulates that the signal x is perpendicular to the corresponding row of Ω, and hence in totality x lies in a corresponding union of subspaces. These two models are equivalent only when the dictionary Ω is an orthogonal basis, but in general they produce different solutions. If Ωx is $k-$sparse, x is referred to as $l = p - k$ cosparse, as we are interested in having this quantity be as large as possible.

The associated optimization problem is

$$\begin{aligned} &\min_{x} \|\Omega x\|_p \\ &\text{such that } \|Ax - y\|_q \leq \sigma, \end{aligned} \tag{4.13}$$

where $p = 0$ is the original nonconvex problem and $p = 1$ is its corresponding convexification and $q \in [1, \infty]$ specifies a norm for the data misfit. The choice of p will have a significant impact on our solution as we shall see.

We recast this problem in our composite convex framework by introducing the variables $r = Ax - y$, $z = \Omega x$, which gives

$$\begin{aligned} &\min_{x} \|z\|_p \\ &\text{such that } \; \|r\|_q \leq \sigma \\ &\qquad\qquad Ax - y - r = 0 \\ &\qquad\qquad \Omega x - z = 0. \end{aligned}$$

The value function is therefore

$$\begin{aligned} v(\tau) = \min_{x,z,r} \; & \frac{1}{2}\|Ax - y - r\|_2^2 + \frac{1}{2}\|\Omega x - z\|_2^2 \\ \text{such that} \; & \|r\|_q \leq \sigma \\ & \|z\|_p \leq \tau. \end{aligned}$$

Projecting out (z, r) yields

$$\begin{aligned} z(x) &= P_{\|\cdot\|_p \leq \tau}(\Omega x) \\ r(x) &= P_{\|\cdot\|_q \leq \sigma}(Ax - y), \end{aligned}$$

which gives the final problem that we solve

$$v(\tau) = \min_x \frac{1}{2}\|Ax - y - r(x)\|_2^2 + \frac{1}{2}\|\Omega x - z(x)\|_2^2. \tag{4.14}$$

A reference MATLAB implementation for evaluating the objective and gradient in (4.14) is shown in Listing (4.1) for the noiseless case where $\sigma = 0$. Note that to solve ℓ_0 minimization problem, one can merely replace the ℓ_1 projection with ℓ_0 projection, i.e., hard thresholding.

```
function [f,g] = cosparsity_obj(A,b,x,Omega,tau)
  r = A*x-b;
  z = Omega*x;
  y = NormL1_project(z,tau);
  z = z-y;
  f = 0.5*norm(r)^2 + 0.5*norm(z)^2;
  if nargout >=2
      g = A'*r + Omega'*z;
  end
end
```

Listing 4.1 MATLAB code for computing the VELVET objective and gradient for the cosparsity subproblem

Since the mapping c from (4.1) is linear in this case, by Theorem 4.5, we have that

$$v'(\tau) = -\|P_{\|\cdot\|_p \leq \tau}(\Omega x) - \Omega x\|_{p^*}$$

where $\|\|_{p^*}$ is the dual norm to $\|\|_p$.

Chambolle-Pock

We compare our method to the Chambolle-Pock method [64], which solves the problem

$$\min_x F(Kx) + G(x)$$

where F, G are two convex, typically non-smooth functions, via the following iterations

$$\begin{aligned} y_{n+1} &\leftarrow P^{\gamma}_{F^*}(y_n + \gamma K \bar{x}_n) \\ x_{n+1} &\leftarrow P^{\omega}_{G}(x_n - \omega K^T y_{n+1}) \\ \bar{x}_{n+1} &\leftarrow x_{n+1} + \theta(x_{n+1} - x_n). \end{aligned}$$

Note that the ℓ_1 analysis problem (4.13) fits in to this framework with $F = \|\cdot\|_1$, $K = \Omega \in \mathbb{R}^{m\times n}$, and $G(x) = i_{\|A\cdot - b\|_2 \leq \sigma}(x)$. The parameters γ, ω should be such that $\gamma, \omega = 1/\|K\|_2$, which is equal to 1 for a tight frame operator. Note here that $F^*(y) = i_{\|\cdot\|_\infty \leq 1}(y)$, so $P^{\gamma}_{F^*}(y)$ is merely projection on to $[-1, 1]^m$. In the noise-free case, $\sigma = 0$, since A restricts the values of the vector at specified indices Γ, the P^{ω}_{G} term becomes

$$P^{\omega}_{G}(y) = \begin{cases} b_i & \text{if } i \in \Gamma \\ y_i & \text{otherwise.} \end{cases}$$

Split Bregman method

We also compare our method to the split Bregman method of [48], which solves the ℓ_1 analysis problem via the following iterations

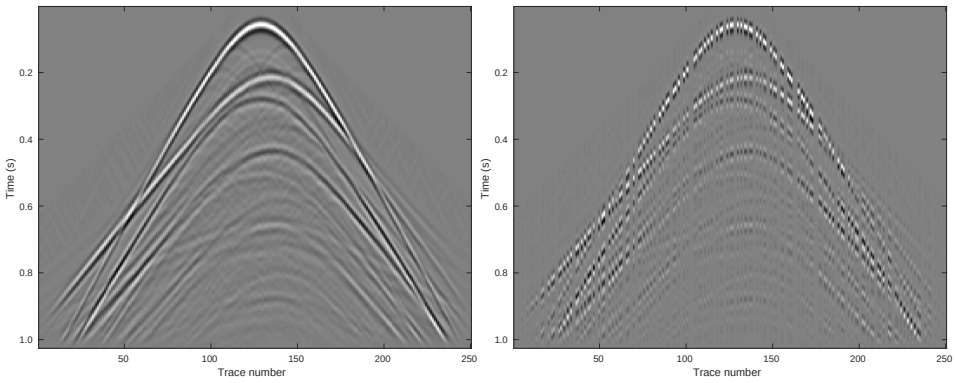
$$\begin{aligned} x^{k+1} &= \arg\min_x \frac{\mu}{2}\|Ax - b + c^k\|_2^2 + \frac{\lambda}{2}\|\Omega x - d^k + f^k\|_2^2 \\ d^{k+1} &= \arg\min_d \|d\|_1 + \frac{\lambda}{2}\|d - \Omega x^{k+1} - f^k\|_2^2 \\ f^{k+1} &= f^k + \delta_f(\Omega x^{k+1} - d^{k+1}) \\ c^{k+1} &= c^k + \delta_c(Ax^{k+1} - b) \end{aligned}$$

In what follows, Ω is the curvelet operator [54], which is not an explicit matrix. As such, we use LSQR [198] to solve the least-squares problem for x^{k+1} with a relative tolerance of 10^{-3}. We run this algorithm for 200 iterations and use an early stopping criteria to quit if the relative residual in the subproblem for x^{k+1} is approximately one. Curvelets form a tight frame, i.e., $\Omega^T\Omega = I$, and LSQR terminates within a relatively small number of iterations.

Seismic Data Interpolation

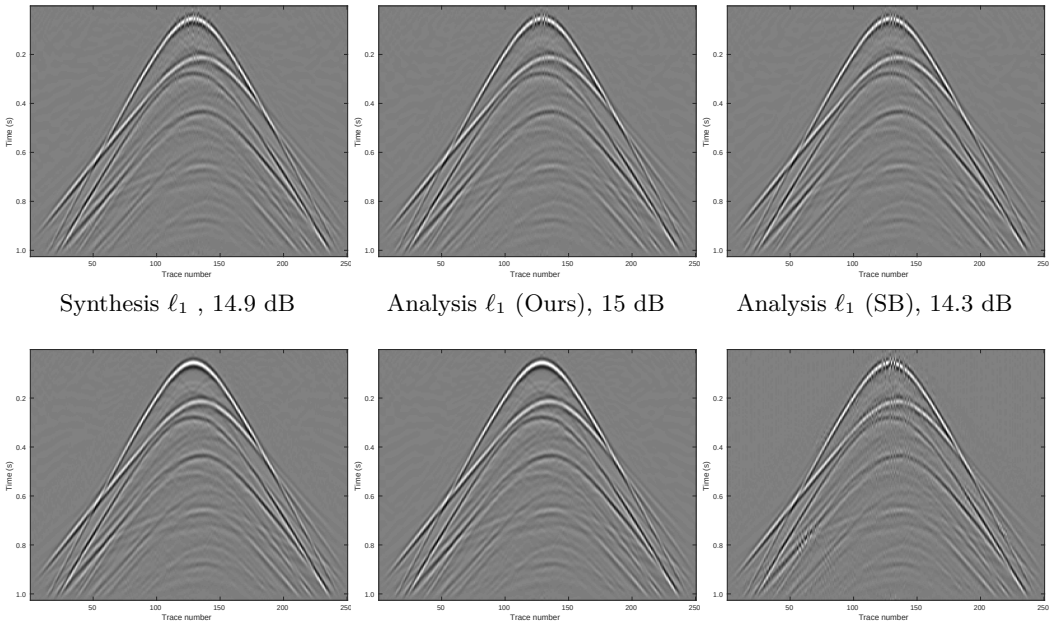
We test this approach on a realistic seismic data set by first randomly removing 50% of the traces from a single shot with 256 receivers and 1024 time samples. We compare the synthesis formulation in (4.12), solved with SPGL1 with a Curvelet sparsity dictionary, with the analysis formulation in (4.13) with both $p = 0$ and $p = 1$. We also use Chambolle-Pock method previously discussed, run for 2000 iterations, as well as the Split Bregman algorithm run for 200 iterations. For the $p = 0$ case, we also consider the GAP method [186] that directly tackles the non-convex analysis problem. The data is shown in Figure 4.2 and the results are show in Figure 4.3 and Table 4.1. As expected, the ℓ_0 methods perform much better than the ℓ_1 methods, although the corresponding theoretical guarantees are less developed in the former

case compared to the latter. Our proposed VELVET algorithm outperforms the GAP method on this small example, both in terms of recovery time and quality, and is much simpler to implement.



True Data Input Data

Figure 4.2 True and subsampled signal, 50% receivers removed



Synthesis ℓ_1 , 14.9 dB Analysis ℓ_1 (Ours), 15 dB Analysis ℓ_1 (SB), 14.3 dB

Gap ℓ_0 , 23.4 dB Analysis ℓ_0, 23.7 dB Analysis ℓ_1, CP, 10.0 dB

Figure 4.3 Recovery of a common source gather (fixed source coordinates). Displayed values are SNR in dB.

Method	Test SNR (dB)	Time (s)
Synthesis ℓ_1 (SPGL1)	14.9	112
Analysis ℓ_1 (Ours)	15.0	77.2
Analysis ℓ_1 (CP)	10.0	460
Analysis ℓ_1 (SB)	14.3	188
Analysis ℓ_0 (Ours)	23.7	77.4
Analysis ℓ_0 (GAP)	23.4	117

Table 4.1 Cosparse recovery results

Robust Total Variation Deblurring

A classic inverse problem is deblurring where one seeks to estimate a sharp image from a blurred, noisy version. We model this scenario as

$$y = Hx + n$$

where H is the blurring operator, x is the true image, n is the additive noise, and y is the noisy blurred image. H is often modeled as a convolution with a Gaussian (or similar) kernel, which exhibits quickly decaying singular values. As such, simple least-squares inversion is numerically insufficient to estimate the true signal since the components of the noise in the subspace generated by the small singular values of H become greatly amplified in the reconstruction, as shown in Figure 4.5. Even in the noiseless-case, the small singular values, which often correspond to the details of the image, are difficult for a Krylov method to invert, resulting in a low-fidelity reconstruction.

One popular approach to regularize this problem is to use robust total-variation regularization [219], which corresponds to solving

$$\begin{aligned} &\min_x \|x\|_{TV} \\ &\text{such that } \|Hx - y\|_1 \leq \sigma \end{aligned} \tag{4.15}$$

where σ is an estimate of the noise level of n, $\|n\|_1$, and

$$\|x\|_{TV} = \sum_i \sum_j \sqrt{(x_{i+1,j} - x_{i,j})^2 + (x_{i,j+1} - x_{i,j})^2}$$

is the isotropic TV (pseudo) norm. We can equivalently write $\|x\|_{TV} = \|Dx\|_{1,2}$, where D is a finite difference matrix stacking the horizontal and vertical derivative matrices and $\|X\|_{1,2}$ is the mixed $\ell_{1,2}$ norm that computes the column norms of the matrix X. The TV norm promotes cartoon-like structures in images, since the ℓ_1 component of the norm tends to set the gradient of the image at a large number of points to zero. This problem is a straightforward instance of the the analysis

program in (4.13), where $p = (1, 2)$ and $q = 1$, and so we employ the same code as previously.

To validate our composite convex framework, we deblur the standard "Cameraman" image with the following parameters. The blurring matrix is a convolution of a Gaussian kernel with width 9 and standard deviation 4 and we update the τ parameter at most 20 times, with the results displayed in Figure 4.4.

True Image | Blurred Image, SNR = 15.0 dB | TV Regularized Image, SNR = 30.0 dB, T = 88.2s

Figure 4.4 TV deblurred image in the noise-free case.

Compared to handling (4.13) directly, methods that solve the penalized formulation of the TV problem (4.15)

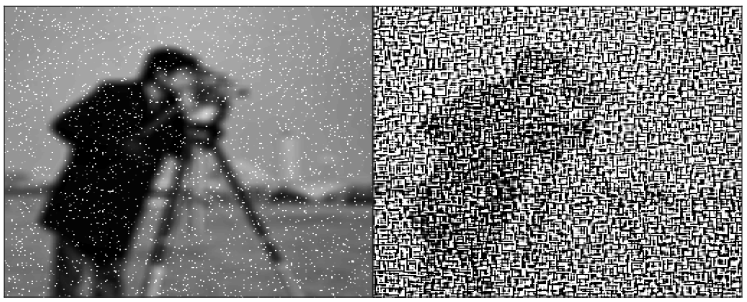
$$\min_x \|Hx - y\|_1 + \lambda \|x\|_{TV} \tag{4.16}$$

through the Alternating Direction of Multipliers (ADMM) method or other such splitting techniques [66, 65] must have an accurate estimate of the λ parameter. There is little theoretical insight in to choosing this parameter apriori and the optimal λ parameter must be estimated on a per-image basis. When coupled with the other parameters one must choose in order to ensure the ADMM method converges sufficiently quickly, the number of parameters that must be known becomes a high dimensional search space. Our method, on the other hand, does not have such extraneous parameters to estimate due to the inherent secant/Newton method for updating τ.

We compare our approach to noisy TV-based deblurring to those that solve (4.16), such as [242, 23, 66]. The results in the aforementioned articles all involve a degree of hand-tuning of the λ parameter for each input image, which is untenable when applying such techniques to a given image, in particular when the true image is unknown. We compare our method to the approach of [242], as the software provided by the other references proved too fragile to make the necessary modifications to construct a baseline comparison. Applying both our method and the method of [242] to deblur and denoise the Cameraman image results in the images in Figure 4.5. In this example, the same blurring kernel has been applied

with the addition of high amplitude but sparse noise to 5% of the pixels in the image.

In order to fairly compare this approach to our own, we search over a range of 50 λ parameters ranging logarithmically in $[10^{-2}, 10^{2}]$. Unfortunately the dependence of the SNR on the λ parameter the premature stopping criteria employed by these algorithms does not allow us to simply halt our increase of λ when the SNR has reached its true peak. As such, we stop increasing λ at iteration i when $SNR(x_i) < 0.8 * max_{j=1,\ldots,i} SNR(x_j)$. All of the other parameters are set to their default values. Even despite this optimal λ choice, the resulting image is very cartoonish and not particularly sharp, which is consistent with the results in [242]. In reality, if we did not have the reference image available, it would be exceedingly difficult to choose λ optimally in this fashion for these methods and we would have to rely on the ad-hoc "eyeball norm". Our method, on the other hand, does not require the choice of this parameter and reaches a satisfactory solution of its own accord, albeit in a slightly longer computational time due to the need to solve our subproblems relatively accurately. The least-squares solution, as predicted, performs poorly as the noise is amplified through the inversion of the subspace corresponding to the small singular values of B, although one could add damping or terminate the Krylov method earlier in order to mitigate some of these effects.



Blurred noisy image, SNR = 10.6 dB

Least-squares solution, SNR = -6.74 dB, T = 1.34s

Convex composite solution, SNR = 28.1 dB, T = 134s

[242] solution, SNR = 18.2 dB, T = 55.4s

Figure 4.5 TV deblurred image in the noisy case.

Audio Declipping

Another promising application of cosparsity-based regularization is *declipping*. Clipping, or magnitude saturation, occurs when the magnitude of a signal is greater than the range of values the acquisition device is able to distinguish. For audio signals in particular, the perception of clipping may range from hearing "clicks and pops" to hearing additive noise, whereby the discontinuities introduced by the clipping result in a large number of apparent harmonics. We consider the true audio signal represented as a vector $x^* \in \mathbb{R}^n$, where n is the number of time samples, and its clipped version $y \in \mathbb{R}^n$ is induced from the *hard clipping* operator M, $y = M(x)$ with

$$M(x^*)_i = \begin{cases} x_i^* & \text{for } |x_i^*| \leq \delta \\ \text{sign}(x_i^*)\delta & \text{otherwise} \end{cases}.$$

Here $\tau > 0$ is the clipping level. This observation model is idealized yet nevertheless convenient for distinguishing the clipped parts of the signal from the non-clipped samples by means of the highest amplitude threshold present. We split the index set $[n] = \{1, 2, \ldots, n\}$ in to a partition of three sets $[n] = I_t \cup I_+ \cup I_n$. Here I_t denotes the unclipped portions of the signal, I_+ are the indices of the signal clipped to the value $+\tau$ and similarly for I_-. Taken together, these three indices induce constraints on any candidate estimate x of the true signal via the inequalities

$$\begin{aligned} x_i &= y_i \text{ for } i \in I_t \\ x_i &\geq y_i \text{ for } i \in I_+ \\ x_i &\leq y_i \text{ for } i \in I_-. \end{aligned}$$

We denote the set of signals $x \in \mathbb{R}^n$ satisfying the above inequalities as $S(y)$. Merely satisfying these constraints is clearly not sufficient to reconstruct the original signal. As such, various regularization techniques have been previously proposed to declip or restore the original signal, including minimizing higher order derivative energy [120], sparsity [4, 233, 155], and cosparsity [154, 153]. The ℓ_p-cosparse model, as in the seismic case, attempts to solve the problem

$$\begin{aligned} &\min_x \|\Omega x\|_p \\ &\text{s.t.} \quad x \in S(y), \end{aligned} \tag{4.17}$$

where $p = 0$ or $p = 1$.

The analogous sparsity model is

$$\begin{aligned} &\min_x \|x\|_p \\ &\text{s.t.} \quad Dx \in S(y). \end{aligned}$$

The pointwise constraints are much more challenging to satisfy in this case, and so we focus our efforts on the cosparsity case.

The analysis dictionary we employ for natural audio signals is the Gabor transform (i.e., the short time Fourier transform) [209], composed of time-windowed complex exponentials with varying temporal and frequency shifts. Despite its redundancy, the Gabor transform Ω is a tight frame, and so $\Omega^H \Omega = I$. Our implementation is provided by [201]. We can solve problem (4.17) via the VELVET method, where the value function is written as

$$\begin{aligned} v(\tau) = \min_x \ & \frac{1}{2}\|\Omega x - z(x)\|_2^2 \\ \text{s.t.} \ \ & x \in S(y), \end{aligned}$$

and $z(x) = P_{\|\cdot\|_p \leq \tau}(\Omega x)$. Here the constraints $S(y)$ are pointwise, which leads to the projector

$$P_{S(y)}(\hat{y})_i = \begin{cases} y_i & \text{for } i \in I_t \\ \max(y_i, \tau) & \text{for } i \in I_+ \\ \min(y_i, -\tau) & \text{for } i \in I_- \end{cases}.$$

We use the projected quasi-Newton method of [223] to evaluate $v(\tau)$ and the secant method to update the τ parameter.

Our signals are normalized to have magnitude 1 and clipped to values ranging from $0.05 \leq \tau \leq 0.9$. To assess the quality of our recovery using VELVET, we use the signal-to-distortion ratio [154]

$$SDR(z) = -20 \log_{10} \left(\frac{\|x^*_{I_+ \cup I_-} - z_{I_+ \cup I_-}\|_2}{\|x^*_{I_+ \cup I_-}\|_2} \right),$$

which measures the deviation from the true signal on $I_+ \cup I_-$, the set of indices where the signal has been clipped.

The results from aggressively clipping the reference "Glockenspiel" signal to $\delta = 0.05$ are shown in Figure 4.6, resulting in 74% of the signal values being clipped. Despite this agressive threshold, ℓ_0-based cosparsity interpolation is able to recover a fairly reasonable estimate of the true signal, whereas the ℓ_1 convex relaxation performs very poorly, hardly improving the input SDR whatsoever. The theoretical underpinnings behind this large gap in performance between ℓ_0 and ℓ_1-based cosparsity are an interesting open topic for future research. Intuitively, as our sampling operator decreases the amplitudes of the signal to lie in a given range, the corresponding inner products with the Gabor frame elements is reduced in magnitude as well. Therefore the ℓ_1 norm of the coefficients of the clipped signal is smaller than the original signal, which implies that our original signal is no longer a solution to the optimization problem in this case. Compared to the performance of algorithms that attempt to solve (4.17) with $p = 0$, such as CoDec-HT [153] or A-SPADE [154], our method has a natural mechanism for increasing the sparsity level k. Increasing the sparsity by a fixed amount, as in the aforementioned algorithms, cannot terminate until a number of iterations proportional to the true sparsity of the signal, which may be large. On the other hand, the secant method employed by our VEL-

VET algorithm makes much faster progress towards the true sparsity automatically. In these examples, for instance, we update the τ parameter at most 10 times. Although ℓ_0 analysis performs significantly better than the corresponding ℓ_1 analysis reconstruction, the computational time for the former is significantly longer as τ increases, as shown in Table 4.2. Intuitively, this observation corroborates with the fact that ℓ_0-minimization is an NP-hard problem even in the simplest case [187].

Method	Number of parameter updates	SDR (dB)	Time (s)
Analysis ℓ_1	8	2.36	33
Analysis ℓ_1	10	2.36	45
Analysis ℓ_0	8	12.7	180
Analysis ℓ_0	10	14.6	377

Table 4.2 Audio declipping produces much better results with $p = 0$ compared to $p = 1$, but the computational times become daunting as the ℓ_0-norm increases.

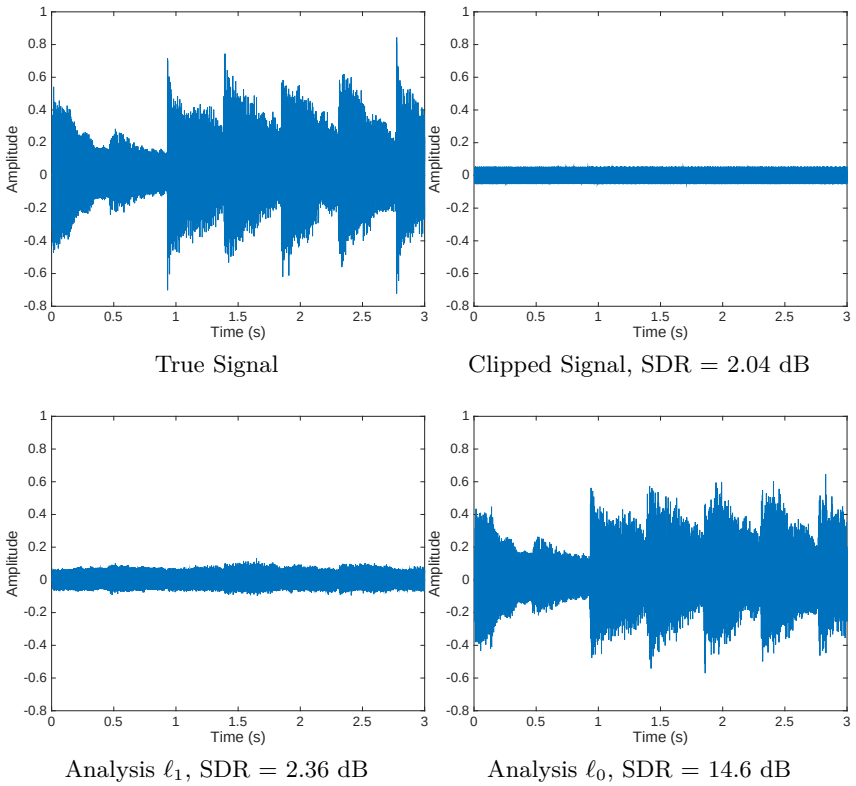


Figure 4.6 Declipping the "Glockenspiel" audio file. The first three seconds are shown.

4.4.2 Robust Tensor PCA / Completion

In this section, we solve the robust tensor completion problem introduced in Section 4.1

$$\min_{x \in \mathcal{M}} \|\mathcal{A}(\phi(x)) - b\|_1. \tag{4.18}$$

The associated value function for (4.18) is

$$\phi(\tau) = \min_{x \in \mathcal{M}} \frac{1}{2}\|\mathcal{A}(\phi(x)) - b - z(x)\|_2^2.$$

with

$$\begin{aligned} z(x) = \arg\min_{z} \ & \frac{1}{2}\|\mathcal{A}(\phi(x)) - b - z\|_2^2 \\ \text{such that } & \|z\|_1 \leq \tau. \end{aligned}$$

We only have to 'plug in' the soft-thresholded residual $z(x)$ when computing the full data residual and the rest of the codes use the standard HT optimization framework. Note that $z_0(x) = 0$ for all x, so $\phi(0)$ corresponds to the standard HT tensor completion problem. Each subproblem associated to an evaluation of $\phi(\tau)$ is warm-started with the previous parameter estimate x and converges quickly, due to the Gauss-Newton method used in evaluating ϕ. Although the mapping ϕ is nonlinear, we note that the ℓ_1 norm is a gauge we use the corresponding formula

$$\phi'(\tau) = -\|\mathcal{A}(\phi(x)) - b - z(x)\|_\infty,$$

which appears to hold numerically in our tests. Note that the overall problem in (4.18) is non-convex, since the image of the mapping $x \mapsto \phi(x)$ is the non-convex manifold of Hierarchical Tucker tensors.

To validate this approach, we interpolate a single frequency slice of data generated from the BG Compass model with 68 x 68 sources and 201 x 201 receivers. We remove 75% of the receivers randomly and, to 5% of the remaining receivers, add high amplitude noise with energy equal to that of the remaining signal. We perform 5 updates of τ, i.e., (4.7), starting from $\tau = 0$ and use a relative function stopping tolerance of 0.0005 when computing $\phi(\tau)$ and a maximum inner iteration count of 20. We compare this approach to standard HT completion without the ℓ_1 penalty, i.e., merely computing $\phi(0)$. The results are displayed in Figure 4.7. We also compare this result to the Huber loss function (3.4), i.e., we solve

$$\min_x \sum_{i=1}^{n} \kappa_\delta(\mathcal{A}(\phi(x))_i - b_i)$$

for a variety of parameters δ. For the Huber loss, the best performing δ is similar in terms of computational time and quality to the result ℓ_1, although the ℓ_1 still performs slightly better in terms of recovery quality. Knowledge of this best-case δ is not determined apriori, however, and departure from this optimal parameter quickly degrades the test SNR, as shown in Table 4.4. The one-norm minimization

in this case does not have any additional hyperparameters to estimate compared to the Huber loss function yet still produces high quality results.

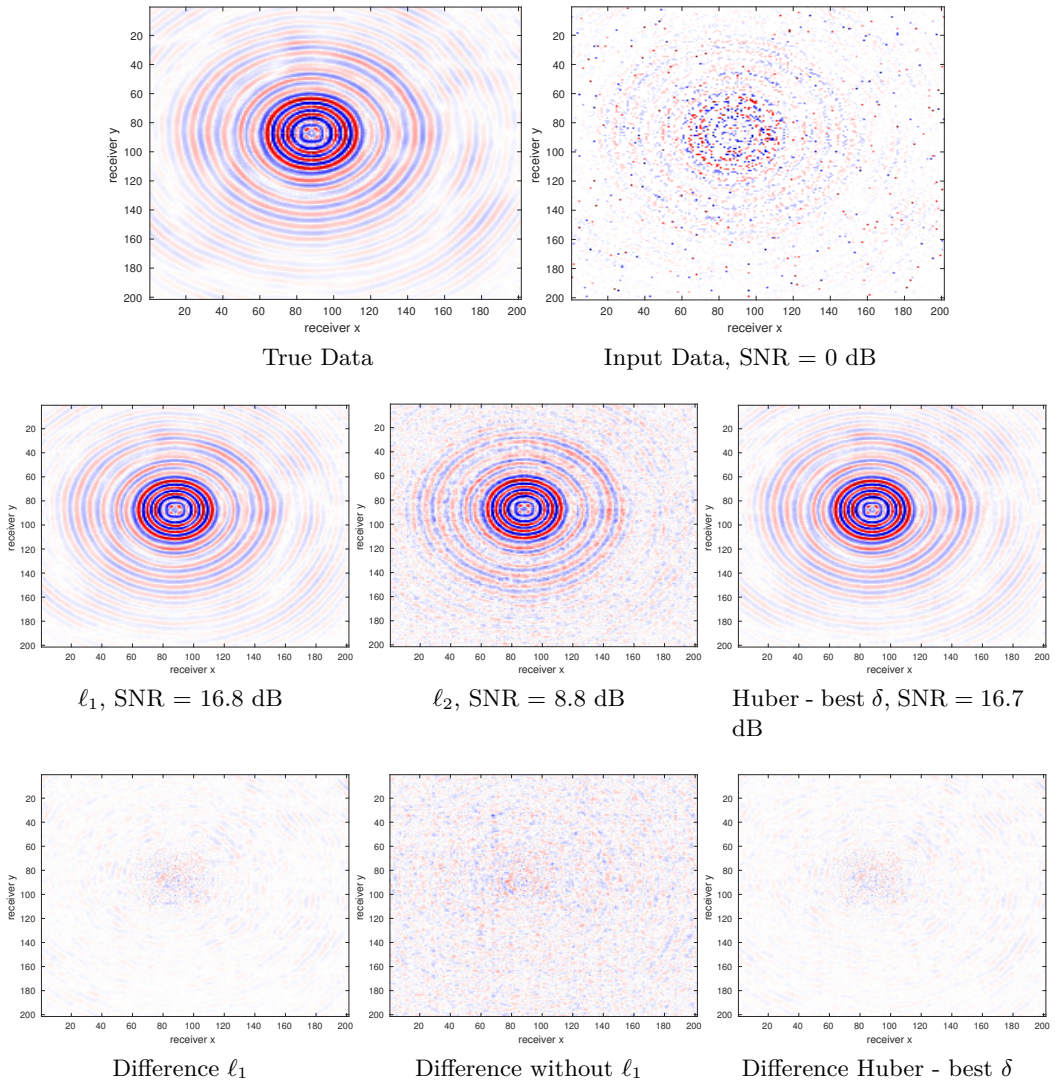


Figure 4.7 Recovery of a common source gather.

	Test SNR (dB)	Time (s)
With ℓ_1	16.2	1072
Without ℓ_1	7.68	632
Huber, best	15.9	1003

Table 4.3 Summary of recovery results

Huber δ	Test SNR (dB)	Time (s)
10^{-6}	10.1	1578
$5 \cdot 10^{-6}$	13.4	1657
10^{-5}	15.5	1594
$5 \cdot 10^{-5}$	15.9	1003
10^{-4}	14.7	926
$5 \cdot 10^{-4}$	8.32	928
10^{-3}	7.68	984.6

Table 4.4 Huber recovery performance versus δ parameter

4.4.3 One-bit Compressed Sensing

The standard model of compressed sensing described in Section 4.4.1 assumes infinite-bit precision in the measurement vector. Practitioners often neglect the differences between the infinite precision model and its standard 64−bit floating point numerical discretization. When the measurement device quantizes the measurements to a precision other than 64−bits, this precision discrepancy can introduce unintended errors in the recovered signal if not properly accounted for. The authors in [39] consider the extreme case of quantization, where the magnitude information of b is discarded and one only has access to the signs of the measurements, $\text{sign}(b)$. Unlike in the classical compressed sensing regime, the one-bit quantization results in an inherent loss of information. Roughly speaking, if one stipulates that a point lies on a number of hyperplanes as in standard CS, with a sufficient number of these constraints, there will be at most one point that satisfies them. On the other hand, if one stipulates that a point lies in a number of halfspaces associated with these hyperplanes, the set of points that satisfy these constraints, irrespective of their number, will have nonzero volume. As such, in the one-bit CS paradigm, there is inherent ambiguity introduced by the quantization process that one can merely ask for an approximation to the true x rather than recovering it exactly. More precisely, from [Lemma 1, 141], if $x \in \cup_{i=1}^{L} S_i$ is in a union of L subspaces S_i, each of dimension K, and we acquire $M \geq 2K$ 1-bit measurements in the vector b, then b contains at most $K \log_2(2Me/K) + \log_2(L)$ information bits and consequently the error for *any* reconstruction decoder satisfies $\epsilon_{\text{opt}} \geq \Omega(K/M)$. When $K \ll M$, the error of any one-bit decoder can perform no better than $\Omega(1/M)$.

The work of [202] proves that when A is i.i.d. Gaussian, we can recover an approximation of x by solving the following optimization program

$$\begin{aligned} \min_x \ & \|x\|_p \\ \text{such that} \ & \text{sign}(Ax) = y \\ & \|Ax\|_1 = m \end{aligned}$$

when $p = 1$, although we can consider the case when $p = 0$ as well. As we are merely measuring the signs of Ax, we cannot distinguish between x and cx for any $c > 0$ and thus cannot determine the norm of x from the measurements. The normalization $\|Ax\|_1 = m$ precludes x from being zero, but m can be set to any other positive constant in this case.

Rewriting $\text{sign}(Ax) = y$ as $Ax \odot y \geq 0$ was suggested in [202] as being an equivalent constraint, with the latter being easier to express as constraints for a linear program. It should be noted that this equivalence is not entirely correct, as the condition $Ax \odot y > 0$ results in the correct signs for the recovered signal. This results in a constraint set which is open, however, and there is no guarantee that the resulting optimization program has a solution. By closing the constraint as $Ax \odot y \geq 0$, we expand the constraint set to include points that may not exactly fit the data, but the resulting program has a solution and we can compute it in polynomial time. With this point in mind, we move to solve this program with the convex composite method. The constraint $Ax \odot y \geq 0$ is equivalent to $\|\max(-y \odot Ax, 0)\|_1 = 0$, which results in the equivalent optimization problem

$$\begin{aligned} \min_{x,r} \ & \|x\|_1 \\ \text{such that} \ & \|\max(r, 0)\|_1 = 0 \\ & \|r\|_1 = m \\ & r = -y \odot Ax \end{aligned}$$

The resulting value function reads as

$$\begin{aligned} v(\tau) = \min_{x,r} \ & \frac{1}{2}\| - y \odot Ax - r\|_2^2 \\ \text{such that} \ & \|\max(r, 0)\|_1 = 0 \\ & \|r\|_1 = m \\ & \|x\|_1 \leq \tau \end{aligned}$$

and the variable projected $r(x)$ is

$$\begin{aligned} r(x) = \operatorname*{argmin}_r \ & \frac{1}{2}\| - y \odot Ax - r\|_2^2 \\ \text{such that} \ & \|\max(r, 0)\|_1 = 0 \\ & \|r\|_1 = m \end{aligned}$$

which is equivalent to

$$\begin{aligned} r(x) = \operatorname*{argmin}_r \ & \frac{1}{2}\| - y \odot Ax - r\|_2^2 \\ \text{such that} \ & r \leq 0 \\ & \|r\|_1 = m. \end{aligned}$$

We solve this projection by the simple $O(n \log n)$ algorithm in [92] for positively constrained ℓ_1 minimization with reversed signs.

In what follows, we use three measures of quality when assessing the success or failure of a recovered signal. The Hamming Distance [141] between two vectors in the Boolean cube $\overline{a}, \overline{b} \in \{-1, 1\}^m$ is defined as

$$d_H(\overline{a}, \overline{b}) := \frac{1}{m} \sum_{i=1}^{m} \overline{a}_i \oplus \overline{b}_i$$

where $a \oplus b$ is the XOR operation between a, b with $a \oplus b = 0$ if $a = b$ and 1 otherwise, when $a, b \in \{-1, 1\}$. This distance satisfies $d_H \in [0, 1]$ and is used to compare $d_H(\text{sign}(A\hat{x}), y)$ at an estimated solution $\hat{x}$. Also introduced in [141] is a distance measure in signal space, defined as

$$d_S(x, y) := \frac{1}{\pi} \arccos\langle x, y \rangle$$

for vectors x, y with $\|x\|_2 = \|y\|_2 = 1$ and satisfies $d_S \in [0, 1]$. We consider the inverse of this distance $\dfrac{1}{d_S(x, y)}$ as a score for a candidate solution (i.e., higher is better). We also use the traditional ℓ_2 distance to the true signal, $\|x - x_{\text{true}}\|_2$ as a quality metric merely for presentational purposes, as the ℓ_2 difference can be mapped to d_S through the polarization identity.

We compare this approach to the BIHT algorithm of [141], which involves an iterative gradient descent procedure followed by thresholding on to the top k components of the signal. Fixing $n = 1000$ and $k = 100$, we consider two instances of signals, sparse signals which have k nonzero coefficients with $N(0, 1)$ entries and compressible signals that are k-sparse satisfying $x_{\sigma(i)} \propto i^{-2}$. Here σ indexes the nonzero coefficients of x. We consider the subsampling case with $m = 500$ as well as the supersampling case with $m = 2000$ and we average each recovery experiment over 100 trials. The BIHT algorithm is run for at most 15000 iterations and is terminated early when the number of sign mismatches between the predicted and observed data is zero.

The BIHT algorithm significantly outperforms the composite-convex approach when the signal is purely sparse, as shown in Tables 4.5 and 4.7. This strong performance is very much dependent on the knowledge of the true sparsity k, however, and when this sparsity level is unknown or difficult to estimate, the performance of the estimator drops significantly, as when we use the parameter $4k$ instead of k for this algorithm. As our method does not assume knowledge of the underlying sparsity, it is unaffected by this parameter, although is significantly less reliable than when k is known. This strong dependence of the performance of BIHT on k also manifests itself when considering the compressible signal case. Tables 4.6 and 4.8 show the decreased performance of BIHT in this case compared to the improved performance of our approach for compressible signals in both the oversampling and undersampling scenarios. In most instances, the ℓ_0 norm outperforms the corresponding ℓ_1

relaxation for the convex composite method, which is expected. The sparsity level that we employ here is rather high relative to the ambient dimension and is thus a challenging case from a compressed sensing point of view. The expected error behaves as [141]

$$\epsilon = O\left(\sqrt{\frac{k}{m}\log\left(\frac{mn}{k}\right)}\right)$$

which is uninformative in the subsampled case and around 0.7 in the oversampled case, which is pessimistic given the experiments below.

	$\|x - x_{\text{true}}\|_2$	$d_H(\text{sign}(Ax), y)$	$d_S(x, x_{\text{true}})^{-1}$
CC - ℓ_0	$2.34 \cdot 10^{-1}$	$2.15 \cdot 10^{-2}$	13.53
CC - ℓ_1	$2.37 \cdot 10^{-1}$	$5.43 \cdot 10^{-2}$	13.30
BIHT - k	$1.49 \cdot 10^{-1}$	0	21.62
BIHT - $4k$	$4.56 \cdot 10^{-1}$	0	6.85

Table 4.5 $m = 2000, n = 1000, k = 100$, sparse signal

	$\|x - x_{\text{true}}\|_2$	$d_H(\text{sign}(Ax), y)$	$d_S(x, x_{\text{true}})^{-1}$
CC - ℓ_0	$4.04 \cdot 10^{-2}$	$5.91 \cdot 10^{-3}$	88.34
CC - ℓ_1	$4.81 \cdot 10^{-2}$	$1.33 \cdot 10^{-2}$	72.05
BIHT - k	$1.54 \cdot 10^{-1}$	0	20.65
BIHT - $4k$	$4.54 \cdot 10^{-1}$	0	6.87

Table 4.6 $m = 2000, n = 1000, k = 100$, compressible signal

	$\|x - x_{\text{true}}\|_2$	$d_H(\text{sign}(Ax), y)$	$d_S(x, x_{\text{true}})^{-1}$
CC - ℓ_0	$7.68 \cdot 10^{-1}$	$5.5 \cdot 10^{-2}$	4.01
CC - ℓ_1	$7.34 \cdot 10^{-1}$	$1.3 \cdot 10^{-1}$	4.20
BIHT - k	$8.12 \cdot 10^{-1}$	0	3.77
BIHT - $4k$	$9.62 \cdot 10^{-1}$	0	3.13

Table 4.7 $m = 500, n = 1000, k = 100$, sparse signal

	$\|x - x_{\text{true}}\|_2$	$d_H(\text{sign}(Ax), y)$	$d_S(x, x_{\text{true}})^{-1}$
CC - ℓ_0	$8.57 \cdot 10^{-2}$	$8.5 \cdot 10^{-3}$	42.10
CC - ℓ_1	$9.78 \cdot 10^{-2}$	$1.55 \cdot 10^{-2}$	35.85
BIHT - k	$6.92 \cdot 10^{-1}$	0	4.46
BIHT - $4k$	$9.44 \cdot 10^{-1}$	0	3.20

Table 4.8 $m = 500, n = 1000, k = 100$, compressible signal

4.5 Discussion

In our developments, we have shown that this level set . Although the experimental results are promising, we have not entirely eliminated the possible negative side effects from solving the general, non-convex problem. Indeed, although the outer secant method will always converge reasonably (assuming sufficient smoothness on the value function), evaluating the value function itself may result in iterates being trapped in a local, but not global minimum, in particular if we warm-start the iterations with the solution from the previous problem. We have not observed this pathological case in our experiments, but the observant reader may readily construct such pathological examples based on the geometric considerations shown in Figure 4.1. For problems where c is linear, this approach performs well in practice. Clearly there is a theoretical gap between the gradient descent method analyzed in this chapter and the projected LBFGS method used to evaluate the value function. To the best of our knowledge, this is the first algorithm for convex composite minimization that has been experimentally validated on a wide variety of medium to large-scale problem instances.

4.6 Conclusion

In this chapter, we have proposed a new technique for solving composite-convex optimization problems. This class of problems is quite general and encompasses a number of important applications such as robust tensor principal component analysis and cosparsity-based compressed sensing. Instead of dealing with the non-smooth problem in its original form, which would result in slow convergence for large-scale problems, the original problem induces a corresponding value function, which we aim to evaluate. The value function is a function of the level set parameter, which aims to increase this scalar through a Newton or secant root finding scheme until the level set and the image of the inner, smooth mapping coincide. At this point, the original problem is solved. Evaluating the value function is much simpler than minimizing the original problem due to the smoothness of the objective function. We have proved that a simple gradient descent scheme converges linearly when evaluating these subproblems using analysis related to the Polyak-Lojasiewicz

inequality, although other more practically appealing methods such as LBFGS are available. Applying the level-set approach to a variety of signal reconstruction and classification problems has shown that this method is competitive for solving this class of problems. We are even able to handle non-convex problems without the tuning of additional hyperparameters. One limitation of this approach is that the subproblems have to be solved relatively accurately in order for the secant method to update the τ parameter. The authors in [17] study the performance of secant updates when inexact upper and lower bounds are available for $v(\tau)$ from partially solving the subproblem and duality, respectively. It is in this line of reasoning that we may be able to develop further insights in to incorporating inexactness in to this method in future work.

It is also important to note that the theoretical derivations for computing the value function derivatives when c is nonlinear are thus far incomplete. Although the straightforward formula appears to be true empirically, it is difficult to prove this fact for general nonlinear mappings. This is an important application for nonlinear models such as rank-k parametrizations of low rank matrices $X = LL^T$ for $L \in \mathbb{R}^{n\times k}$ and smooth parametrizations for higher order tensors such as the Hierarchical Tucker mapping. We note that, owing to the use of first-order methods to evaluate $v(\tau)$, these techniques are most appropriately applied to problems that are well-conditioned. The constants in the convergence bound in Proposition 4.4 degrade significantly as the conditioning of the problem worsens. Potentially accelerated first order methods such as Nesterov's method [190] could be used to ease the dependence on the square of the condition number of the mapping c (in the linear case), but we leave this as a topic for future research.

4.7 Acknowledgements

We would like to thank Aleksandr Aravkin for his helpful feedback on an early version of this work.

Chapter 5

A Unified 2D/3D Large Scale Software Environment for Nonlinear Inverse Problems

5.1 Introduction

Solving large scale inverse problems is a challenging endeavour for a number of reasons, not least of which is the sheer volume of prerequisite knowledge required. Developing numerical methods for inverse problems involves the intersection of a number of fields, in particular numerical linear algebra, nonlinear non-convex optimization, numerical partial differential equations, as well as the particular area of physics or biology the problem is modelled after, among others. As a result, many software packages aim for a completely general approach, implementing a large number of these components in various sub-modules and interfaced in a hierarchical way. There is often a danger with approaches that increase the cognitive load on the user, forcing them to keep the conceptual understanding of many components of the software in their minds at once. This high cognitive load can result in prolonging the initial setup time of a new researcher, delaying the time that they are actually productive while they attempt to comprehend how the code behaves. Moreover, adhering to a software design model that does not make intuitive sense can disincentivize modifications and improvements to the codebase. In an ideal world, a researcher with a general knowledge of the subject area should be able to sit in front of a well-designed software package and easily associate the underlying mathematics with the code they are presented. If a researcher is interested in prototyping high level algorithms, she is not necessarily interested in having to deal with the minutia of compiling a large number of software packages, manually managing memory, or writing low level code in C or Fortran in order to implement, for example, a simple stochastic optimization algorithm. Researchers are at their best when actually per-

forming research and software should be designed to facilitate that process as easily as possible.

Academic software environments for inverse problems are not necessarily geared towards high-performance, making use of explicit modeling matrices or direct solvers for 3D problems. Given the enormous computational demands of solving such problems, industrial codes focus on the performance-critical aspect of the problem and are often written in a low-level language without focusing on proper design. These software engineering decisions results in code that is hard to understand, maintain, and improve. Fortran veterans who have been immersed in the same software environment for many years are perfectly happy to squeeze as much performance out of their code as possible, but cannot easily integrate higher-level algorithms in to an existing codebase. As a result of this disparity, the translation of higher-level academic research ideas to high-performance industrial codes can be lost, which inhibits the uptake of new academic ideas in industry and vice-versa.

One of the primary examples in this work is the seismic inverse problem and variants thereof, which are notable in particular for their large computational requirements and industrial applications. Seismic inverse problems aim to reconstruct an image of the subsurface of the earth from multi-experiment measurements conducted on the surface. An array of pressure guns inject a pressure differential in to the water layer, which in turn generates a wave that travels to the ocean floor. These waves propagate in to the earth itself, reflect off of various discontinuities, before traveling back to the surface to be measured at an array of receivers. Our goal in this problem, as well as many other boundary-value problems, is to reconstruct the coefficients of the model (i.e., the wave equation in the time domain or the Helmholtz equation in the frequency domain) that describes this physical system such that the waves generated by our model agree with those in our measured data.

The difficulty in solving industrial-scale inverse problems arises from the various constraints imposed by solving a real-world problem. Acquired data can be noisy, lack full coverage, and, in the seismic case, can miss low and high frequencies [246] as a result of equipment and environmental constraints. Particularly in the seismic case, missing low frequencies results in a highly-oscillatory objective function with multiple local minima, requiring a practitioner to estimate an accurate starting model, while missing high frequencies results in a loss of detail [261]. Realistically sized problems involve the propagation of hundreds of wavelengths in geophysical [112] and earthquake settings [146], where wave phenomena require a minimum number of points per wavelength to model meaningfully [135]. These constraints can lead to large models and the resulting system matrices become too large to store explicitly, let alone invert with direct methods.

Our goal in this work is to outline a software design approach to solving partial differential equation (PDE) constrained optimization problems that allows users to operate with the high-level components of the problem such as objective function evaluations, gradients, and Hessians, irrespective of the underlying PDE or dimensionality. With this approach, a practitioner can design and prototype inversion

algorithms on a complex 2D problem and, with minimal code changes, apply these same algorithms to a large scale 3D problem. The key approach in this instance is to structure the code in a hierarchical and modular fashion, whereby each module is responsible for its own tasks and the entire system structures the dependencies between modules in a tiered fashion. In this way, the entire codebase becomes much easier to test, optimize, and understand. Moreover, a researcher who is primarily concerned with the high level 'building blocks' of an inversion framework can simply work with these units in a standalone fashion and rely on default configurations for the lower level components. By using a proper amount of information hiding through abstraction, users of this code can delve as deeply in to the code architecture as they are interested in. We also aim to make our 'code look like the math' as much as possible, which will help our own development as well as that of future researchers, and reduce the cognitive load required for a researcher to start performing research.

There are a number of existing software frameworks for solving inverse problems with varying goals in mind. The work of [241] provides a C++ framework built upon the abstract Rice Vector Library [197] for time domain modeling and inversion, which respects the underlying Hilbert spaces where each vector lives by automatically keeping track of units and grid spacings, among other things. The low-level nature of the language it is built in exposes too many low level constructs at various levels of its hierarchy, making integrating changes in to the framework cumbersome. The Seiscope toolbox [182] implements high level optimization algorithms in the low-level Fortran language, with the intent to interface in to existing modeling and derivative codes using reverse communication. As we highlight below, this is not necessarily a beneficial strategy and merely obfuscates the codebase, as low level languages should be the domain of computationally-intensive code rather than high-level algorithms. The Trilinos project [124] is a large collection of packages written in C++ by a number of domain-specific experts, but requires an involved installation procedure, has no straightforward entrance point for PDE-constrained optimization, and is not suitable for easy prototyping. Implementing a modelling framework in PETSc [20], such as in [156], let alone an inversion framework, exposes too many of the unnecessary details at each level of the hierarchy given that PETSc is written in C. The Devito framework [166] offers a high-level symbolic Python interface to generate highly optimized stencil-based C- code for time domain modelling problems, with extensions to inversion. This is promising work that effectively delineates the high-level mathematics from the low-level computations. We follow a similar philosophy in this work. This work builds upon ideas in [256], which was a first attempt to implement a high-level inversion framework in Matlab.

Software frameworks in the finite-element regime have been successfully applied to optimal control and other PDE-constrained optimization problems. The Dolfin framework [98] employs a high-level description of the PDE-constrained problem written in the UFL language for specifying finite elements, which is subsequently compiled in to lower level finite element codes with the relevant adjoint equations derived and solved automatically for the objective and gradient. For the geophysical examples, the finite element method does not easily lend itself to applying

a perfectly-matched layer to the problem compared to finite differences, although some progress has been made in this front, i.e., see [73]. In general, finite difference methods are significantly easier to implement, especially in a matrix-free manner, than finite element methods, although the latter have a much stronger convergence theory. Moreover, for systems with oscillatory solutions such as the Helmholtz equation, applying the standard 7-point stencil to the problem is inadvisable due to the large amount of numerical dispersion introduced, resulting in a system matrix with a large number of unknowns. This fact, along with the indefiniteness of the underlying matrix, makes it very challenging to solve with standard Krylov methods. More involved approaches are needed to adequately discretize such equations, see, e.g., [251, 191, 69]. The SIMPEG package [76] is designed in a similar spirit to the considerations in this work, but does not fully abstract away unnecessary components from the user and is not designed with large-scale computations in mind as it lacks inherent parallelism. The Jinv package [220] is written in Julia in a similar spirit to this work with an emphasis on finite element discretizations using the parallel MUMPS solver [8] for computing the fields and is parallelized over the number of source experiments.

When considering the performance-understandability spectrum for designing inverse problem software, it is useful to consider Amdahl's law [199]. Roughly speaking, Amdahl's law states that in speeding up a region of code, through parallelization or other optimizations, the speedup of the overall program will always be limited by the remainder of the program that does not benefit from the speedup. For instance, speeding up a region where the program spends 50% of its time by a factor of 10 will only speed up the overall program by a maximum factor of 1.8. For any PDE-based inverse problem, the majority of the computational time is spent solving the PDEs themselves. Given Amdahl's law and a limited budget of researcher time, this would imply that there is virtually no performance benefit in writing both the 'heavy lifting' portions of the code as well as the auxiliary operations in a low level language, which can impair readability and obscure the role of the individual component operations in the larger framework. Rather, one should aim to use the strengths of a high level language to express mathematical ideas cleanly in code and exploit the efficiency of a low level language, at the proper instance, to speed up primitive operations such as a multi-threaded matrix-vector product. These considerations are also necessary to manage the complexity of the code and ensure that it functions properly. Researcher time, along with computational time, is valuable and we should aim to preserve productivity by designing these systems with these goals in mind.

It is for this reason that we choose to use Matlab to implement our parallel inversion framework as it offers the best balance between access to performance-critical languages such as C and Fortran, while allowing for sufficient abstractions to keep our code concise and loyal to the underlying mathematics. A pure Fortran implementation, for instance, would be significantly more difficult to develop and understand from an outsider's perspective and would not offer enough flexibility for our purposes. Python would also potentially be an option for implementing this

framework. At the time of the inception of this work, we found that the relatively new scientific computing language Julia [29] was in too undeveloped of a state to facilitate all of the abstractions we needed; this may no longer be the case as of this writing.

5.1.1 Our Contributions

Using a hierarchical approach to our software design, we implement a framework for solving inverse problems that is flexible, comprehensible, efficient, scalable, and consistent. The flexibility arises from our design decisions, which allow a researcher to swap components (parallelization schemes, linear solvers, preconditioners, discretization schemes, etc.) in and out to suit her needs and the needs of her local computational environment. Our design balances efficiency and understandability through the use of object oriented programming, abstracting away the low-level mechanisms of the computationally intensive components through the use of the SPOT framework [93]. The SPOT methodology allows us to abstract away function calls as matrix-vector multiplications in Matlab, the so-called matrix-free approach. By abstracting away the lower level details, our code is clean and resembles the underlying mathematics. This abstraction also allows us to swap between using explicit, sparse matrix algebra for 2D problems and efficient, multi-threaded matrix-vector multiplications for 3D problems. The overall codebase is then agnostic to the dimensionality of m, which encourages code reuse when applying new algorithms to large scale problems. Our hierarchical design also decouples parallel data distribution from computation, allowing us to run the same algorithm as easily on a small 2D problem using a laptop as on a large 3D problem using a cluster. We also include unit tests that demonstrate that our code accurately reflects the underlying mathematics in Section (5.5.1). We call this package WAVEFORM (soft-WAre enVironmEnt For nOnlinear inveRse probleMs), which can be obtained at https://github.com/slimgroup/WAVEFORM.

In this work, we also propose a new multigrid-based preconditioner for the 3D Helmholtz equation that only requires matrix-vector products with the system matrix at various levels of discretization, i.e., is matrix-free at each multigrid level, and employs standard Krylov solvers as smoothers. This preconditioner allows us to operate on realistically sized 3D seismic problems without exorbitant memory costs. Our numerical experiments demonstrate the ease of which we can apply high level algorithms to solving the PDE-based parameter estimation problem and its variants while still having full flexibility to swap modeling code components in and out as we choose.

5.2 Preamble: Theory and Notation

To ensure that this chapter is sufficiently self-contained, we outline the basic structure and derivations of our inverse problem of interest. Lowercase, letters such as

x, y, z denote vectors and uppercase letters such as A, B, C denote matrices or linear operators of appropriate size. To distinguish between continuous objects and their discretized counterparts, with a slight abuse of notation, we will make the spatial coordinates explicit for the continuous objects, i.e., sampling $u(x, y, z)$ on a uniform spatial grid results in the vector u. Vectors u can depend on parameter vectors m, which we indicate with $u(m)$. The adjoint operator of a linear mapping $x \mapsto Ax$ is denoted as A^* and the conjugate Hermitian transpose of a complex matrix B is denoted B^H. If B is a real-valued matrix, this is the standard matrix transpose.

Our model inverse problem is the multi-source parameter estimation problem. Given our data $d_{i,j}$ depending on the i^{th} source and j^{th} frequency, our measurement operator P_r, and the linear partial differential equation $H(m)u(m) = q$ depending on the model parameter m, find the model m that minimizes the misfit between the predicted and observed data, i.e.,

$$\min_{m, u_{i,j}} \sum_{i}^{N_s} \sum_{j}^{N_f} \phi(P_r u_{i,j}, d_{i,j})$$
$$\text{subject to } H_j(m) u_{i,j} = q_{i,j}.$$

Here $\phi(s, t)$ is a smooth misfit function between the inputs s and t, often the least-squares objective $\phi(s, t) = \frac{1}{2}\|s - t\|_2^2$, although other more robust penalties are possible, see e.g., [11, 10]. The indices i and j vary over the number of sources N_s and number of frequencies N_f, respectively. For the purposes of notational simplicity, we will drop this dependence when the context permits. Note that our design is specific to boundary-value problems rather than time-domain problems, which have different computational and storage challenges.

A well known instance of this setup is the full waveform inversion problem in exploration seismology, which involves discretizing the constant-density Helmholtz equation

$$\begin{aligned}(\nabla^2 + m(x, y, z))u(x, y, z) &= S(\omega)\delta(x - x_s)\delta(y - y_s)\delta(z - z_s) \\ \lim_{r \to \infty} r \left(\frac{\partial}{\partial r} - i\sqrt{m} \right) u(x, y, z) &= 0\end{aligned} \tag{5.1}$$

where $\nabla^2 = \partial_x^2 + \partial_y^2 + \partial_z^2$ is the Laplacian, $m(x, y, z) = \frac{\omega^2}{v^2(x,y,z)}$ is the wavenumber, ω is the angular frequency and $v(x, y, z)$ is the gridded velocity, $S(\omega)$ is the per-frequency source weight, (x_s, y_s, z_s) are the spatial coordinates of the source, and the second line denotes the Sommerfeld radiation condition [235] with $r = \sqrt{x^2 + y^2 + z^2}$. In this case, $H(m)$ is any finite difference, finite element, or finite volume discretization of (5.1). Other examples of such problems include electrical impedance tomography using a simplified form of Maxwell's equations, which can be reduced to Poisson's equation [71, 34, 5], X-ray tomography, which can be thought of as the inverse problem corresponding to a transport equation, and synthetic aperture radar, using the wave equation [188]. For realistically sized industrial

problems of this nature, in particular for the seismic case, the size of the model vector m is often $\mathcal{O}(10^9)$ and there can be $\mathcal{O}(10^5)$ sources, which prevents the full storage of the fields. Full-space methods, which use a Newton iteration on the associated Lagrangian system, such as in [30, 208], are infeasible as a large number of fields have to be stored and updated in memory. As a result, these large scale inverse problems are typically solved by eliminating the constraint and reformulating the problem in an unconstrained or reduced form as

$$\min_m f(m) := \sum_i^{N_s} \sum_j^{N_f} \phi(P_r H_j(m)^{-1} q_{i,j}, d_{i,j}). \tag{5.2}$$

We assume that we our continuous PDEs are posed on a rectangular domain Ω with zero Dirichlet boundary conditions. For PDE problems that require sponge or perfectly matched layers, we extend Ω to $\Omega' \supset \Omega$ and vectors defined on Ω are extended to Ω' by extension in the normal direction. In this extended domain for the acoustic Helmholtz equation, for instance, we solve

$$\begin{aligned}(\partial_x^2 + \partial_y^2 + \partial_z^2 + \omega^2 m(x,y,z))u(x,y,z) = \delta(x-s_x)\delta(y-s_x)\delta(z-s_z) \quad (x,y,z) \in \Omega \\ (\tilde{\partial_x}^2 + \tilde{\partial_y}^2 + \tilde{\partial_z}^2 + \omega^2 m(x,y,z)) = 0 \quad (x,y,z) \in \Omega' \setminus \Omega,\end{aligned}$$

where $\tilde{\partial_x} = \dfrac{1}{\eta(x)}\partial_x$, for appropriately chosen univariate PML damping functions $\eta(x)$, and similarly for y, z. This results in solutions u that decay exponentially for $x \in \Omega' \setminus \Omega$. We refer the reader to [28, 72, 122] for more details.

We assume that the source functions $q_{i,j}(x)$ are localized in space around the points $\{x_i^s\}_{i=1}^{n_s}$, which make up our source grid. The measurement or sampling operator P_r samples function values defined on Ω' at the set of receiver locations $\{x_k^r\}_{k=1}^{n_r}$. In the most general case, the points x_k^r can vary per-source (i.e., as the location of the measurement device is dependent on the source device), but we will not consider this case here. In either case, the source grid can be independent from the receiver grid.

While the PDE itself is linear, the mapping that predicts data $F(m) := m \mapsto P_r H(m)^{-1} q$, the so-called the forward modeling operator, is known to be highly nonlinear and oscillatory in the case of the high frequency Helmholtz equation [240], which corresponds to propagating anywhere between 50-1000 wavelengths for realistic models of interest. Without loss of generality, we will consider the Helmholtz equation as our prototypical model in the sequel. The level of formalism, however, will ultimately be related to parameter estimation problems that make use of real-world data, which is inherently band-limited. We will therefore not be overly concerned with convergence as the mesh- or element-size tends to zero, as the informative capability of our acquired data is only valid up until a certain resolution dictated by the resolution of our measurement device. We will focus solely on the discretized formulation of the problem from hereon out.

We can compute relevant derivatives of the objective function with straightforward, ableit cumbersome, matrix calculus. Consider the state equation $u(m) = H(m)^{-1}q$. Using the chain rule, we can derive straightforward expressions for the directional derivative $Du(m)[\delta m]$ as

$$Du(m)[\delta m] = -H(m)^{-1}DH(m)[\delta m]u(m). \tag{5.3}$$

Here $DH(m)[\delta m]$ is the directional derivative of the mapping $m \mapsto H(m)$, which is assumed to be smooth. To emphasize the dependence on the linear argument, we let T denote the linear operator defined by $T\delta m = DH(m)[\delta m]u(m)$, which outputs a vector in model space. Note that $T = T(m, u(m))$, but we drop this dependence for notational simplicity. We let T^* denote the adjoint of the linear mapping $\delta m \mapsto T\delta m$. The associated adjoint mapping of (5.3) is therefore

$$Du(m)[\cdot]^* y = -T^* H(m)^{-H} y.$$

The (forward) action of the Jacobian J of the forward modelling operator $F(m) = P_r u(m)$ is therefore given by $J\delta m = P_r Du(m)[\delta m]$.

We also derive explicit expressions for the Jacobian adjoint, Gauss-Newton Hessian, and full Hessian matrix-vector products, as outlined in Table 5.1, although we leave the details for Appendix (A). For even medium sized 2D problems, it is computationally infeasible to store these matrices explicitly and therefore we only have access to matrix-vector products. The number of matrix-vector products for each quantity are outlined in Table 5.2 and are per-source and per-frequency. By adhering to the principle of 'the code should reflect the math', once we have the relevant formula from Table 5.1, the resulting implementation will be as simple as copying and pasting these formula in to our code in a straightforward fashion, which results in little to no computational overhead, as we shall see in the next section.

$u(m)$	$H(m)^{-1}q$
$Du(m)[\delta m]$	$-H(m)^{-1}T\delta m$
$Du(m)[\cdot]^* y$	$-T^* H(m)^{-H} y$
$F(m)$	$Pu(m)$
$J\delta m := DF(m)[\delta m]$	$PDu(m)[\delta m]$
$T\delta m$	$DH(m)[\delta m]u(m)$, e.g., (A.2)
$T^* z$	PDE-dependent, e.g., (A.3)
$DT^*[\delta m, \delta u]z$	PDE-dependent, e.g., (A.4)
$V(m)$	$-H(m)^{-H}P^T\nabla\phi$
$DV(m)[\delta m]$	$H(m)^{-H}(-DH(m)[\delta m]V(m) - P^T\nabla^2\phi(Pu)[PDu(m)[\delta m]])$
$H_{GN}\delta m := J^H J\delta m$	$T^* H(m)^{-H}P^T PH(m)^{-1}T\delta m$
$\nabla f(m)$	$T^* V(m)$
$\nabla^2 f(m)[\delta m]$	$DT^*[\delta m, Du(m)[\delta m]]V(m) + T^* DV(m)[\delta m]$

Table 5.1 Quantities of interest for PDE-constrained optimization.

Quantity	# PDEs
$f(m)$	1
$f(m), \nabla f(m)$	2
$H_{GN}\delta m$	3
$\nabla^2 f(m)[\delta m]$	4

Table 5.2 Number of PDEs (per frequency/source) for each optimization quantity of interest

5.3 From Inverse Problems to Software Design

Given the large number of summands in (5.2) and the high cost of solving each PDE $u_{i,j} = H_j(m)^{-1}q_{i,j}$, there are a number of techniques one can employ to reduce up per-iteration costs and to increase convergence speed to a local minimum, given a fixed computational budget. Stochastic gradient techniques [37, 36] aim to reduce the per iteration costs of optimization algorithms by treating the sum as an expectation and approximate the average by a random subset. The batching strategy aims to use small subsets of data in earlier iterations, making significant progress towards the solution for lower cost. Only in later iterations does the algorithm require a larger number of per-iteration sources to ensure convergence [103]. In a distributed parallel environment, the code should also employ batch sizes that are commensurate with the available parallel resources. One might also seek to solve the PDEs to a lower tolerance at the initial stages, as in [257], and increasing the tolerance as iterations progress, an analogous notion to batching. By exploiting curvature information by minimizing a quadratic model of $f(m)$ using the Gauss-Newton method, one can further enhance convergence speed of the outer algorithm.

In order to employ these high-level algorithmic techniques and facilitate code reuse, our optimization method should be a black-box, in the sense that it is completely oblivious to the underlying structure of the inverse problem, calling a user-defined function that returns an objective value, gradient, and Gauss-Newton Hessian operator. Our framework should be flexible enough so that, for 2D problems, we can afford to store the sparse matrix $H(m)$ and utilize the resulting efficient sparse linear algebra tools for inverting this matrix, while for large-scale 3D problems, we can only compute matrix-vector products with $H(m)$ with coefficients constructed on-the-fly. Likewise, inverting $H(m)$ should only employ Krylov methods that use these primitives, such as FGMRES [221]. These restrictions are very realistic for the seismic inverse problem case, given the large number of model and data points involved as well as the limited available memory for each node in a distributed computational environment. In the event that a new robust preconditioner developed for the PDE system matrix, we should be able to easily swap out one algorithm for another, without touching the outer optimization method. Likewise, if researchers develop a much more efficient stencil for discretizing the PDE, develop a new misfit objective [11], or add model-side constraints [200], we would like to easily integrate

such changes in to the framework with minimal code changes. Our code should also expose an interface to allow a user or algorithm to perform source/frequency subsampling from arbitrarily chosen indices.

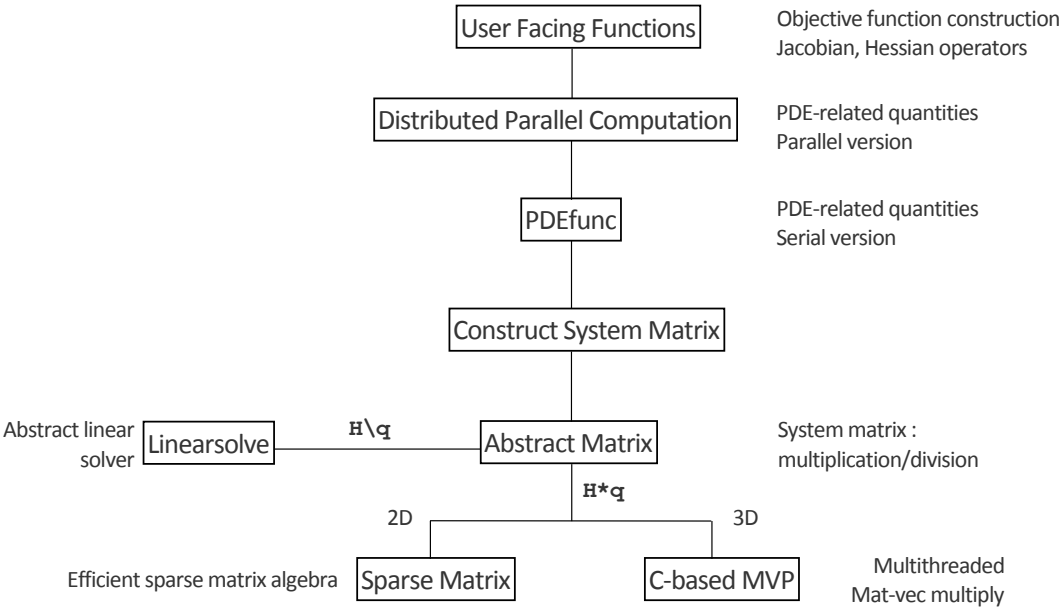


Figure 5.1 Software Hierarchy.

We decouple the various components of the inverse problem context by using an appropriate software hierarchy, which manages complexity level-by-level and will allow us to test components individually to ensure their correctness and efficiency, as shown in Figure 5.1. Each level of the hierarchy is responsible for a specific set of procedural requirements and defers the details of lower level computations to lower levels in the hierarchy.

At the topmost level, our user-facing functions are responsible for constructing a misfit function suitable for black-box optimization routines such as LBFGS or Newton-type methods [171, 272, 113]. This procedure consists of

- handling subsampling of sources/frequencies for the distributed data volume
- coarsening the model, if required (for low frequency 3D problems)
- constructing the function interface that returns the objective, gradient, and requested Hessian at the current point.

For the Helmholtz case in particular, coarsening the model allows us to keep the number of degrees of freedom to a minimum, in line with the requirements of the discretization. In order to accommodate stochastic optimization algorithsm, we provide an option for a batch mode interface, which allows the objective function to take in as input both a model vector and a set of source-frequency indices. This option allows either the user or the outer stochastic algorithm to dynamically specify

which sources and frequencies should be computed at a given time. This auxiliary information is passed along to lower layers of the hierarchy. Additionally, we provide methods to construct the Jacobian, Gauss-Newton, and Full Hessian as SPOT operators. For instance, when writing Hess$*v$ as in a linear solver, Matlab implicitly calls the lower level functions that actually solve the PDEs to compute this product, all the while never forming the matrix explicitly. Information hiding in this fashion allows us to use existing linear solver codes written in Matlab (Conjugate Gradient, MINRES, etc.) to solve the resulting systems. For the parameter inversion problem, the user can choose to have either the Gauss-Newton or full Hessian operators returned at the current point.

Lower down the hierarchy, we have our `PDEfunc` layer. This function is responsible for computing the quantities of interest, i.e., the objective value, gradient, Hessian or Gauss-Newton Hessian matrix-vector product, forward modelling operator, or linearized forward modelling operator and its adjoint. At this stage in the hierarchy, we are concerned with 'assembling' the relevant quantities based on PDE solutions in to their proper configurations, rather than how exactly to obtain such solutions, i.e., we implement the formulas in Table 5.1. Here the PDE system matrix is a SPOT operator that has methods for performing matrix-vector products and matrix-vector divisions, which contains information about the particular stencil to use as well as which linear solvers and preconditioners to call. When dealing with 2D problems, this SPOT operator is merely a shallow wrapper around a sparse matrix object and contains its sparse factorization, which helps speed up solving PDEs with multiple right hand sides. In order to discretize the delta source function, we use Kaiser-windowed sinc interpolation [133]. At this level in the hierarchy, our code is not sensitive to the discretization or even the particular PDE we are solving, as we demonstrate in Section (5.5.4) with a finite volume discretization of the Poisson equation. To illustrate how closely our code resembles the underlying mathematics, we include a short snippet of code from our `PDEfunc` below in Figure 5.1.

In our parallel distribution layer, we compute the result of `PDEfunc` in an embarrassingly parallel manner by distributing over sources and frequencies and summing the results computed across various Matlab workers. This distribution scheme uses Matlab's Parallel Toolbox, which is capable of Single Program Multiple Data (SPMD) computations that allow us to call `PDEfunc` identically on different subsets of source/frequency indices. The data is distributed as in Figure 5.2. The results of the local worker computations are then summed together (for the objective, gradient, adjoint-forward modelling, Hessian-vector products) or assembled in to a distributed vector (forward modelling, linearized forward modelling). Despite the ease of use in performing these parallel computations, the parallelism of Matlab is **not** fault-tolerant, in that if a single worker or process crashes at any point in computing a local value with `PDEfunc`, the parent calling function aborts as well. In a large-scale computing environment, this sort of behaviour is unreliable and therefore we recommend swapping out this 'always-on' approach with a more resilient model such as a map reduce paradigm. One possible implementation workaround is to measure the elapsed time of each worker, assuming that the work is distributed evenly. In

```
% Set up interpolation operators
% Source grid -> Computational Grid
Ps = opInterp('sinc',model.xsrc,xt,model.ysrc,yt,model.zsrc,zt);
% Computational grid -> Receiver grid
Pr = opInterp('sinc',model.xrec,xt,model.yrec,yt,model.zrec,zt)';
% Sum along sources dimension
sum_srcs = @(x) to_phys*sum(real(x),2);
% Get Helmholtz operator, computational grid struct, its derivative
[Hk,comp_grid,T,DT_adj] = discrete_pde_system(v,model,freq(k),params);
U = H \ Q;
switch func
 case OBJ
    [phi,dphi] = misfit(Pr*U,getData(Dobs,data_idx),current_src_idx,freq_idx);
    f = f + phi;
    if nargout >= 2
        V = H' \ ( -Pr'* dphi);
        g = g + sum_srcs(T(U)'*V);
    end

 case FORW_MODEL
    output(:,data_idx) = Pr*U;

 case JACOB_FORW
    dm = to_comp*vec(input);
    dU = H\(-T(U)*dm);
    output(:,data_idx) = Pr*dU;

 case JACOB_ADJ
    V = H'\( -Pr'* input(:,data_idx) );
    output = output + sum_srcs(T(U)'*V);

 case HESS_GN
    dm = to_comp*vec(input);
    dU = H\(-T(U)*dm);
    dU = H'\(-Pr'*Pr*dU);
    output = output + sum_srcs(T(U)*dU);

 case HESS
    dm = to_comp*vec(input);
    [~,dphi,d2phi] = misfit(Pr*U,getData(Dobs,data_idx),current_src_idx,freq_idx);
    dU = H\(-T(U)*dm);
    V = H'\(-Pr'*dphi);
    dV = H'\(-T(V)*dm - Pr'* reshape(d2phi*vec(Pr*dU),nrec,size(U,2)));
    output = output + sum_srcs(DT_adj(U,dm,dU)*V + T(U)*dV);
```

Listing 5.1 Excerpt from the code of `PDEfunc`.

the event that a worker times out, one can simply omit the results of that worker and sum together the remaining function and gradient values. In some sense, one can account for random machine failure or disconnection by considering it another source of stochasticity in an outer-level stochastic optimization algorithm.

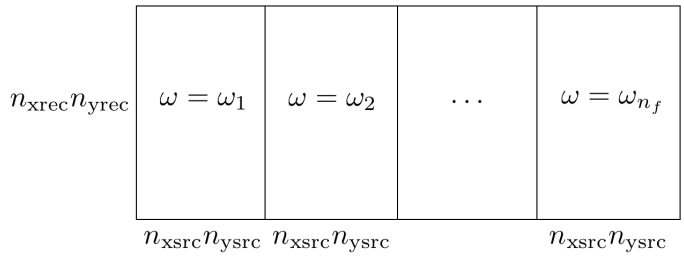


Figure 5.2 Data distributed over the joint (source, frequency) indices.

Further down the hierarchy, we construct the system matrix for the PDE in the following manner. As input, we require the current parameters m, information about the geometry of the problem (current frequency, grid points and spacing, options for number of PML points), as well as performance and linear solver options (number of threads, which solver/preconditioner to use, etc.). This function also extends the medium parameters to the PML extended domain, if required. The resulting outputs are a SPOT operator of the Helmholtz matrix, which has suitable routines for performing matrix-vector multiplications and divisions, a struct detailing the geometry of the pml-extended problem, and the mappings T and $DT^*[\delta m]$ from Table 5.1. It is in this function that we also construct the user-specified preconditioner for inverting the Helmholtz (or other system) matrix.

The actual operator that is returned by this method is a SPOT operator that performs matrix-vector products and matrix-vector divisions with the underlying matrix, which may take a variety of forms. In the 2D regime, we can afford to explicitly construct the sparse matrix and we utilize the efficient sparse linear algebra routines in Matlab for solving the resulting linear system. In this case, we implement the 9-point optimal discretization of [70], which fixes the problems associated to the 9-point discretization of [142], and use the sparse LU decomposition built in to Matlab for inverting the system. These factors are computed at the initialization of the system matrix and are reused across multiple sources. In the 3D regime, we implement a stencil-based matrix-vector product (i.e., one with coefficients constructed on the fly) written in C++, using the compact 27-point stencil of [191] along with its adjoint and derivative. The stencil-based approach allows us to avoid having to keep, in this case, 27 additional vectors of the size of the PML-extended model in memory. This implementation is multi-threaded along the $z-$axis using OpenMP and is the only 'low-level' component in this software framework, geared towards high performance. Since this primitive operation is used throughout the inverse problem framework, in particular for the iterative matrix-vector division, any performance improvements made to this code will propagate throughout the

entire codebase. Likewise, if a 'better' discretization of the PDE becomes available, it can be easily integrated in to the software framework by swapping out the existing function at this stage, without modifying the rest of the codebase. When it is constructed, this SPOT operator also contains all of the auxiliary information necessary for performing matrix-vector products (either the matrix itself, for 2D, or the PML-extended model vector and geometry information, for 3D) as well as for matrix-vector divisions (options for the linear solver, preconditioner function handle, etc.).

We allow the user access to multiplication, division, and other matrix operations like Jacobi and Kaczmarz sweeps through a unified interface, irrespective of whether the underlying matrix is represented explicitly or implicitly via function calls. For the explicit matrix case, we have implemented such basic operations in Matlab. When the matrix is represented implicitly, these operations are presented by a standard function handle or a 'FuncObj' object, the latter of which mirrors the Functor paradigm in C++. In the Matlab case, the 'FuncObj' stores a reference to a function handle, as well as a list of arguments. These arguments can be partially specialized upon construction, whereby only some of the arguments are specified at first. Later on when they are invoked, the remainder of their arguments are passed along to the function. In a sense, they implement the same functionality of anonymous functions in Matlab without the variable references to the surrounding workspace, which can be costly when passing around vectors of size N^3 and without the variable scoping issues inherent in Matlab. We use this construction to specify the various functions that implement the specific operations for the specific stencils. Through this delegation pattern, we present a unified interface to the PDE system matrix irrespective of the underlying stencil or even PDE in question.

When writing `u = H\q` for this abstract matrix object H, Matlab calls the 'linsolve' function, which delegates the task of solving the linear system to a user-specified solver. This function sets up all of the necessary preamble for solving the linear system with a particular method and preconditioner. Certain methods such as the row-based CGMN method, introduced in [] and applied to seismic problems in [257], require initial setup, which is performed here. This construction allows us to easily reuse the idea of 'solving a linear system with a specific method' in setting up the multi-level preconditioner of the next section, which reduces the overall code complexity since the multigrid smoothers themselves can be described in this way.

In order to manage the myriad of options available for computing with these functions, we distinguish between two classes of options and use two separate objects to manage them. 'LinSolveOpts' is responsible for containing information about solving a given linear system, i.e., which solver to use, how many inner/outer iterations to perform, relative residual tolerance, and preconditioner. 'PDEopts' contains all of the other information that is pertinent to `PDEfunc` (e.g., how many fields to compute at a time, type of interpolation operators to use for sources/receivers, etc.) as well as propagating the options available for the PDE discretization (e.g., stencil to use, number of PML points to extend the model, etc.).

5.3.1 Extensions

This framework is flexible enough for us to integrate additional models beyond the standard adjoint-state problem. We outline two such extensions below.

Penalty Method

Given the highly oscillatory nature of the forward modelling operator, due to the presence of the inverse Helmholtz matrix $H(m)^{-1}q$, one can also consider relaxing the exact constraint $H(m)u(m) = q$ in to an unconstrained and penalized form of the problem. This is the so-called Waveform Reconstruction Inversion approach [259], which results in the following problem, for a least-squares data-misfit penalty,

$$\min_m = \frac{1}{2}\|Pu(m) - d\|_2^2 + \frac{\lambda^2}{2}\|H(m)u(m) - q\|_2^2 \tag{5.4}$$

where $u(m)$ solves the least-squares system

$$u(m) = \operatorname*{argmin}_u \frac{1}{2}\|Pu - d\|_2^2 + \frac{\lambda^2}{2}\|H(m)u - q\|_2^2. \tag{5.5}$$

The notion of variable projection [14] underlines this method, whereby the field u is projected out of the problem by solving (5.5) for each fixed m. We perform the same derivations for problem (5.4) in Appendix A. Given the close relationship between the two methods, the penalty method formulation is integrated into the same function as our FWI code, with a simple flag to change between the two computational modes.

2.5D

For a 3D velocity model that is invariant with respect to one dimension, i.e., $v(x, y, z) = h(x, z)$ for all y, so $m(x, y, z) = \omega^2 g(x, z)$ for $g(x, z) = \dfrac{1}{h^2(x, z)}$, we can take a Fourier transform in the y-coordinate of (5.1) to obtain

$$(\partial_x^2 + \partial_z^2 + \omega^2 g(x, z) - k_y^2)\hat{u}(x, k_y, z) = S(\omega)\delta(x - x_s)\delta(z - z_s)e^{-ik_y y_s}.$$

This so-called 2.5D modeling/inversion allows us to mimic the physical behaviour of 3D wavefield propagation (e.g., $1/r$ amplitude decay vs $1/r^{1/2}$ decay in 2D, point sources instead of line sources, etc.) without the full computational burden of solving the 3D Helmholtz equation [236]. We solve a series of 2D problems instead, as follows.

Multiplying both sides by $e^{ik_y y_s}$ and setting $\tilde{u}_{k_y}(x, z) = e^{ik_y y_s}\hat{u}(x, k_y, z)$, we have that, for each fixed k_y, $\tilde{u}_{k_y}$ is the solution of $H(k_y)\tilde{u}_{k_y} = S(\omega)\delta(x - x_s)\delta(z - z_s)$ where $H(k_y) = (\partial_x^2 + \partial_z^2 + \omega^2 g(x, z) - k_y^2)$.

We can recover $u(x, y, z)$ by writing

$$\begin{aligned} u(x,y,z) &= \frac{1}{2\pi}\int_{-\infty}^{\infty} \hat{u}_{k_y}(x,z)e^{ik_y y}dk_y \\ &= \frac{1}{2\pi}\int_{-\infty}^{\infty} \tilde{u}_{k_y}(x,z)e^{ik_y(y-y_s)}dk_y \\ &= \frac{1}{\pi}\int_{0}^{\infty} \tilde{u}_{k_y}(x,z)\cos(k_y(y-y_s))dk_y \\ &= \frac{1}{\pi}\int_{0}^{k_{nyq}} \tilde{u}_{k_y}(x,z)\cos(k_y(y-y_s))dk_y. \end{aligned}$$

Here the third line follows from the symmetry between k_y and $-k_y$ and the fourth line restricts the integral to the Nyquist frequency given by the sampling in y, namely $k_{nyq} = \frac{\pi}{\Delta y}$ [236]. One can further restrict the range of frequencies to $[0, p \cdot k_c]$ where $k_c = \frac{\omega}{\min_{x,z} v(x,z)}$ is the so-called critical frequency and $p \geq 1, p \approx 1$. In this case, waves corresponding to a frequency much higher than that of k_c do not contribute significantly to the solution. By evaluating this integral with, say, a Gauss-Legendre quadrature, the resulting wavefield can be expressed as

$$u(x,y,z) = \sum_{i=1}^{N} w_i \tilde{u}_{k_y^i}(x,z),$$

which is a sum of 2D wavefields. This translates in to operations such as computing the Jacobian or Hessian having the same sum structure and allows us to incorporate 2.5D modeling and inversion easily in to the resulting software framework.

5.4 Multi-level Recursive Preconditioner for the Helmholtz Equation

Owing to the PML layer and indefiniteness of the underlying PDE system for sufficiently high frequencies, the Helmholtz system matrix is complex-valued, non-Hermitian, indefinite, and therefore challenging to solve using Krylov methods without adequate preconditioning. There have been a variety of ideas proposed to precondition the Helmholtz system, including multigrid methods [239], methods inverting the shifted Laplacian system [215, 205, 96], sweeping preconditioners [95, 172, 207, 173], domain decomposition methods [237, 238, 38, 104], and Kaczmarz sweeps [258], among others. These methods have varying degrees of applicability in this framework. Some methods such as the sweeping preconditioners rely on having explicit representations of the Helmholtz matrix and keeping track of dense LU factors on multiple subdomains. Their memory requirements are quite steep as a result and they require a significant amount of bookkeeping to program correctly, in addition to their large setup time, which is prohibitive in inversion algorithms where the velocity is being updated. Many of these existing methods also make

stringent demands of the number of points per wavelength N_{ppw} needed to succeed, in the realm of 10 to 15, which results in prohibitively large system matrices for the purposes of inversion. As the standard 7-point discretization requires a high N_{ppw} in order to adequately resolve the solutions of the phases [191], this results in a large computational burden as the system size increases.

Our aim in this section is to develop a preconditioner that is scalable, in that it uses the multi-threading resources on a given computational node, easy to program, matrix-free, without requiring access to the entries of the system matrix, and leads to a reasonably low number of outer Krylov iterations. It is more important to have a larger number of sources distributed to multiple nodes rather than using multiple nodes solving a single problem when working in a computational environment where there is a nonzero probability of node failure.

We follow the development of the preconditioners in [51, 165], which uses a multigrid approach to preconditioning the Helmholtz equation. The preconditioner in [51] approximates a solution to (5.2) with a two-level preconditioner. Specifically, we start with a standard multigrid V-cycle [41] as described in Figure 5.1. The smoothing operator aims to reduce the amplitude of the low frequency components of the solution error, while the restriction and prolongation operators. For an extensive overview of the multigrid method, we refer the reader to [41]. We use linear interpolation as the prolongation operator and its adjoint as for restriction, although other choices are possible [86].

Algorithm 5.1 Standard multigrid V-cycle

V-cycle to solve $H_f x_f = b_f$ on the fine scale

Input: Current estimate of the fine-scale solution x_f

Smooth the current solution using a particular smoother (Jacobi, CG, etc.) to produce $\tilde{x}_f$

Compute the residual $r_f = b_f - A_f \tilde{x}_f$

Restrict the residual, right hand side to the coarse level $r_c = R r_f$, $b_c = R b_f$

Approximately solve $H_c x_c = r_c$

Interpolate the coarse-scale solution to the fine-grid, add it back to $\tilde{x}_f$, $\tilde{x}_f \leftarrow x_f + P x_c$

Smooth the current solution using a particular smoother (Jacobi, CG, etc.) to produce $\tilde{x}_f$

Output: $\tilde{x}_f$

In the original work, the coarse-scale problem is solved with GMRES preconditioned by the approximate inverse of the shifted Laplacian system, which is applied via another V-cycle procedure. In this case, we note that the coarse-scale problem is merely an instance of the original problem and since we apply the V-cycle in (5.1) to precondition the original problem, we can recursively apply the same method to precondition the coarse-scale problem as well. This process cannot be iterated beyond two-levels typically because of the minimum grid points per wavelength sampling requirements for wave phenomena [77].

Unfortunately as-is, the method of [51] was only designed with the standard, 7-point stencil in mind, rather than the more robust 27-point stencil of [191]. The work of [165] demonstrates that a straightforward application of the previous preconditioner to the new stencil fails to converge. The authors attempt to extend these ideas to this new stencil by replacing the Jacobi iterations with the CGMN algorithm [31], which acts as a smoother in its own right through the use of Kaczmarz sweeps [145]. For realistically sized problems, this method performs poorly as the CGMN method is inherently sequential and attempts to parallelize it result in degraded convergence performance [31], in particular when implemented in a stencil-based environment. As such, we propose to reuse the existing fast matrix-vector kernels we have developed thus far and set our smoother to be GMRES [222] with an identity preconditioner. Our coarse level solver is FGMRES [221], which allows us to use our nonlinear, iteration-varying preconditioner. Compared to our reference stencil-based Kaczmarz sweep implementation in C, a single-threaded matrix-vector multiplication is 30 times faster. In the context of preconditioning linear systems, this means that unless the convergence rate of the Kaczmarz sweeps are $30/k_s$ faster than the GMRES-based smoothers, we should stick to the faster kernel for our problems. Although the computational complexity of a Kaczmarz sweep is similar to the computational complexity of a matrix-vector product, the performance of the sweeps are cache-bound. In order to speed up this smoother, one could use the CARP-CG algorithm [108], which parallelizes the Kaczmarz sweeps. We also experimentally observed that using a shifted Laplacian preconditioner, as in [51, 165] on the second level caused an increase in the number of outer iterations, slows down convergence. As such, we have replaced preconditioning the shifted Laplacian system by preconditioning the Helmholtz itself, solved with FGMRES, which results in much faster convergence.

A diagram of the full algorithm, which we denote ML-GMRES, is depicted in Figure 5.3. This algorithm is matrix-free, in that we do not have to construct any of the intermediate matrices explicitly and instead merely compute matrix-vector products, which will allow us to apply this method to large systems.

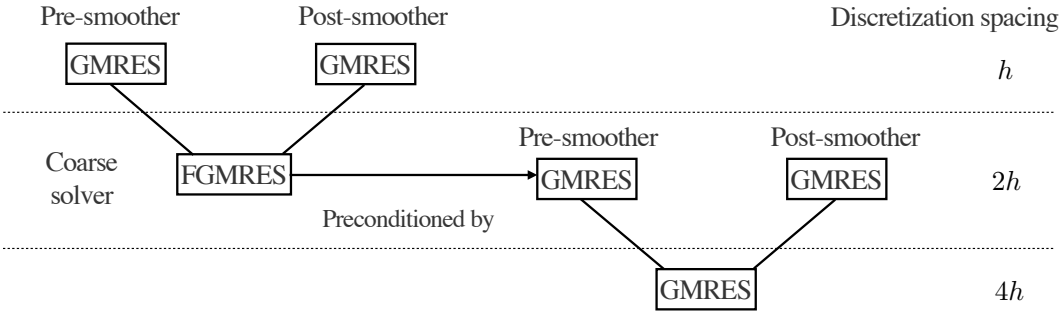


Figure 5.3 ML-GMRES preconditioner. The coarse-level problem (relative to the finest grid spacing) is preconditioned recursively with the same method as the fine-scale problem.

We experimentally study the convergence behaviour of this preconditioner on a 3D constant velocity model and leave a theoretical study of ML-GMRES for future work. By fixing the preconditioner memory vector to be $(k_{s,o}, k_{s,i}, k_{c,o}, k_{c,i}) = (3, 5, 3, 5)$, we can study the performance of the preconditioner as we vary the number of wavelengths in each direction, denoted n_λ, as well as the number of points per wavelength, denoted n_{ppw}. For a fixed domain, as the former quantity increases, the frequency increases while as the latter quantity increases, the grid sampling decreases. As an aside, we prefer to parametrize our problems in this manner as these quantities are independent of the scaling of the domain, velocity, and frequency, which are often obscured in preconditioner examples in research. These two quantities, n_λ and n_{ppw}, on the other hand, are explicitly given parameters and comparable across experiments. We append our model with a number of PML points equal to one wavelength on each side. Using 5 threads per matrix-vector multiply, we use FGMRES with 5 inner iterations as our outer solver, solve the system to a relative residual of 10^{-6}. The results are displayed in Table 5.3.

n_λ/n_{ppw}	6	8	10
5	2 (43^3, 4.6)	2 (57^3, 6.9)	2 (71^3, 10.8)
10	3 (73^3, 28.9)	2 (97^3, 38.8)	2 (121^3, 76.8)
25	8 (161^3, 809)	3 (217^3, 615)	3 (271^3, 1009.8)
40	11 (253^3, 4545)	3 (337^3, 2795)	3 (421^3, 4747)
50	15 (311^3, 11789)	3 (417^3, 5741)	3 (521^3, 10373)

Table 5.3 Preconditioner performance as a function of varying points per wavelength and number of wavelengths. Values are number of outer FGMRES iterations. In parenthesis are displayed the number of grid points (including the PML) and overall computational time (in seconds), respectively.

We upper bound the memory usage of ML-GMRES as follows. We note that the FGMRES(k_o, k_i), GMRES(k_o, k_i) solvers, with k_o outer iterations and k_i inner iterations, store $2k_i + 6$ vectors and $k_i + 5$ vectors, respectively. In solving the discretized $H(m)u = q$ with $N = n^3$ complex-valued unknowns, we require $(2k_i + 6)N$ memory for the outer FGMRES solver, $2N$ for additional vectors in the V-cycle, in addition to $(k_{s,i}+5)N$ memory for the level-1 smoother, $(2k_{c,i}+6)(N/8)$ memory for the level-2 outer solver, $(k_{s,i} + 5)(N/8)$ memory for the level-2 smoother, and $(k_{c,i}+5)(N/64)$ memory for the level-3 solver. The total peak memory of the entire solver is therefore

$$(2k_i+6)N+\max((k_{s,i}+5)N, (2k_{c,i}+6)(N/8)+\max((k_{s,i}+5)N/8, (2k_{c,i}+6)N/64))$$

Using the same memory settings as before, our preconditioner requires at most 26 vectors to be stored. Although the memory requirements of our preconditioner are seemingly large, they are approximately the same cost as storing the entire 27-point stencil coefficients of $H(m)$ explicitly and much smaller than the LU factors in (Table 1, [191]). The memory requirements of this preconditioner can of course be

reduced by reducing k, k_s or k_c, at the expense of an increased computational time per wavefield solve. To compute the real world memory usage of this preconditioner, we consider a constant model problem with 10 wavelengths and a varying number of points per wavelength so that the total number of points, including the PML region, is fixed. In short, we are not changing the effective difficulty of the problem, but merely increase the oversampling factor to ensure that the convergence remains the same with increasing model size. The results in Table 5.4 indicate that ML-GMRES is performing slightly better than expected from a memory point of view and the computational time scaling is as expected.

Grid Size	Time(s)	Peak memory (GB)	Number of vectors of size N
128^3	46	0.61	$20N$
256^3	213	5.8	$23N$
512^3	1899	48.3	$24N$

Table 5.4 Memory usage for a constant-velocity problem as the number of points per wavelength increases

Despite the strong empirical performance of this preconditioner, we note that performance tends to degrade as $n_{\text{ppw}} < 8$, even though the 27–point compact stencil is rated for $n_{\text{ppw}} = 4$ in [191]. Intuitively this makes sense as on the coarsest grid $n_{\text{ppw}} < 2$, which is under the Nyquist limit, and leads to stagnating convergence. In our experience, this discrepancy did not disappear when using the shifted Laplacian preconditioner on the coarsest level. It remains to be seen how multigrid methods for the Helmholtz equation with $n_{\text{ppw}} < 8$ will be developed in future research.

5.5 Numerical Examples

5.5.1 Validation

To verify the validity of this implementation, specifically that the code as implemented reflects the underlying mathematics, we ensure that the following tests pass. The Taylor error test stipulates that, for a sufficiently smooth multivariate function $f(m)$ and an arbitrary perturbation δm, we have that

$$\begin{aligned} f(m + h\delta m) - f(m) &= O(h) \\ f(m + h\delta m) - f(m) - h\langle \nabla f(m), \delta m \rangle &= O(h^2) \\ f(m + h\delta m) - f(m) - h\langle \nabla f(m), \delta m \rangle - \frac{h^2}{2}\langle \delta m, \nabla^2 f(m)\delta m \rangle &= O(h^3). \end{aligned}$$

We verify this behaviour numerically for the least-squares misfit $f(m)$ and for a constant 3D velocity model m and a random perturbation δm by solving for our fields to a very high precision, in this case up to a relative residual of 10^{-10}. As

shown in Figure 5.4, our numerical codes indeed pass this test, up until the point where h becomes so small that the numerical error in the field solutions dominates the other sources of error.

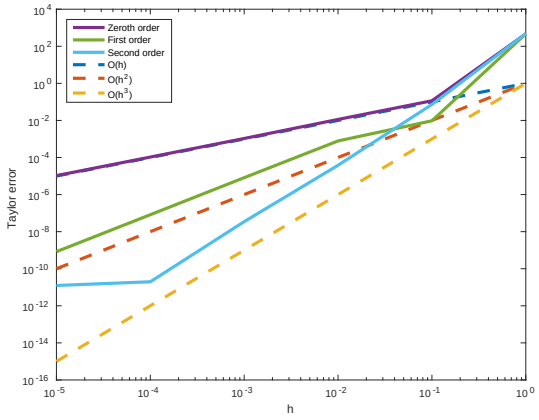


Figure 5.4 Numerical Taylor error for a 3D reference model.

The adjoint test requires us to verify that, for the functions implementing the forward and adjoint matrix-vector products of a linear operator A, we have, numerically,

$$\langle Ax, y\rangle = \langle x, A^H y\rangle,$$

for all vectors x, y of appropriate length. We suffice for testing this equality for randomly generated vectors x and y, made complex if necessary. Owing to the presence of the PML extension operator for the Helmholtz equation, we set x and y, if necessary, to be zero-filled in the PML-extension domain and along the boundary of the original domain. We display the results in Table 5.5.

	$\langle Ax, y\rangle$	$\langle x, A^H y\rangle$	Relative difference
Helmholtz	$6.5755 - 5.2209i \cdot 10^0$	$6.5755 - 5.2209i \cdot 10^0$	$2.9004 \cdot 10^{-15}$
Jacobian	$1.0748 \cdot 10^{-1}$	$1.0748 \cdot 10^{-1}$	$1.9973 \cdot 10^{-9}$
Hessian	$-1.6465 \cdot 10^{-2}$	$-1.646 \cdot 10^{-2}$	$1.0478 \cdot 10^{-10}$

Table 5.5 Adjoint test results for a single instance of randomly generated vectors x, y, truncated to four digits for spacing reasons. The linear systems involved are solved to the tolerance of 10^{-10}.

We also compare the computed solutions in a homogeneous medium to the corresponding analytic solutions in 2D and 3D, i.e.,

$$G(x, x_s) = -\frac{i}{4} H_0(\kappa \|x - x_s\|_2) \qquad \text{in 2D}$$

$$G(x, x_s) = \frac{e^{i\kappa\|x - x_s\|_2}}{4\pi\|x - x_s\|_2} \qquad \text{in 3D}$$

where $\kappa = \dfrac{2\pi f}{v_0}$ is the wavenumber, f is the frequency in Hz, v_0 is the velocity in m/s, and H_0 is the Bessel function of the third kind (the Hankel function). Fig (5.5, 5.6, 5.7) show the analytic and numerical results of computing solutions with the 2D, 3D, and 2.5D kernels, respectively. Here we see that the overall phase dispersion is low for the computed solutions, although we do incur a somewhat larger error around the source region as expected. The inclusion of the PML also prevents any visible artificial reflections from entering the solutions, as we can see from the (magnified) error plots.

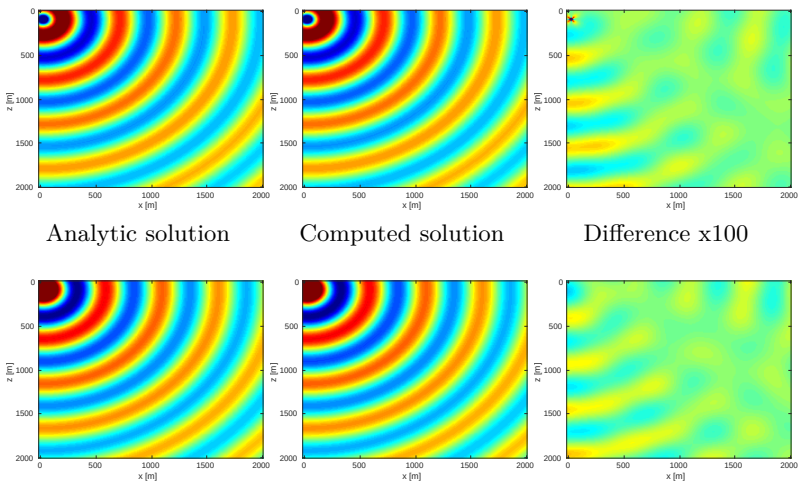


Figure 5.5 Analytic and numerical solutions for the 2D Helmholtz equation for a single source. Difference is displayed on a colorbar 100x smaller than the solutions. Top row is the real part, bottom row is the imaginary part.

5.5.2 Full Waveform Inversion

To demonstrate the effectiveness of our software environment, we perform a simple 2D FWI experiment on a 2D slice of the 3D BG Compass model. We generate data on this 2km x 4.5km model (discretized on a 10m grid) from 3Hz to 18Hz (in 1Hz increments) using our Helmholtz modeling kernel. Employing a frequency continuation strategy allows us to mitigate convergence issues associated with local minima [261]. That is to say, we partition the entire frequency spectrum $3, 4, \ldots, 18$ in to overlapping subsets, select a subset at which to invert invert the model, and use the resulting model estimate as a warm-start for the next frequency band. In our case, we use frequency bands of size 4 with an overlap of 2 between bands. At each stage of the algorithm, we invert the model using 20 iterations of a box-constrained LBFGS algorithm from [223]. An excerpt from the full script that produces this example is shown in Listing (5.2). The results of this algorithm are shown in Figure 5.8. As we are using a band with a low starting frequency (3Hz in

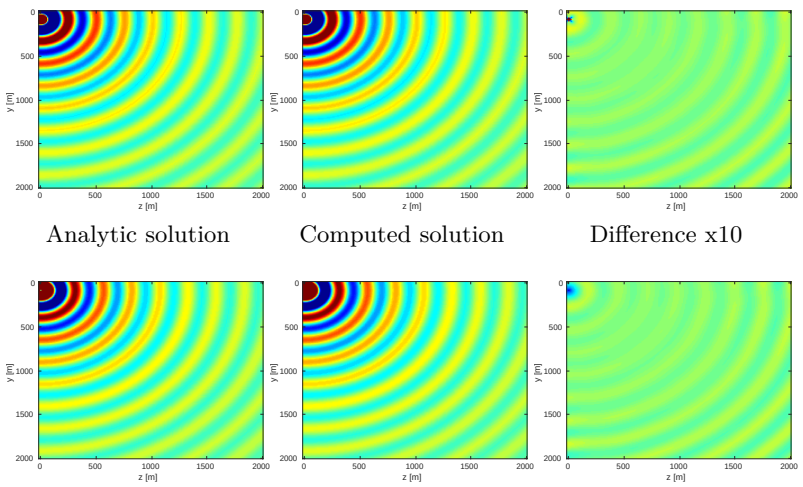


Figure 5.6 Analytic and numerical solutions for the 3D Helmholtz equation (depicted as a 2D slice) for a single source. Difference is displayed on a colorbar 10x smaller than the solutions. Top row is the real part, bottom row is the imaginary part.

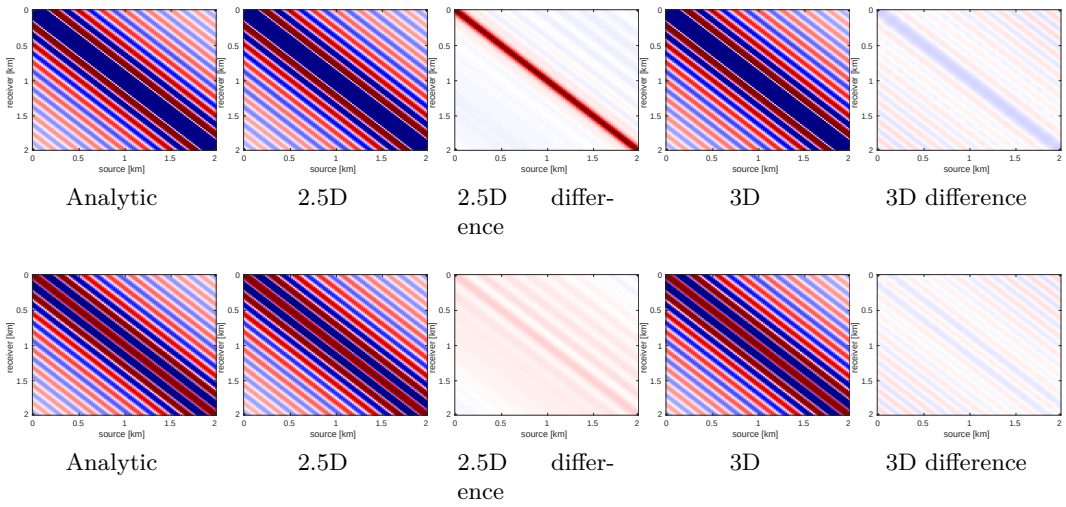


Figure 5.7 Analytic and numerical solutions for the 2.5D Helmholtz system for a generated data volume with 100 sources, 100 receivers, and 100 y-wavenumbers. The 2.5D data took 136s to generate and the 3D data took 8200s, both on a single machine with no data parallelization. Top row: real part, bottom row: imaginary part.

this case), FWI is expected to perform well in this case, which we see that it does. Although an idealized experimental setup, our framework allows for the possibility of testing out algorithms that make use of higher starting frequencies, or use frequency extrapolation as in [263, 170], with minimal code changes.

```
% Set the initial model (a smoothed version of the true model)
mest = m0;
% Loop over subsets of frequencies
for j=1:size(freq_partition,1)
    % Extract the current frequencies at this batch
    fbatch = freq_partition(j,:);
    % Select only sources at this frequency batch
    srcfreqmask = false(nsrc,nfreq);
    srcfreqmask(:,fbatch) = true;
    params.srcfreqmask = srcfreqmask;
    % Construct objective function for these frequencies
    obj = misfit_setup(mest,Q,Dobs,model,params);
    % Call the box constrained LBFGS method
    mest = minConf_TMP(obj,mest,mlo,mhi,opts);
end
```

Listing 5.2 Excerpt from the script that produces this example

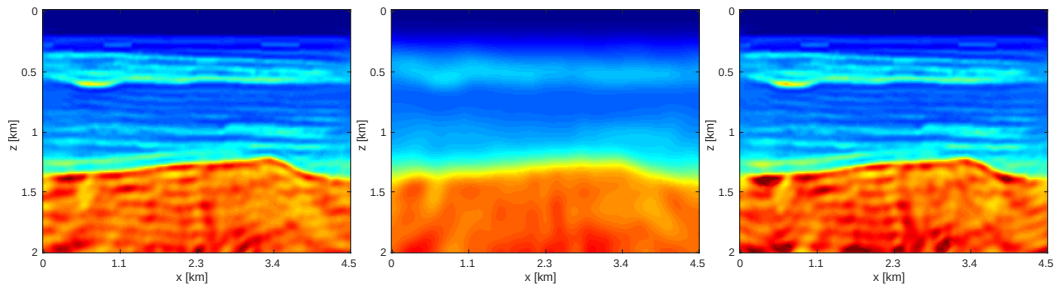


Figure 5.8 True (left) and initial (middle) and inverted (right) models.

5.5.3 Sparsity Promoting Seismic Imaging

The seismic imaging problem aims to reconstruct a high resolution reflectivity map δm of the subsurface, given some smooth background model m_0, by inverting the (overdetermined) Jacobian system

$$J(m_0)\delta m \approx \delta D.$$

In this simple example, δD is the image of the true perturbation under the Jacobian. Attempting to tackle the least-squares system directly

$$\min_{\delta m} \sum_{i_s \in S} \sum_{i_f \in F} \|J_{i_s,i_f} \delta m - \delta D_{i_s,i_f}\|_2^2,$$

where S indexes the sources and F indexes the frequencies, is computationally daunting due to the large number of sources and frequencies used. One straightforward approach is to randomly subsample sources and frequencies, i.e., choose $S' \subset S$ and $F' \subset F$ and solve

$$\min_{\delta m} \sum_{i_s \in S'} \sum_{i_f \in F'} \|J_{i_s,i_f} \delta m - \delta D_{i_s,i_f}\|_2^2,$$

but the Jacobian can become rank deficient in this case, despite it being full rank normally. One solution to this problem is to use sparse regularization coupled with randomized subsampling in order to force the iterates to head towards the true perturbation while still reducing the per-iteration costs. There have been a number of instances of incorporating sparsity of a seismic image in the Curvelet domain [55, 56], in particular [131, 127, 249]. We use the Linearized Bregman method [195, 47, 46], which solves

$$\begin{aligned} &\min_x \|x\|_1 + \lambda \|x\|_2^2 \\ &\text{s.t.} \quad Ax = b. \end{aligned}$$

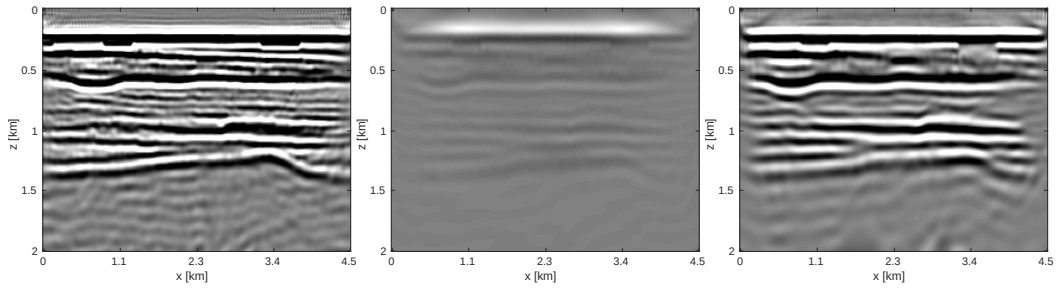
Coupled with random subsampling as in [132], the iterations are shown in Algorithm (5.2), a variant of which is used in [63]. Here $S_\lambda(x) = \text{sign}(x) \max(0, |x| - \lambda)$ is the componentwise soft-thresholding operator. In Listing (5.3), the reader will note the close adherence of our code to Algorithm (5.2), aside from some minor bookkeeping code and pre- and post-multiplying by the curvelet transform to ensure the sparsity of the signal. In this example, we place 300 equispaced sources and 400 receivers at the top of the model and generate data for 40 frequencies from 3-12 Hz. At each iteration of the algorithm, we randomly select 30 sources and 10 frequencies (corresponding to the 10 parallel workers we use) and set the number of iterations so that we perform 10 effective passes through the entire data. Compared to the image estimate obtained from an equivalent (in terms of number of PDEs solved) method solving the least-squares problem with the LSMR [101] method, the randomly subsampled method has made significantly more progress towards the true solution, as shown in Figure 5.9.

Algorithm 5.2 Linearized Bregman with per-iteration randomized subsampling

for $k = 1, 2, \ldots, T$
  Draw a random subset of indices I_k
  $z_{k+1} \leftarrow z_k - t_k A_{I_k}^T (b_{I_k} - A_{I_k} x_k)$
  $x_{k+1} \leftarrow S_\lambda(x_k)$

```
for k=1:T
  % Draw a random subset of sources and frequencies
  Is = rand_subset(nsrc,round(0.2*nsrc));
  If = rand_subset(nfreq,parpool_size());
  % Mask the objective to the sources/frequencies drawn
  srcfreqmask = false(nsrc,nfreq);
  srcfreqmask(Is,If) = true;
  params.srcfreqmask = srcfreqmask;
  % Construct the subsampled Jacobian operator + data
  A = oppDF(m0,Q,model,params);
  b = distributed_subsample_data(b_full,Is,If);
  % Linearized Bregman algorithm
  r = A*x-b;
  ATr = A'*r;
  t = norm(r)^2/norm(ATr)^2;
  z = z - t*ATr;
  x = C'*softThreshold(C*z,lambda);
end
```

Listing 5.3 Excerpt from the code that produces this example



True image | Full data inversion | Linearized Bregman inversion

Figure 5.9 Sparse seismic imaging - full data least-squares inversion versus linearized Bregman with randomized subsampling

5.5.4 Electromagnetic Conductivity Inversion

To demonstrate the modular and flexible nature of our software framework, we replace the finite difference Helmholtz PDE with a simple finite volume discretization of the variable coefficient Poisson equation

$$\nabla(\sigma \cdot \nabla u) = q$$

where σ is the spatially-varying conductivity coefficient. We discretize the field on the vertices of a regular rectangular mesh and σ at the cell centers, which results in

the system matrix, denoted $H(\sigma)$, written abstractly as

$$H(\sigma) = \sum_{i=1}^{5} A_i \text{diag}(B_i \sigma)$$

for constant matrices A_i, B_i. This form allows us to easily derive expressions for $T : \delta\sigma \mapsto DH(\sigma)[\delta m]u$ and T^* as

$$T\delta\sigma = \sum_{i=1}^{5} A_i \text{diag}(B_i \delta\sigma) u$$
$$T^* z = \sum_{i=1}^{5} B_i^H \text{diag}(\overline{u}) A_i^H z.$$

With these expressions in hand, we merely can slot the finite volume discretization and the corresponding directional derivative functions in to our framework without modifying any other code.

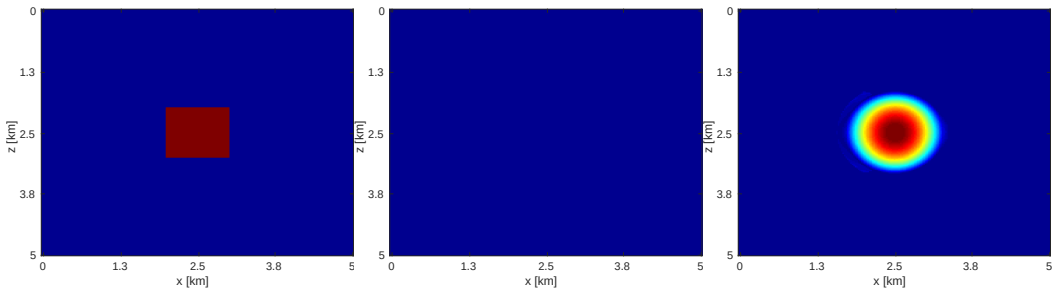
Consider a simple constant conductivity model containing a square anomaly with a 20% difference compared to the background, encompassing a region that is 5km x 5km with a grid spacing of 10m, as depicted in Figure 5.10. We place 100 equally spaced sources and receivers at depths $z = 400m$ and $z = 1600m$, respectively. The pointwise constraints we use are the true model for $z < 400m$ and $z > 1600m$ and, for the region in between, we set $\sigma_{\min} = \min_x \sigma(x)$ and $\sigma_{\max} = 2 \max_x \sigma(x)$. Our initial model is a constant model with the correct background conductivity, shown in Figure 5.10. Given a current estimate of the conductivity σ_k, we minimize a quadratic model of the objective subject to bound constraints, i.e.,

$$\sigma_{k+1} = \underset{\sigma}{\text{argmin}} \langle \sigma - \sigma_k, g_k \rangle + \frac{1}{2} \langle \sigma - \sigma_k, H_k(\sigma - \sigma_k) \rangle$$
$$\text{s.t.} \quad \sigma_{\min} \leq \sigma \leq \sigma_{\max}$$

where g_k, H_k are the gradient and Gauss-Newton Hessian, respectively. We solve 5 of these subproblems, using 5 objective/gradient evaluations to solve each subproblem using the bound constrained LBFGS method of [223]. As we do not impose any constraints on the model itself and the PDE itself is smoothing, we are able to recover a very smooth version of the true model, shown in Figure 5.10, with the attendant code shown in Listing (5.4). This discretization and optimization setup is by no means the optimal method to invert such conductivity models but we merely outline how straightforward it is to incorporate different PDEs in to our framework. Techniques such as total variation regularization [2] can be incorporated in to this framework by merely modifying our objective function.

```
obj = misfit_setup(sigma0,Q,D,model,params);
sigma = sigma0;
for i=1:5
  % Evaluate objective
  [f,g,h] = obj(sigma);
  % Construct quadratic model
  q = @(x) quadratic_model(g,h,x,sigma);
  % Minimize quadratic model, subject to pointwise constraints
  sigma = minConf_TMP(q,sigma,sigma_min,sigma_max,opts);
end
```

Listing 5.4 Excerpt from the script that produces this example



True Model Initial Model Inverted Model

Figure 5.10 Inversion results when changing the PDE model from the Helmholtz to the Poisson equation

5.5.5 Stochastic Full Waveform Inversion

Our software design makes it relatively straightforward to apply the same inversion algorithm to both a 2D and 3D problem in turn, while changing very little in the code itself. We consider the following algorithm for inversion, which will allow us to handle the large number of sources and number of model parameters, as well as the necessity of pointwise bound constraints [128] on the intermediate models. Our problem has the form

$$\min_m \frac{1}{N_s} \sum_{i=1}^{N_s} f_i(m)$$
$$\text{s.t.} \quad m \in C$$

where m is our model parameter (velocity or slowness), $f_i(m) = \frac{1}{2}\|P_r H(m)^{-1} q_i - d_i\|_2^2$ is the least-squares misfit for source i, and C is our convex constraint set, which is $C = \{m : m_{LB} \leq m \leq m_{UB}\}$ in this case. When we have p parallel processes and $N_s \gg p$ sources, in order to efficiently make progress toward the solution of the above problem, we stochastically subsample the objective and approximately

minimize the resulting problem, i.e., at the kth iteration we solve

$$m_k = \underset{m}{\operatorname{argmin}} \frac{1}{|I_k|} \sum_{i \in I_k} f_i(m)$$
$$\text{s.t.} \quad m \in C$$

for $I_k \subset \{1, \ldots, N_s\}$ drawn uniformly at random and $|I_k| = p \ll N_s$. We use the approximate solution m_k as a warm start for the next problem and repeat this process a total of T times. Given that our basic unit of work in these problems is computing the solution of a PDE, we limit the number of iterations for each subproblem solution so that each subproblem can be evaluated at a constant multiple of evaluating the objective and gradient with the full data, i.e., $r_{\text{sub}}\lceil \frac{N_s}{p} \rceil$ iterations for a constant r_{sub}. If an iteration stagnates, i.e., if the line search fails, we increase the size of $|I_k|$ by a fixed amount. This algorithm is similar in spirit to the one proposed in [257].

2D Model BG Compass

We apply the above algorithm to the BG Compass model, with the same geometry and source/receiver configuration as outlined in Section (5.5.2). We limit the total number of passes over the entire data to be equal to 50% of those used in the previous example, with 10 re-randomization steps and $r_{\text{sub}} = 2$. Our results are shown in Figure 5.11. Despite the smaller number of overall PDEs solved, the algorithm converges to a solution that is qualitatively hard to distinguish from Figure 5.8. The overall model error as a function of number of subproblems solved is depicted Figure 5.12. The model error stagnates as the number of iterations in a given frequency batch rises and continues to decrease when a new frequency batch is drawn.

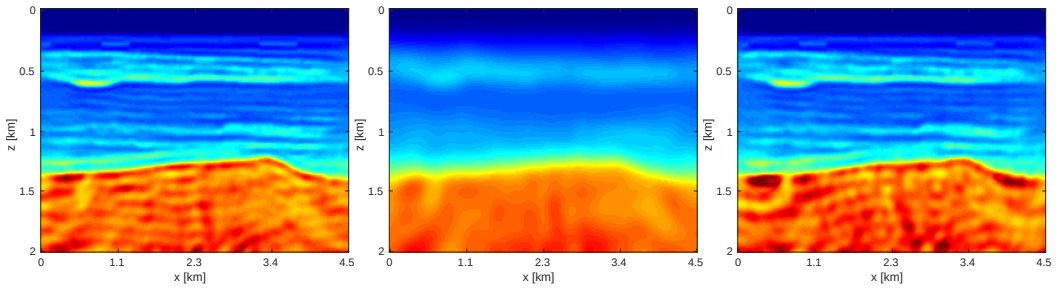


Figure 5.11 True (left) and initial (middle) and inverted (right) models

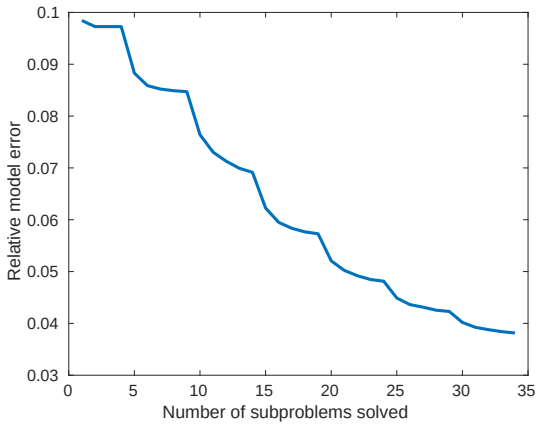


Figure 5.12 Relative model error as a function of the number of randomized subproblems solved.

3D Model - Overthrust

We apply the aforementioned algorithm to the SEG/EAGE Overthrust model, which spans 20km x 20km x 5km and is discretized on a 50m x 50m x 50m grid with a 500m deep water layer and minimum and maximum velocities of $1500m/s$ and $6000m/s$. The ocean floor is covered with a 50 x 50 grid of sources, each with 400m spacing, and a 396 x 396 grid of receivers, each with 50m spacing. The frequencies we use are in the range of $3 - 5.5Hz$ with $0.25Hz$ sampling, corresponding to 4s of data in the time domain, and inverted a single frequency at a time. The number of wavelengths in the x, y, z directions vary from $(40, 40, 10)$ to $(73, 73, 18)$, respectively, with the number of points per wavelength varying from 9.8 at the lowest frequency to 5.3 at the highest frequency. The model is clamped to be equal to the true model in the water layer and is otherwise allowed to vary between the maximum and minimum velocity values. In practical problems, some care must be taken to ensure that discretization discrepancies between the boundary of the water and the true earth model are minimized. Our initial model is a significantly smoothed version of the true model and we limit the number of stochastic redraws to $T = 3$, so that we are performing the same amount of work as evaluating the objective and gradient three times with the full data. The number of unknown parameters is 14,880,000 and the fields are inverted on a grid with 39,859,200 points, owing to the PML layers in each direction. We use 100 nodes with 4 Matlab workers each of the Yemoja cluster for this computation. Each node has 128GB of RAM and a 20-core Intel processor. We use 5 threads for each matrix-vector product and subsample sources so that each Matlab worker solves a single PDE at a time (i.e., $|I_k| = 400$ in the above formulation) and set the effective number of passes through the data for each subproblem to be one, i.e., $r_{\text{sub}} = 1$. Despite the limited number of passes through the data, our code is able to make substantial progress towards the true model, as shown in Figures 5.14 and 5.15. Unlike in the 2D case, we are limited in our ability to invert higher frequencies in a reasonable amount of time and therefore our

inverted results do not fully resolve the fine features, especially in the deeper parts of the model.

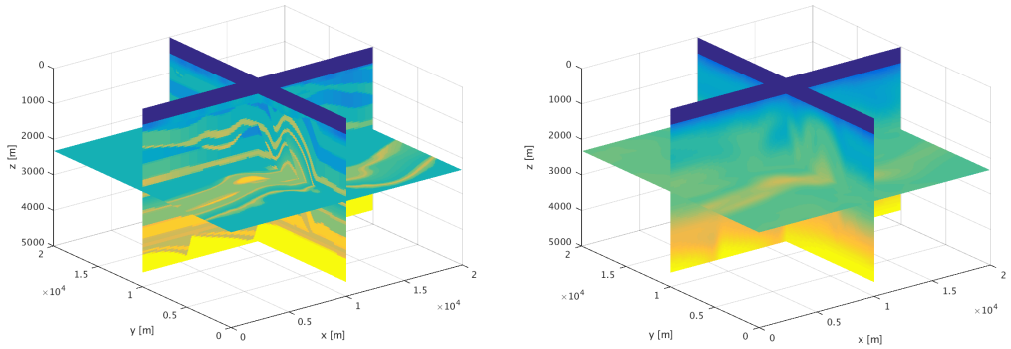


Figure 5.13 True (left) and initial (right) models

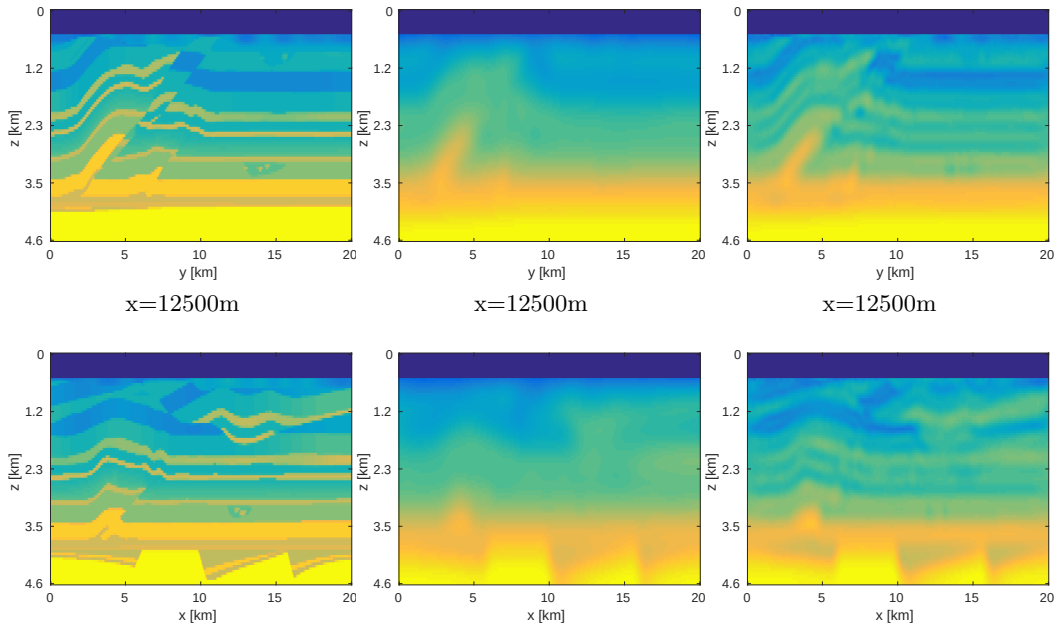


Figure 5.14 True model (left), initial model (middle), inverted model (right) for a variety of fixed coordinate slices

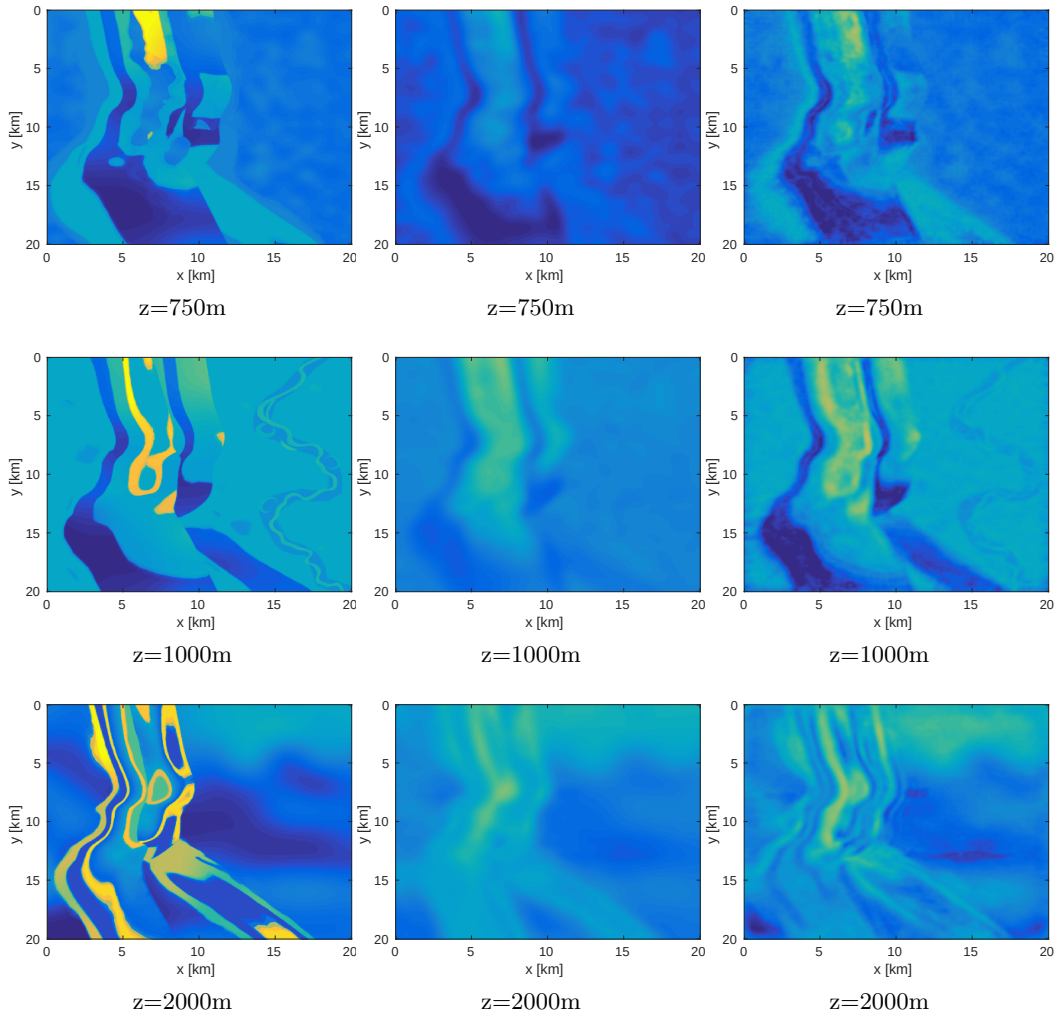


Figure 5.15 True model (left), initial model (middle), inverted model (right) for a variety of fixed z coordinate slices

5.6 Discussion

There are a number of problem-specific constraints that have motivated our design decisions thus far. Computing solutions to the Helmholtz equation in particular is challenging due to the indefiniteness of the underlying system for even moderate frequencies and the sampling requirements of a given finite difference stencil. These challenges preclude one from simply employing the same sparse-matrix techniques in 2D for the 3D case. Direct methods that store even partial LU decomposition of the system matrix are infeasible from a memory perspective, unless one allows for using multiple nodes to compute the PDE solutions. Even in that case, one can run in to resiliency issues in computational environments where nodes have a non-zero probability of failure over the lifetime of the computation. Regardless of the dimen-

sionality, our unified interface for multiplying and dividing the Helmholtz system with a vector abstracts away the implementation specific details of the underlying matrix, while still allowing for high performance. These design choices give us the ability to scale from simple 2D problems on 4 cores to realistically sized 3D problems running on 2000 cores with minimal engineering effort. The acoustic Helmholtz equation itself is a model resulting from a number of simplifying assumptions made on the physical properties of the earth, such as constant density and ignoring elastic effects [228]. These assumptions are made in the interest of reducing the computational complexity of solving the PDEs, but simultaneously reduce the explanatory power of the model. If one has a sufficiently powerful computational environment, these simplified assumptions can be removed and the resulting PDE solution kernels can be integrated in to this software system.

Although our choice of Matlab has enabled us to succinctly design and prototype our code, there have been a few stumbling blocks as a result of this language choice. There is an onerous licensing issue for the Parallel Toolbox, which makes scaling to a large number of workers costly. The Parallel Toolbox is built on MPI, which is simply insufficient for performing large scale computations in an environment that can be subject to disconnections and node failures and cannot be swapped out for another parallelization scheme within Matlab itself. In an environment where one does not have full control over the computational hardware, such as on Amazon's cloud computing services, this paradigm is untenable. For interfacing with the C/Fortran language, there is a large learning curve for compiling MEX files, which are particularly constructed C files that can be called from with Matlab. Matlab offers its Matlab Coder product that allows one to, in principle, compile any function in to a MEX file, and thus reap potential performance benefits. The compilation process has its limits in terms of functionality, however, and cannot, for instance, compile our framework easily.

Thankfully, there have been a few efforts to help alleviate these issues. The relatively new numerical computing language Julia has substantially matured in the last few years, making it a viable competitor to Matlab. Julia aims to bridge the gap between an interpreted and a compiled language, the former being easier to prototype in and the latter being much faster by offering a just-in-time compiler, which balances ease of use and performance. Julia is open source, whereas Matlab is decidedly not, allows for a much more fine-grained control over parallelization, and has other attractive features such as built-in interfacing to C, is strongly-typed, although flexibly so, and has a large, active package ecosystem. We aim to reimplement the framework described in this paper in Julia in the future. The Devito framework [166] is another such approach to balance the high-level mathematics and low-level performance in PDE-constrained problems specifically through the compilation of symbolic Python to C on-the-fly. Devito incurs some initial setup time to process the symbolic expressions of the PDEs and compile them in to runnable C binaries, but this overhead is negligible compared to the cost of time-stepping solutions and only has to be performed once for a PDE with fixed parameters. This is one possible option to speed up matrix-vector products, or allow for user-specified order of

accuracy at runtime, if the relevant complex-valued extensions can be written. We leave this as an option for future work.

5.7 Conclusion

The designs outlined in this paper make for an inversion framework that successfully obfuscates the inner workings of the construction, solution, and recombination of solutions of PDE systems. As a result, the high-level interfaces exposed to the user allow a researcher to easily construct algorithms dealing with the outer structure of the problem, such as stochastic subsampling or solving Newton-type methods, rather than being hindered by the complexities of, for example, solving linear systems or distributing computation. This hierarchical and modular approach allows us to delineate the various components associated to these computations in a straightforward and demonstrably correct way, without sacrificing performance. Moreover, we have demonstrated that this design allows us to easily swap different PDE stencils, or even PDEs themselves, while still keeping the outer, high-level interfaces intact. This design allows us to apply a large number high-level algorithms to large-scale problems with minimal amounts of effort.

With this design, we believe that we have struck the right balance between readability and performance and, by exposing the right amount of information at each level of the software hierarchy, researchers should be able to use this codebase as a starting point for developing future inversion algorithms.

Chapter 6

Conclusion

Inverse problems are an important class of problems that enable us to indirectly estimate parameters of a region from surface measurements. The acquired dsata is high-dimensional and often has missing entries due to practical constraints. There is a large theoretical and practical interest in having fully sampled and noise-free data without the need to acquire all of the samples of the volume in the field. As such, developing methods to interpolate large multidimensional volumes when the acquired data is subsampled and noisy is of great importance. Once one has an accurate estimate of the fully sampled data volume, one must come to terms with the enormous complexity and large number of components that must be dealt with (from a software perspective) when solving the inverse problem itself. In this thesis, we have proposed techniques that are particular to the three subtopics of missing data completion for low rank tensors, convex composite optimization, and inverse problem software design. In what follows, we discuss our contributions to these topics as well as various open questions that remain on these topics.

Manifold Tensor Completion Chapter 2 introduced the smooth manifold of Hierarchical Tucker tensors, following the theoretical developments of [253]. By exploiting the quotient geometry inherent in HT tensors, along with the description of the horizontal space, we introduced a Riemannian metric on to this manifold that is invariant under the group action. As a result, this metric extends to the underlying quotient manifold. From a numerical point of view, we can compute inner products using the concrete HT parameters that respects the group action and thus are implicitly computing these quantities on the abstract manifold of equivalence classes in a well-defined manner. Equipped with this Riemannian metric, we derived the expressions for the Riemannian gradient that can be computed efficiently using multilinear products, and considered a sparse-data version of these expressions that avoid having to form the full data volume. We also specified other important components of solving optimization problems on the HT manifold by deriving a retraction, vector transport, and a line search method. In order to speed up the convergence of the resulting optimization algorithms, we derived simplified expressions for the Gauss-Newton Hessian that can be computed through the associated

Gramian matrices, which encode the singular values of the full tensor. These matrices, and their derivatives, can be used to impose regularization on missing data interpolation problems with high subsampling regimes by keeping the algorithm iterates away from the boundary of the manifold. Unlike for the Tucker case, this regularization is explicitly available for algorithmic use, as well as for theoretical considerations. Using this construction, we prove that our algorithms converge to a stationary point, which is typically sufficient when analyzing nonconvex problems. In practice, these methods appear to recover low rank tensors successfully and we demonstrate the effectiveness of this approach on a number of large-scale seismic interpolation examples. Our methods are efficient and can recover volumes with high subsampling ratios.

There are a number of open questions pertaining to tensor completion that are important both practically and theoretically. The necessity of working with this nonconvex but low-rank parametrization of this class of tensors arises from the fact that explicitly storing and computing the full volumes is simply not feasible. As a result, unlike in nuclear-norm minimization problems where the matrix variable is stored explicitly, we must have some estimate on the underlying ranks of the HT format in order for it to estimate the unknown tensor accurately. Cross-validation to estimate these ranks becomes computationally daunting, given that we have a vector of parameters over which to iterate. For tensors arising from the discretization of a multivariate function, the ability to decompose it in to low-rank factors can be qualitatively linked to the smoothness of the function itself [224]. These bounds are not tight and unfortunately uninformative when handling a given tensor. It remains to be seen as to how to determine these ranks in a rigorous manner apriori. As for the tensor completion problem itself, very little has been rigorously shown in the literature that the solution to the nonconvex optimization program is close to the true unknown tensor. Results have been shown when the sampling operator is subgaussian [211] or consists of partial Fourier measurements [212], but being able to prove that the tensor can be recovered under pointwise sampling, as for the matrix case, remains an open problem.

Convex Composite Optimization Chapter 3 is a precursor to Chapter 4, wherein we prove results about smooth convex functions satisfying the Polyak-Lojasiewicz inequality. Such functions are in some sense the most general class of convex functions that converge linearly towards a global minimizer with steepest descent iterations. Although strongly convex functions satisfy the PL inequality, they are not the only ones that do so and in this chapter we examine the effect of various transformations of a function on its PL constant. We study the behaviour of the PL constant under composition with a smooth mapping. We also prove that Moreau regularization can only decrease the PL-constant of a given function and there is a natural upper bound for the PL constant of the Moreau envelope of a function, irrespective of the function itself. The functions that realize this upper bound are indicator functions for convex sets. This in turn shows that distance functions satisfy the PL inequality despite not being strongly convex, which we use in the following chapter. We also examined the PL behaviour of the well-known Hu-

ber function, which is the Moreau envelope of the $\ell 1$ norm. Although the original function $\|x\|_1$ does not satisfy the PL-inequality, the Huber function does and we characterize the region on which this inequality holds.

We study the solution to convex composite optimization programs in Chapter 4, which are a general class of (usually nonconvex) problems given by minimizing a non-smooth but convex function composed with a smooth inner function. The non-smoothness of the outer function prevents methods that solve the problem directly from scaling, since such methods are inherently saddled by having to solve non-smooth subproblems. Non-smooth convex optimization has sublinear convergence in the worst-case, which is far too slow of a rate when the problem sizes become large. Instead, we propose to use a level set method that switches the role of the objective and constraints, resulting in smooth subproblems with simple constraints. By studying the associated value function, which parametrizes the tradeoff between increasing the objective to its minimum value and satisfying the constraints, we employ a secant or Newton method to update this parameter and converge towards a solution. As the secant method converges superlinearly and the Newton method converges quadratically, we have to update our cooling parameter only a small number of times. Despite the fact that our subproblems are non-convex and the distance function itself lacks strong convexity, if the nonlinear mapping is sufficiently well-behaved, we prove that steepest descent converges linearly to a solution. We apply this method to a variety of convex and nonconvex problems, many of which can be construed as analysis problems (in the sense of compressed sensing). This method is successful at interpolating seismic data, performing total variation deblurring, declipping audio signals, and performing robust low-rank tensor completion, also known as robust tensor principal component analysis. In particular for robust completion, the resulting codes are very simple to implement as they merely involve a minor change in the objective function. This fact allows us to derive robust tensor completion from the code used in Chapter 2 with little modification. Although we are able to handle nonconvex $\ell 0$ problems, which are NP-complete, we show that as the number of parameters increases, the associated computational times increase drastically as one would expect when solving such problems. Our method is computationally attractive as we do not introduce additional hyperparameters that must be empirically estimated from the unknown signal.

Despite the success of our approach for transform-based problems, it is not altogether clear how one could solve these subproblems in a stochastic manner, when c and h are sufficiently separable. One may have to forgo the variable projection altogether and formulate algorithms on the joint (x, z) variables. The large amount of existing research on stochastic optimization methods, which has demonstrated the effectiveness of these methods when the dimensionality and number of data points are high, suggest that this avenue would be fruitful for solving realistically-sized problems. Additionally, what conditions should one impose on the nonlinear mapping c so that the resulting subproblems $v(\tau)$ satisfy the conditions of Theorem 4.5? Namely that $v(\tau + \Delta\tau) = v_{lin}(\tau + \Delta\tau) + O(|\Delta\tau|^2)$ for the expression of $v'(\tau)$ to be given by the theorem. Intuitively, the level set constraint $h(z) \leq \tau$ only acts on the

z variable and so we would expect this expression for $v'(\tau)$ to hold irrespective of whether c is linear or nonlinear. Numerically, this expression for $v'(\tau)$ appears to hold when compared to the finite difference approximation $v'(\tau) \approx (v(\tau+h)-v(\tau))/h$ for the nonconvex robust tensor completion problem, and so it may be the case that the Hierarchical Tucker mapping ϕ is sufficiently close to its linearization so that the theorem holds. It is also interesting to ask how one could extend the PL-inequality convergence analysis to Newton or quasi-Newton methods and how the resulting theory would manifest for $\mathcal{C}^2$ convex functions.

Inverse Problem Software Design In Chapter 5, we consider solving large scale inverse problems from a software design perspective. Given the enormous engineering challenges involved in solving such problems, in addition to the large number of scientific fields one must be versed in in order to tackle such problems, previous efforts to design such frameworks have either been well-designed or high performance, but rarely both. This chapter showcases a new organization for solving inverse problems that decouples the components in to discrete modules that are responsible for specific tasks and organized in a hierarchical fashion. As a result, one has flexibility in replacing a baseline implementation of each component with an implementation that suits a particular users needs. This separation of responsibilities also enables us to interface with low-level C code when needed for matrix-vector products while keeping the rest of the code in high-level Matlab, which makes it significantly easier to comprehend, maintain, and extend. Several extensions of this approach are provided to 2.5D inversion and so-called waveform reconstruction inversion, which can be integrated in to the entire framework with minimal effort. Some advantages afforded by this design are that the code is able to solve problems irrespective of their dimensionality, switching automatically between efficient sparse matrix algebra for 2D problems, where the memory requirements are less daunting, and efficient stencil-based Krylov methods for 3D problems. We also introduce a new preconditioner for Helmholtz systems based on previous recursive multigrid-based preconditioners, which empirically requires a constant number of iterations if the points per wavelength of the discretized problem is sufficiently high. As we form all of our matrix-vector products implicitly, our preconditioner requires little memory– on the order of storing the sparse system matrix explicitly,. It is also able to take advantage of local parallel resources rather than requiring internode parallelism in order to scale effectively to distributed systems with occasional node failures. Our codebase passes several necessary unit tests to ensure that the code as written reflects the underlying mathematics of the problem. We showcase our framework on a variety of nonlinear and linear inversion problems, with accompanying scripts that closely mirror the associated high level algorithms. Our framework is able to handle randomized subsampling, more complex algorithms such as Linearized Bregman, inverse problems with different associated PDEs, as well as large scale 3D problems.

In future work, we plan to extend this framework to cloud-based computing, at which point we will have to forgo a Matlab implementation and move towards a tractable language such as Julia. An open algorithmic question that remains is, given the Nyquist sampling criteria of two points per wavelength for oscillatory

PDE solutions, is it possible to design a multigrid preconditioner for the Helmholtz equation that employs two coarsened levels? Coarsening to two levels, as opposed to one, allows us to greatly reduce the computational cost of inverting the lower-level systems, but impose a sampling requirement of eight points per wavelength on the finest level, which is too stringent. The 3D stencil we implement from [191] has low dispersion down to four points per wavelength, but this minimum threshold is somewhat wasted on the high sampling at the finest level.

Bibliography

[1] P.A. Absil, R. Mahony, and R. Sepulchre. *Optimization algorithms on matrix manifolds*. Princeton Univ Press, 2008. → pages 6, 23, 24, 30, 31, 32, 33, 39, 41

[2] A Abubaker and Peter M Van Den Berg. Total variation as a multiplicative constraint for solving inverse problems. *IEEE Transactions on Image Processing*, 10(9):1384–1392, 2001. → pages 116

[3] Evrim Acar, Daniel M Dunlavy, and Tamara G Kolda. A scalable optimization approach for fitting canonical tensor decompositions. *Journal of Chemometrics*, 25(2):67–86, 2011. → pages 12

[4] Amir Adler, Valentin Emiya, Maria G Jafari, Michael Elad, Rémi Gribonval, and Mark D Plumbley. Audio inpainting. *IEEE Transactions on Audio, Speech, and Language Processing*, 20(3):922–932, 2012. → pages 79

[5] Andy Adler, Romina Gaburro, and William Lionheart. Electrical impedance tomography. In *Handbook of Mathematical Methods in Imaging*, pages 599–654. Springer, 2011. → pages 95

[6] Vinicius Albani, Uri M. Ascher, Xu Yang, and Jorge P. Zubelli. Data driven recovery of local volatility surfaces. *Inverse Problems and Imaging*, 11(5):2–2, 2017. ISSN 1930-8337. doi: 10.3934/ipi.2017038. → pages 2

[7] Vinicius Albani, Uri M Ascher, and Jorge P Zubelli. Local volatility models in commodity markets and online calibration. *Journal of Computational Finance*, To Appear. → pages 3

[8] Patrick R Amestoy, Iain S Duff, and J-Y L'excellent. Multifrontal parallel distributed symmetric and unsymmetric solvers. *Computer methods in applied mechanics and engineering*, 184(2):501–520, 2000. → pages 93

[9] A. Y Aravkin, R. Kumar, H. Mansour, B. Recht, and F. J Herrmann. A robust SVD-free approach to matrix completion, with applications to interpolation of large scale data. Technical report, University of British Columbia, February 2013. → pages 5

[10] Aleksandr Aravkin, Tristan Van Leeuwen, and Felix Herrmann. Robust full-waveform inversion using the Student's t-distribution. In *SEG Technical Program Expanded Abstracts 2011*, pages 2669–2673. Society of Exploration Geophysicists, 2011. → pages 7, 95

[11] Aleksandr Aravkin, Michael P Friedlander, Felix J Herrmann, and Tristan Van Leeuwen. Robust inversion, dimensionality reduction, and randomized sampling. *Mathematical Programming*, pages 1–25, 2012. → pages 95, 98

[12] Aleksandr Aravkin, Rajiv Kumar, Hassan Mansour, Ben Recht, and Felix J Herrmann. Fast methods for denoising matrix completion formulations, with applications to robust seismic data interpolation. *SIAM Journal on Scientific Computing*, 36(5):S237–S266, 2014. → pages 2, 5

[13] Aleksandr Aravkin, Dmitriy Drusvyatskiy, and Tristan van Leeuwen. Variable projection without smoothness. *arXiv preprint arXiv:1601.05011*, 2016. → pages 65

[14] Aleksandr Y Aravkin and Tristan Van Leeuwen. Estimating nuisance parameters in inverse problems. *Inverse Problems*, 28(11):115016, 2012. → pages 9, 65, 104

[15] Aleksandr Y Aravkin, James V Burke, and Michael P Friedlander. Variational properties of value functions. *SIAM Journal on optimization*, 23 (3):1689–1717, 2013. → pages 70, 71

[16] Aleksandr Y. Aravkin, Rajiv Kumar, Hassan Mansour, Ben Recht, and Felix J. Herrmann. Fast methods for denoising matrix completion formulations, with applications to robust seismic data interpolation. *SIAM Journal on Scientific Computing*, 36(5):S237–S266, 10 2014. doi: 10.1137/130919210. URL https://www.slim.eos.ubc.ca/Publications/Public/Journals/SIAMJournalOnScientificComputing/2014/aravkin2014SISCfmd/aravkin2014SISCfmd.pdf. (SISC). → pages 53

[17] Aleksandr Y Aravkin, James V Burke, Dmitriy Drusvyatskiy, Michael P Friedlander, and Scott Roy. Level-set methods for convex optimization. *arXiv preprint arXiv:1602.01506*, 2016. → pages 64, 89

[18] Simon R Arridge. Optical tomography in medical imaging. *Inverse problems*, 15(2):R41, 1999. → pages 1

[19] B. W. Bader, T. G. Kolda, et al. Matlab tensor toolbox version 2.5. http://www.sandia.gov/~tgkolda/TensorToolbox/, Accessed January 1, 2012. → pages 12

[20] Satish Balay, J Brown, Kris Buschelman, Victor Eijkhout, W Gropp, D Kaushik, M Knepley, L Curfman McInnes, B Smith, and Hong Zhang. PETSc users manual revision 3.3. *Computer Science Division, Argonne National Laboratory, Argonne, IL*, 2012. → pages 92

[21] Jonas Ballani and Lars Grasedyck. Tree adaptive approximation in the hierarchical tensor format. *Preprint*, 141, 2013. → pages 43

[22] Jonas Ballani, Lars Grasedyck, and Melanie Kluge. Black box approximation of tensors in Hierarchical Tucker format. *Linear Algebra and its Applications*, 438(2):639 – 657, 2013. ISSN 0024-3795. doi: 10.1016/j.laa.2011.08.010. Tensors and Multilinear Algebra. → pages 11, 13

[23] Alvaro Barbero and Suvrit Sra. Modular proximal optimization for multidimensional total-variation regularization. *arXiv preprint arXiv:1411.0589*, 2014. → pages 77

[24] Gilbert Bassett Jr and Roger Koenker. Asymptotic theory of least absolute error regression. *Journal of the American Statistical Association*, 73(363): 618–622, 1978. → pages 6

[25] Heinz H Bauschke and Patrick L Combettes. *Convex analysis and monotone operator theory in Hilbert spaces*. Springer Science & Business Media, 2011. → pages 57

[26] Amir Beck and Marc Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM journal on imaging sciences*, 2 (1):183–202, 2009. → pages 1

[27] James Bennett, Stan Lanning, et al. The Netflix prize. In *Proceedings of KDD cup and workshop*, volume 2007, page 35. New York, NY, USA, 2007. → pages 4

[28] Jean-Pierre Berenger. A perfectly matched layer for the absorption of electromagnetic waves. *Journal of computational physics*, 114(2):185–200, 1994. → pages 96

[29] Jeff Bezanson, Alan Edelman, Stefan Karpinski, and Viral B Shah. Julia: A fresh approach to numerical computing. *arXiv preprint arXiv:1411.1607*, 2014. → pages 94

[30] George Biros and Omar Ghattas. Parallel Lagrange–Newton–Krylov–Schur methods for PDE-constrained optimization. part i: The Krylov–Schur solver. *SIAM Journal on Scientific Computing*, 27(2):687–713, 2005. → pages 96

[31] Åke Björck and Tommy Elfving. Accelerated projection methods for computing pseudoinverse solutions of systems of linear equations. *BIT Numerical Mathematics*, 19(2):145–163, 1979. → pages 107

[32] J.D. Blanchard, J. Tanner, and Ke Wei. Conjugate gradient iterative hard thresholding: Observed noise stability for compressed sensing. *Signal Processing, IEEE Transactions on*, 63(2):528–537, Jan 2015. ISSN 1053-587X. doi: 10.1109/TSP.2014.2379665. → pages 42

[33] Hans Georg Bock. Numerical treatment of inverse problems in chemical reaction kinetics. In *Modelling of chemical reaction systems*, pages 102–125. Springer, 1981. → pages 1

[34] Liliana Borcea. Electrical impedance tomography. *Inverse problems*, 18(6): R99, 2002. → pages 95

[35] Brett Borden. Mathematical problems in radar inverse scattering. *Inverse Problems*, 18(1):R1, 2001. → pages 1

[36] Léon Bottou. Online learning and stochastic approximations. *On-line learning in neural networks*, 17(9):142, 1998. → pages 98

[37] Léon Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer, 2010. → pages 98

[38] Yassine Boubendir, Xavier Antoine, and Christophe Geuzaine. A quasi-optimal non-overlapping domain decomposition algorithm for the Helmholtz equation. *Journal of Computational Physics*, 231(2):262–280, 2012. → pages 105

[39] Petros T Boufounos and Richard G Baraniuk. 1-bit compressive sensing. In *Information Sciences and Systems, 2008. CISS 2008. 42nd Annual Conference on*, pages 16–21. IEEE, 2008. → pages 84

[40] Stephen Boyd, Lin Xiao, and Almir Mutapcic. Subgradient methods. 2003. → pages 6

[41] William L Briggs, Van Emden Henson, and Steve F McCormick. *A multigrid tutorial*. SIAM, 2000. → pages 106

[42] Brian H Brown. Electrical impedance tomography (EIT): a review. *Journal of medical engineering & technology*, 27(3):97–108, 2003. → pages 1

[43] James V Burke. Second order necessary and sufficient conditions for convex composite NDO. *Mathematical Programming*, 38(3):287–302, 1987. → pages 63

[44] James V Burke and Michael C Ferris. A Gauss—Newton method for convex composite optimization. *Mathematical Programming*, 71(2):179–194, 1995. → pages 63

[45] Francesca Cagliari, Barbara Di Fabio, and Claudia Landi. The natural pseudo-distance as a quotient pseudo-metric, and applications. *AMS Acta, Universita di Bologna*, 3499, 2012. → pages 40

[46] Jian-Feng Cai, Stanley Osher, and Zuowei Shen. Convergence of the linearized Bregman iteration for L1-norm minimization. *Mathematics of Computation*, 78(268):2127–2136, 2009. → pages 114

[47] Jian-Feng Cai, Stanley Osher, and Zuowei Shen. Linearized Bregman iterations for compressed sensing. *Mathematics of Computation*, 78(267): 1515–1536, 2009. → pages 114

[48] Jian-Feng Cai, Stanley Osher, and Zuowei Shen. Split Bregman methods and frame based image restoration. *Multiscale modeling & simulation*, 8(2): 337–369, 2009. → pages 74

[49] Jian-Feng Cai, Emmanuel J Candès, and Zuowei Shen. A singular value thresholding algorithm for matrix completion. *SIAM Journal on Optimization*, 20(4):1956–1982, 2010. → pages 2, 5

[50] Tianxi Cai, T Tony Cai, and Anru Zhang. Structured matrix completion with applications to genomic data integration. *arXiv preprint arXiv:1504.01823*, 2015. → pages 5

[51] Henri Calandra, Serge Gratton, Xavier Pinel, and Xavier Vasseur. An improved two-grid preconditioner for the solution of three-dimensional Helmholtz problems in heterogeneous media. *Numerical Linear Algebra with Applications*, 20(4):663–688, 2013. ISSN 1099-1506. doi: 10.1002/nla.1860. → pages 106, 107

[52] Emmanuel Candes and Benjamin Recht. Exact matrix completion via convex optimization. *Communications of the ACM*, 55(6):111–119, 2012. → pages 2

[53] Emmanuel Candes, Mark Rudelson, Terence Tao, and Roman Vershynin. Error correction via linear programming. In *Foundations of Computer Science, 2005. FOCS 2005. 46th Annual IEEE Symposium on*, pages 668–681. IEEE, 2005. → pages 72

[54] Emmanuel Candes, Laurent Demanet, David Donoho, and Lexing Ying. Fast discrete curvelet transforms. *Multiscale Modeling & Simulation*, 5(3): 861–899, 2006. → pages 74

[55] Emmanuel J Candes and David L Donoho. Curvelets: A surprisingly effective nonadaptive representation for objects with edges. Technical report, DTIC Document, 2000. → pages 114

[56] Emmanuel J Candès and David L Donoho. New tight frames of curvelets and optimal representations of objects with piecewise C2 singularities. *Communications on pure and applied mathematics*, 57(2):219–266, 2004. → pages 114

[57] Emmanuel J Candes and Terence Tao. Decoding by linear programming. *IEEE transactions on information theory*, 51(12):4203–4215, 2005. → pages 4, 72

[58] Emmanuel J Candes and Terence Tao. Near-optimal signal recovery from random projections: Universal encoding strategies? *IEEE transactions on information theory*, 52(12):5406–5425, 2006. → pages 4, 72

[59] Emmanuel J Candès, Xiaodong Li, Yi Ma, and John Wright. Robust principal component analysis? *Journal of the ACM (JACM)*, 58(3):11, 2011. → pages 63

[60] Emmanuel J Candes, Yonina C Eldar, Thomas Strohmer, and Vladislav Voroninski. Phase retrieval via matrix completion. *SIAM review*, 57(2): 225–251, 2015. → pages 5

[61] J. Carroll and J. Chang. Analysis of individual differences in multidimensional scaling via an N-way generalization of "Eckart-Young" decomposition. *Psychometrika*, 35:283–319, 1970. ISSN 0033-3123. 10.1007/BF02310791. → pages 12

[62] Khosrow Chadan and Pierre C Sabatier. *Inverse problems in quantum scattering theory*. Springer Science & Business Media, 2012. → pages 1

[63] Xintao Chai, Mengmeng Yang, Philipp Witte, Rongrong Wang, Zhilong Fang, and Felix Herrmann. A linearized Bregman method for compressive waveform inversion. In *SEG Technical Program Expanded Abstracts 2016*, pages 1449–1454. Society of Exploration Geophysicists, 2016. → pages 114

[64] Antonin Chambolle and Thomas Pock. A first-order primal-dual algorithm for convex problems with applications to imaging. *Journal of Mathematical Imaging and Vision*, 40(1):120–145, 2011. → pages 73

[65] Raymond H Chan, Min Tao, and Xiaoming Yuan. Constrained total variation deblurring models and fast algorithms based on alternating direction method of multipliers. *SIAM Journal on imaging Sciences*, 6(1): 680–697, 2013. → pages 77

[66] Stanley H Chan, Ramsin Khoshabeh, Kristofor B Gibson, Philip E Gill, and Truong Q Nguyen. An augmented Lagrangian method for total variation video restoration. *IEEE Transactions on Image Processing*, 20(11): 3097–3111, 2011. → pages 77

[67] Venkat Chandrasekaran, Benjamin Recht, Pablo A Parrilo, and Alan S Willsky. The convex geometry of linear inverse problems. *Foundations of Computational mathematics*, 12(6):805–849, 2012. → pages 57

[68] Guang-Hong Chen, Jie Tang, and Shuai Leng. Prior image constrained compressed sensing (PICCS): a method to accurately reconstruct dynamic CT images from highly undersampled projection data sets. *Medical physics*, 35(2):660–663, 2008. → pages 2

[69] J-B Chen. A 27-point scheme for a 3D frequency-domain scalar wave equation based on an average-derivative method. *Geophysical Prospecting*, 62 (2):258–277, 2014. → pages 93

[70] Zhongying Chen, Dongsheng Cheng, Wei Feng, and Tingting Wu. An optimal 9-point finite difference scheme for the Helmholtz equation with PML. *International Journal of Numerical Analysis & Modeling*, 10(2), 2013. → pages 102

[71] Margaret Cheney, David Isaacson, and Jonathan C Newell. Electrical impedance tomography. *SIAM review*, 41(1):85–101, 1999. → pages 1, 95

[72] WC Chew, JM Jin, and E Michielssen. Complex coordinate stretching as a generalized absorbing boundary condition. *Microwave and Optical Technology Letters*, 15(6):363–369, 1997. → pages 96

[73] Radu Cimpeanu, Anton Martinsson, and Matthias Heil. A parameter-free perfectly matched layer formulation for the finite-element-based solution of the Helmholtz equation. *Journal of Computational Physics*, 296:329–347, 2015. → pages 93

[74] Jon F Claerbout and Francis Muir. Robust modeling with erratic data. *Geophysics*, 38(5):826–844, 1973. → pages 4

[75] Kenneth L Clarkson and David P Woodruff. Sketching for M-estimators: A unified approach to robust regression. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 921–939. Society for Industrial and Applied Mathematics, 2015. → pages 6

[76] Rowan Cockett, Seogi Kang, Lindsey J Heagy, Adam Pidlisecky, and Douglas W Oldenburg. Simpeg: An open source framework for simulation and gradient based parameter estimation in geophysical applications. *Computers & Geosciences*, 85:142–154, 2015. → pages 93

[77] Gary Cohen. *Higher-order numerical methods for transient wave equations*. Springer Science & Business Media, 2013. → pages 106

[78] Ian Craig and John Brown. Inverse problems in astronomy. 1986. → pages 1

[79] Curt Da Silva and F Herrmann. Hierarchical Tucker tensor optimization - applications to 4D seismic data interpolation. In *75th EAGE Conference & Exhibition incorporating SPE EUROPEC 2013*, 2013. → pages iv

[80] Curt Da Silva and Felix Herrmann. A unified 2D/3D software environment for large-scale time-harmonic full-waveform inversion. In *SEG Technical Program Expanded Abstracts 2016*, pages 1169–1173. Society of Exploration Geophysicists, 2016. → pages iv

[81] Curt Da Silva and Felix J. Herrmann. Hierarchical Tucker tensor optimization - applications to tensor completion. In *10th international conference on Sampling Theory and Applications (SampTA 2013)*, pages 384–387, Bremen, Germany, July 2013. → pages iv, 14, 43

[82] Curt Da Silva and Felix J Herrmann. Optimization on the Hierarchical Tucker manifold–applications to tensor completion. *Linear Algebra and its Applications*, 481:131–173, 2015. → pages iv

[83] Curt Da Silva and Felix J. Herrmann. A unified 2D/3D software environment for large scale nonlinear inverse problems. Technical report, University of British Columbia, 2017. → pages iv

[84] L. De Lathauwer, B. De Moor, and J. Vandewalle. A multilinear singular value decomposition. *SIAM Journal on Matrix Analysis and Applications*, 21 (4):1253–1278, 2000. → pages 12, 15

[85] V. De Silva and L.H. Lim. Tensor rank and the ill-posedness of the best low-rank approximation problem. *SIAM Journal on Matrix Analysis and Applications*, 30(3):1084–1127, 2008. → pages 12, 16

[86] Paulus Maria De Zeeuw. Matrix-dependent prolongations and restrictions in a blackbox multigrid solver. *Journal of computational and applied mathematics*, 33(1):1–27, 1990. → pages 106

[87] Laurent Demanet. *Curvelets, Wave Atoms, and Wave Equations*. PhD thesis, California Institute of Technology, 2006. → pages 43

[88] John E Dennis Jr, David M Gay, and Roy E Walsh. An adaptive nonlinear least-squares algorithm. *ACM Transactions on Mathematical Software (TOMS)*, 7(3):348–368, 1981. → pages 63

[89] Pedro Díez. A note on the convergence of the secant method for simple and multiple roots. *Applied Mathematics Letters*, 16(8):1211–1215, 2003. → pages 66

[90] David L Donoho. Compressed sensing. *IEEE Transactions on information theory*, 52(4):1289–1306, 2006. → pages 4, 72

[91] D Drusvyatskiy and C Paquette. Efficiency of minimizing compositions of convex functions and smooth maps. *Optimization Online*, 2016. → pages 63

[92] John Duchi, Shai Shalev-Shwartz, Yoram Singer, and Tushar Chandra. Efficient projections onto the L1-ball for learning in high dimensions. In *Proceedings of the 25th international conference on Machine learning*, pages 272–279. ACM, 2008. → pages 86

[93] M. P. Friedlander E. van den Berg. Spot - a linear-operator toolbox. "http://www.cs.ubc.ca/labs/scl/spot/", Accessed April 1, 2014. → pages 25, 94

[94] Lars Eldén and Berkant Savas. A Newton-Grassmann method for computing the best multilinear rank-(r_1, r_2, r_3) approximation of a tensor. *SIAM Journal on Matrix Analysis and applications*, 31(2):248–271, 2009. → pages 16

[95] Björn Engquist and Lexing Ying. Sweeping preconditioner for the Helmholtz equation: moving perfectly matched layers. *Multiscale Modeling & Simulation*, 9(2):686–710, 2011. → pages 105

[96] Yogi A Erlangga, Cornelis Vuik, and Cornelis Willebrordus Oosterlee. On a class of preconditioners for solving the Helmholtz equation. *Applied Numerical Mathematics*, 50(3):409–425, 2004. → pages 105

[97] Antonio Falcó, Wolfgang Hackbusch, Anthony Nouy, et al. Geometric structures in tensor representations (release 2). 2014. → pages 13

[98] Patrick E Farrell, David A Ham, Simon W Funke, and Marie E Rognes. Automated derivation of the adjoint of high-level transient finite element programs. *SIAM Journal on Scientific Computing*, 35(4):C369–C393, 2013. → pages 92

[99] David J Field. Wavelets, vision and the statistics of natural scenes. *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 357(1760):2527–2542, 1999. → pages 2

[100] Mário AT Figueiredo, Robert D Nowak, and Stephen J Wright. Gradient projection for sparse reconstruction: Application to compressed sensing and other inverse problems. *IEEE Journal of selected topics in signal processing*, 1(4):586–597, 2007. → pages 1

[101] David Chin-Lung Fong and Michael Saunders. LSMR: An iterative algorithm for sparse least-squares problems. *SIAM Journal on Scientific Computing*, 33(5):2950–2971, 2011. → pages 114

[102] Inéz Frerichs. Electrical impedance tomography (EIT) in applications related to lung and ventilation: a review of experimental and clinical activities. *Physiological measurement*, 21(2):R1, 2000. → pages 1

[103] Michael P Friedlander and Mark Schmidt. Hybrid deterministic-stochastic methods for data fitting. *SIAM Journal on Scientific Computing*, 34(3): A1380–A1405, 2012. → pages 98

[104] Martin J Gander and Hui Zhang. Domain decomposition methods for the Helmholtz equation: a numerical investigation. In *Domain Decomposition Methods in Science and Engineering XX*, pages 215–222. Springer, 2013. → pages 105

[105] S. Gandy, B. Recht, and I. Yamada. Tensor completion and low-n-rank tensor recovery via convex optimization. *Inverse Problems*, 27(2):025010, 2011. → pages 12, 14, 39

[106] Euhanna Ghadimi, André Teixeira, Iman Shames, and Mikael Johansson. Optimal parameter selection for the alternating direction method of multipliers (ADMM): quadratic problems. *IEEE Transactions on Automatic Control*, 60(3):644–658, 2015. → pages 63

[107] Gene Golub and Victor Pereyra. Separable nonlinear least squares: the variable projection method and its applications. *Inverse problems*, 19(2):R1, 2003. → pages 63, 65

[108] Dan Gordon and Rachel Gordon. CARP-CG: A robust and efficient parallel solver for linear systems, applied to strongly convection dominated PDEs. *Parallel Computing*, 36(9):495 – 515, 2010. ISSN 0167-8191. doi: 10.1016/j.parco.2010.05.004. → pages 107

[109] L. Grasedyck. Hierarchical singular value decomposition of tensors. *SIAM Journal on Matrix Analysis and Applications*, 31(4):2029–2054, 2010. → pages xii, 13, 15, 17, 19, 32, 36

[110] Lars Grasedyck, Melanie Kluge, and Sebastian Krämer. Alternating directions fitting (ADF) of hierarchical low rank tensors. *Preprint*, 149, 2013. → pages 13

[111] Lars Grasedyck, Daniel Kressner, and Christine Tobler. A literature survey of low-rank tensor approximation techniques. *GAMM-Mitteilungen*, 36(1): 53–78, 2013. → pages 11

[112] Samuel H Gray, John Etgen, Joe Dellinger, and Dan Whitmore. Seismic migration problems and solutions. *Geophysics*, 66(5):1622–1640, 2001. → pages 91

[113] L Grippo, F Lampariello, and S Lucidi. A truncated Newton method with nonmonotone line search for unconstrained optimization. *Journal of Optimization Theory and Applications*, 60(3):401–419, 1989. → pages 99

[114] Antoine Guitton and William W Symes. Robust inversion of seismic data using the Huber norm. *Geophysics*, 68(4):1310–1319, 2003. → pages 7, 60

[115] W. Hackbusch and S. Kühn. A new scheme for the tensor representation. *Journal of Fourier Analysis and Applications*, 15(5):706–722, 2009. → pages 5, 8, 13

[116] Wolfgang Hackbusch. *Tensor spaces and numerical tensor calculus*, volume 42. Springer, 2012. → pages 11

[117] Jutho Haegeman, Tobias J Osborne, and Frank Verstraete. Post-matrix product state methods: To tangent space and beyond. *Physical Review B*, 88 (7):075133, 2013. → pages 13

[118] R.A. Harshman. Foundations of the parafac procedure: models and conditions for an "explanatory" multimodal factor analysis. *UCLA Working Papers in Phonetics*, 16:1–84, 1970. → pages 12

[119] Richard A Harshman. Foundations of the parafac procedure: Models and conditions for an" explanatory" multi-modal factor analysis. 1970. → pages 5

[120] Mark J Harvilla and Richard M Stern. Least squares signal declipping for robust speech recognition. 2014. → pages 79

[121] Trevor Hastie, Saharon Rosset, Robert Tibshirani, and Ji Zhu. The entire regularization path for the support vector machine. *Journal of Machine Learning Research*, 5(Oct):1391–1415, 2004. → pages 63

[122] Frank D Hastings, John B Schneider, and Shira L Broschat. Application of the perfectly matched layer (PML) absorbing boundary condition to elastic wave propagation. *The Journal of the Acoustical Society of America*, 100(5): 3061–3069, 1996. → pages 96

[123] Matthew A Herman and Thomas Strohmer. High-resolution radar via compressed sensing. *IEEE transactions on signal processing*, 57(6): 2275–2284, 2009. → pages 2

[124] Michael A Heroux, Roscoe A Bartlett, Vicki E Howle, Robert J Hoekstra, Jonathan J Hu, Tamara G Kolda, Richard B Lehoucq, Kevin R Long, Roger P Pawlowski, Eric T Phipps, et al. An overview of the trilinos project. *ACM Transactions on Mathematical Software (TOMS)*, 31(3):397–423, 2005. → pages 92

[125] Felix J Herrmann and Gilles Hennenfent. Non-parametric seismic data recovery with curvelet frames. *Geophysical Journal International*, 173(1): 233–248, 2008. → pages 2

[126] Felix J. Herrmann and Xiang Li. Efficient least-squares migration with sparsity promotion. In *EAGE Annual Conference Proceedings*. EAGE, EAGE, 05 2011. → pages 152

[127] Felix J Herrmann and Xiang Li. Efficient least-squares imaging with sparsity promotion and compressive sensing. *Geophysical prospecting*, 60(4):696–712, 2012. → pages 114

[128] Felix J. Herrmann and Bas Peters. Constraints versus penalties for edge-preserving full-waveform inversion. In *SEG Workshop on Where are we heading with FWI; Dallas*, 10 2016. (SEG Workshop, Dallas). → pages 117

[129] Felix J. Herrmann, Deli Wang, Gilles Hennenfent, and Peyman P. Moghaddam. Seismic data processing with curvelets: a multiscale and nonlinear approach. In *SEG Technical Program Expanded Abstracts*, volume 26, pages 2220–2224. SEG, SEG, 2007. doi: 10.1190/1.2792927. → pages 2

[130] Felix J Herrmann, Deli Wang, Gilles Hennenfent, and Peyman P Moghaddam. Curvelet-based seismic data processing: A multiscale and nonlinear approach. *Geophysics*, 73(1):A1–A5, 2007. → pages 2

[131] Felix J Herrmann, Peyman Moghaddam, and Christiaan C Stolk. Sparsity-and continuity-promoting seismic image recovery with curvelet frames. *Applied and Computational Harmonic Analysis*, 24(2):150–173, 2008. → pages 114

[132] FJ Herrmann, N Tu, and E Esser. Fast "online" migration with compressive sensing. *77th Annual International Conference and Exhibition, EAGE*, 2015. → pages 114

[133] Graham J Hicks. Arbitrary source and receiver positioning in finite-difference schemes using Kaiser windowed sinc functions. *Geophysics*, 67(1):156–165, 2002. → pages 100

[134] Frank L Hitchcock. The expression of a tensor or a polyadic as a sum of products. *Studies in Applied Mathematics*, 6(1-4):164–189, 1927. → pages 5

[135] Olav Holberg. Computational aspects of the choice of operator and sampling interval for numerical differentiation in large-scale simulation of wave phenomena. *Geophysical prospecting*, 35(6):629–655, 1987. → pages 91

[136] Sebastian Holtz, Thorsten Rohwedder, and Reinhold Schneider. The alternating linear scheme for tensor optimization in the tensor train format. *SIAM Journal on Scientific Computing*, 34(2):A683–A713, 2012. → pages 13

[137] Sebastian Holtz, Thorsten Rohwedder, and Reinhold Schneider. On manifolds of tensors of fixed TT-rank. *Numerische Mathematik*, 120(4): 701–731, 2012. ISSN 0029-599X. doi: 10.1007/s00211-011-0419-7. → pages 13

[138] L Hu, L Huang, and ZR Lu. Crack identification of beam structures using homotopy continuation algorithm. *Inverse Problems in Science and Engineering*, 25(2):169–187, 2017. → pages 1

[139] Yaohua Hu, Chong Li, and Xiaoqi Yang. On convergence rates of linearized proximal algorithms for convex composite optimization with applications. *SIAM Journal on Optimization*, 26(2):1207–1235, 2016. → pages 64

[140] Bo Huang, Cun Mu, Donald Goldfarb, and John Wright. Provable low-rank tensor recovery. 2014. → pages 12

[141] Laurent Jacques, Jason N Laska, Petros T Boufounos, and Richard G Baraniuk. Robust 1-bit compressive sensing via binary stable embeddings of sparse vectors. *IEEE Transactions on Information Theory*, 59(4):2082–2102, 2013. → pages 84, 86, 87

[142] Churl-Hyun Jo, Changsoo Shin, and Jung Hee Suh. An optimal 9-point, finite-difference, frequency-space, 2-D scalar wave extrapolator. *Geophysics*, 61(2):529–537, 1996. → pages 102

[143] Ian Jolliffe. *Principal component analysis*. Wiley Online Library, 2002. → pages 3

[144] Anatoli Juditsky, Arkadi Nemirovski, et al. First order methods for nonsmooth convex large-scale optimization, i: general purpose methods. *Optimization for Machine Learning*, pages 121–148. → pages 64

[145] Stefan Kaczmarz. Angenäherte auflösung von systemen linearer gleichungen. *Bulletin International de l'Académie Polonaise des Sciences et des Lettres*, 35:355–357, 1937. → pages 107

[146] Hiroo Kanamori. Quantification of earthquakes. *Nature*, 271:411–414, 1978. → pages 91

[147] Hamed Karimi, Julie Nutini, and Mark Schmidt. Linear convergence of gradient and proximal-gradient methods under the Polyak-Lojasiewicz condition. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 795–811. Springer, 2016. → pages 8, 55, 58, 68, 71

[148] Hiroyuki Kasai and Bamdev Mishra. Riemannian preconditioning for tensor completion. *arXiv preprint arXiv:1506.02159*, 2015. → pages 23

[149] Hiroyuki Kasai and Bamdev Mishra. Low-rank tensor completion: a Riemannian manifold preconditioning approach. In *International Conference on Machine Learning*, pages 1012–1021, 2016. → pages 23

[150] Linda Kaufman. A variable projection method for solving separable nonlinear least squares problems. *BIT Numerical Mathematics*, 15(1):49–57, 1975. ISSN 1572-9125. doi: 10.1007/BF01932995. → pages 65

[151] Steven M Kay. Statistical signal processing. *Estimation Theory*, 1, 1993. → pages 63

[152] Boris N. Khoromskij. Tensors-structured numerical methods in scientific computing: Survey on recent advances. *Chemometrics and Intelligent Laboratory Systems*, 110(1):1 – 19, 2012. ISSN 0169-7439. doi: 10.1016/j.chemolab.2011.09.001. → pages 11

[153] Srđan Kitić, Nancy Bertin, and Rémi Gribonval. Audio declipping by cosparse hard thresholding. In *iTwist-2nd international-Traveling Workshop on Interactions between Sparse models and Technology*, 2014. → pages 79, 80

[154] Srđan Kitić, Nancy Bertin, and Rémi Gribonval. Sparsity and cosparsity for audio declipping: a flexible non-convex approach. In *International Conference on Latent Variable Analysis and Signal Separation*, pages 243–250. Springer, 2015. → pages 79, 80

[155] Srdjan Kitic, Laurent Jacques, Nilesh Madhu, Michael Peter Hopwood, Ann Spriet, and Christophe De Vleeschouwer. Consistent iterative hard thresholding for signal declipping. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 5939–5943. IEEE, 2013. → pages 79

[156] Matthew G Knepley, Richard F Katz, and Barry Smith. Developing a geodynamics simulator with PETSc. In *Numerical Solution of Partial Differential Equations on Parallel Computers*, pages 413–438. Springer, 2006. → pages 92

[157] Tamara G Kolda and Brett W Bader. Tensor decompositions and applications. *SIAM review*, 51(3):455–500, 2009. → pages 11, 12, 15

[158] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8), 2009. → pages 5

[159] N. Kreimer and M.D. Sacchi. Tensor completion via nuclear norm minimization for 5D seismic data reconstruction. In *SEG Technical Program Expanded Abstracts 2012*, pages 1–5. Society of Exploration Geophysicists, 2012. → pages 12

[160] Nadia Kreimer and Mauricio D Sacchi. A tensor higher-order singular value decomposition for prestack seismic data noise reduction and interpolation. *Geophysics*, 77(3):V113–V122, 2012. → pages 12

[161] D. Kressner, M. Steinlechner, and B. Vandereycken. Low-rank tensor completion by Riemannian optimization. Technical Report 2, 2014. → pages 13, 14, 28, 37, 39, 41, 42, 44

[162] Daniel Kressner and Christine Tobler. Algorithm 941: Htucker—a Matlab toolbox for tensors in Hierarchical Tucker format. *ACM Trans. Math. Softw.*, 40(3):22:1–22:22, April 2014. ISSN 0098-3500. doi: 10.1145/2538688. → pages 13, 41

[163] Rajiv Kumar, Hassan Mansour, Felix J Herrmann, and Aleksandr Y Aravkin. Reconstruction of seismic wavefields via low-rank matrix factorization in the hierarchical-separable matrix representation. In *SEG Technical Program Expanded Abstracts 2013*, pages 3628–3633. Society of Exploration Geophysicists, 2013. → pages 54

[164] Rajiv Kumar, Curt Da Silva, Okan Akalin, Aleksandr Y Aravkin, Hassan Mansour, Benjamin Recht, and Felix J Herrmann. Efficient matrix completion for seismic data reconstruction. *Geophysics*, 80(5):V97–V114, 2015. → pages 5

[165] Rafael Lago and Felix J. Herrmann. Towards a robust geometric multigrid scheme for Helmholtz equation. Technical Report TR-EOAS-2015-3, UBC, 01 2015. → pages 106, 107

[166] Michael Lange, Navjot Kukreja, Mathias Louboutin, Fabio Luporini, Felippe Vieira, Vincenzo Pandolfo, Paulius Velesko, Paulius Kazakas, and Gerard Gorman. Devito: towards a generic finite difference DSL using symbolic python. *arXiv preprint arXiv:1609.03361*, 2016. → pages 92, 122

[167] Adrian S Lewis and Stephen J Wright. A proximal method for composite minimization. *Mathematical Programming*, pages 1–46, 2008. → pages 63

[168] Adrian S Lewis and Stephen J Wright. A proximal method for composite minimization. *Mathematical Programming*, 158(1-2):501–546, 2016. → pages 64

[169] Chong Li and Xinghua Wang. On convergence of the Gauss-Newton method for convex composite optimization. *Mathematical programming*, 91(2): 349–356, 2002. → pages 63

[170] Yunyue Elita Li and Laurent Demanet. Full-waveform inversion with extrapolated low-frequency data. *GEOPHYSICS*, 81(6):R339–R348, 2016. doi: 10.1190/geo2016-0038.1. → pages 113

[171] Dong C Liu and Jorge Nocedal. On the limited memory BFGS method for large scale optimization. *Mathematical programming*, 45(1):503–528, 1989. → pages 99

[172] Fei Liu and Lexing Ying. Recursive sweeping preconditioner for the 3D Helmholtz equation. *arXiv preprint arXiv:1502.07266*, 2015. → pages 105

[173] Fei Liu and Lexing Ying. Additive sweeping preconditioner for the Helmholtz equation. *Multiscale Modeling & Simulation*, 14(2):799–822, 2016. → pages 105

[174] Christian Lubich. *From quantum to classical molecular dynamics: reduced models and numerical analysis*. European Mathematical Society, 2008. → pages 13

[175] Christian Lubich, Thorsten Rohwedder, Reinhold Schneider, and Bart Vandereycken. Dynamical Approximation by Hierarchical Tucker and Tensor-Train Tensors. *SIAM Journal on Matrix Analysis and Applications*, 34(2):470–494, April 2013. → pages 13

[176] David G Luenberger. *Introduction to linear and nonlinear programming*, volume 28. Addison-Wesley Reading, MA, 1973. → pages 6

[177] Michael Lustig, David Donoho, and John M Pauly. Sparse MRI: The application of compressed sensing for rapid MR imaging. *Magnetic resonance in medicine*, 58(6):1182–1195, 2007. → pages 2

[178] Michael Lustig, David L Donoho, Juan M Santos, and John M Pauly. Compressed sensing MRI. *IEEE signal processing magazine*, 25(2):72–82, 2008. → pages 72

[179] Kaj Madsen and Hans Bruun Nielsen. Finite alogorithms for robust linear regression. *BIT Numerical Mathematics*, 30(4):682–699, 1990. → pages 60

[180] Olvi L Mangasarian and David R. Musicant. Robust linear and support vector regression. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(9):950–955, 2000. → pages 60

[181] Robert Marks. *Introduction to Shannon sampling and interpolation theory*. Springer Science & Business Media, 2012. → pages 2

[182] Ludovic Métivier and Romain Brossier. The seiscope optimization toolbox: a large-scale nonlinear optimization library based on reverse communication. *Geophysics*, 81(2):F11–F25, 2016. → pages 92

[183] B. Mishra and R. Sepulchre. R3MC: A Riemannian three-factor algorithm for low-rank matrix completion. In *53rd IEEE Conference on Decision and Control*, 2014. → pages 18, 23

[184] Bamdev Mishra, Gilles Meyer, Francis Bach, and Rodolphe Sepulchre. Low-rank optimization with trace norm penalty. *SIAM Journal on Optimization*, 23(4):2124–2149, 2013. → pages 42

[185] Cun Mu, Bo Huang, John Wright, and Donald Goldfarb. Square deal: Lower bounds and improved relaxations for tensor recovery. *arXiv preprint arXiv:1307.5870*, 2013. → pages 13

[186] Sangnam Nam, Mike E Davies, Michael Elad, and Rémi Gribonval. The cosparse analysis model and algorithms. *Applied and Computational Harmonic Analysis*, 34(1):30–56, 2013. → pages 72, 74

[187] Balas Kausik Natarajan. Sparse approximate solutions to linear systems. *SIAM journal on computing*, 24(2):227–234, 1995. → pages 81

[188] Frank Natterer. Imaging and inverse problems of partial differential equations. Technical report, Institut fur Numerische und Angewandte Mathematik, 2006. → pages 95

[189] Tamas Nemeth, Chengjun Wu, and Gerard T Schuster. Least-squares migration of incomplete reflection data. *Geophysics*, 64(1):208–221, 1999. → pages 3

[190] Yurii Nesterov. *Introductory lectures on convex optimization: A basic course*, volume 87. Springer Science & Business Media, 2013. → pages 89

[191] Stéphane Operto, Jean Virieux, Patrick Amestoy, Jean-Yves L'Excellent, Luc Giraud, and Hafedh Ben Hadj Ali. 3D finite-difference frequency-domain modeling of visco-acoustic wave propagation using a massively parallel direct solver: A feasibility study. *GEOPHYSICS*, 72(5):SM195–SM211, 2007. doi: 10.1190/1.2759835. → pages 8, 93, 102, 106, 107, 108, 109, 128

[192] Ivan V Oseledets. Tensor-train decomposition. *SIAM Journal on Scientific Computing*, 33(5):2295–2317, 2011. → pages 13

[193] Ivan V Oseledets and SV Dolgov. Solution of linear systems and matrix inversion in the TT-format. *SIAM Journal on Scientific Computing*, 34(5): A2718–A2739, 2012. → pages 13

[194] Ivan V Oseledets and Eugene E Tyrtyshnikov. Breaking the curse of dimensionality, or how to use SVD in many dimensions. *SIAM Journal on Scientific Computing*, 31(5):3744–3759, 2009. → pages 13

[195] Stanley Osher, Yu Mao, Bin Dong, and Wotao Yin. Fast linearized Bregman iteration for compressive sensing and sparse denoising. *arXiv preprint arXiv:1104.0262*, 2011. → pages 114

[196] Samet Oymak, Amin Jalali, Maryam Fazel, Yonina C Eldar, and Babak Hassibi. Simultaneously structured models with application to sparse and low-rank matrices. *arXiv preprint arXiv:1212.3753*, 2012. → pages 12

[197] Anthony D. Padula, Shannon D. Scott, and William W. Symes. A software framework for abstract expression of coordinate-free linear algebra and optimization algorithms. *ACM Trans. Math. Softw.*, 36(2):8:1–8:36, April 2009. ISSN 0098-3500. doi: 10.1145/1499096.1499097. → pages 92

[198] Christopher C Paige and Michael A Saunders. LSQR: An algorithm for sparse linear equations and sparse least squares. 1982. → pages 74

[199] David A Patterson. *Computer architecture: a quantitative approach*. Elsevier, 2011. → pages 93

[200] Bas Peters and Felix J. Herrmann. Constraints versus penalties for edge-preserving full-waveform inversion. Revision 1 submitted to The Leading Edge on November 13, 2016., 2016. → pages 98

[201] Gabriel Peyré. The numerical tours of signal processing. *Computing in Science & Engineering*, 13(4):94–97, 2011. → pages 80

[202] Yaniv Plan and Roman Vershynin. One-bit compressed sensing by linear programming. *Communications on Pure and Applied Mathematics*, 66(8): 1275–1297, 2013. → pages 84, 85

[203] Chayne Planiden and Xianfu Wang. Strongly convex functions, moreau envelopes, and the generic nature of convex functions with strong minimizers. *SIAM Journal on Optimization*, 26(2):1341–1364, 2016. → pages 57, 68

[204] R-E Plessix. A review of the adjoint-state method for computing the gradient of a functional with geophysical applications. *Geophysical Journal International*, 167(2):495–503, 2006. → pages 152

[205] R-E Plessix. A Helmholtz iterative solver for 3D seismic-imaging problems. *Geophysics*, 72(5):SM185–SM194, 2007. → pages 105

[206] David Pollard. Asymptotics for least absolute deviation regression estimators. *Econometric Theory*, 7(02):186–199, 1991. → pages 6

[207] Jack Poulson, Bjorn Engquist, Siwei Li, and Lexing Ying. A parallel sweeping preconditioner for heterogeneous 3D Helmholtz equations. *SIAM Journal on Scientific Computing*, 35(3):C194–C212, 2013. → pages 105

[208] Ernesto E Prudencio, Richard Byrd, and Xiao-Chuan Cai. Parallel full space SQP Lagrange–Newton–Krylov–Schwarz algorithms for pde-constrained optimization problems. *SIAM Journal on Scientific Computing*, 27(4): 1305–1328, 2006. → pages 96

[209] Shie Qian and Dapang Chen. Discrete gabor transform. *IEEE transactions on signal processing*, 41(7):2429–2438, 1993. → pages 80

[210] Holger Rauhut, Reinhold Schneider, and Zeljka Stojanac. Tensor completion in hierarchical tensor representations. *arXiv preprint arXiv:1404.3905*, 2014. → pages 13

[211] Holger Rauhut, Reinhold Schneider, and Željka Stojanac. Tensor completion in hierarchical tensor representations. In *Compressed Sensing and its Applications*, pages 419–450. Springer, 2015. → pages 125

[212] Holger Rauhut, Reinhold Schneider, and Željka Stojanac. Low rank tensor recovery via iterative hard thresholding. *Linear Algebra and its Applications*, 523:220–262, 2017. → pages 125

[213] Benjamin Recht. A simpler approach to matrix completion. *Journal of Machine Learning Research*, 12(Dec):3413–3430, 2011. → pages 5

[214] Benjamin Recht and Christopher Ré. Parallel stochastic gradient algorithms for large-scale matrix completion. *Mathematical Programming Computation*, 5(2):201–226, 2013. → pages 5

[215] CD Riyanti, A Kononov, Yogi A Erlangga, Cornelis Vuik, Cornelis W Oosterlee, R-E Plessix, and Wim A Mulder. A parallel multigrid-based preconditioner for the 3D heterogeneous high-frequency Helmholtz equation. *Journal of Computational physics*, 224(1):431–448, 2007. → pages 105

[216] R Tyrrell Rockafellar and Roger J-B Wets. *Variational analysis*, volume 317. Springer Science & Business Media, 2009. → pages 57

[217] Ralph Tyrrell Rockafeller. *Convex Analysis*. Princeton University Press, 1970. → pages 70

[218] Farbod Roosta-Khorasani, Kees Van Den Doel, and Uri Ascher. Data completion and stochastic algorithms for pde inversion problems with many measurements. *Electron. Trans. Numer. Anal*, 42:177–196, 2014. → pages 2

[219] Leonid I Rudin, Stanley Osher, and Emad Fatemi. Nonlinear total variation based noise removal algorithms. *Physica D: Nonlinear Phenomena*, 60(1-4): 259–268, 1992. → pages 76

[220] Lars Ruthotto, Eran Treister, and Eldad Haber. jInv–a flexible Julia package for PDE parameter estimation. *arXiv preprint arXiv:1606.07399*, 2016. → pages 93

[221] Youcef Saad. A flexible inner-outer preconditioned GMRES algorithm. *SIAM Journal on Scientific Computing*, 14(2):461–469, 1993. → pages 98, 107

[222] Youcef Saad and Martin H Schultz. GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal on scientific and statistical computing*, 7(3):856–869, 1986. → pages 107

[223] Mark W Schmidt, Ewout Van Den Berg, Michael P Friedlander, and Kevin P Murphy. Optimizing costly functions with simple constraints: A limited-memory projected quasi-Newton algorithm. In *AISTATS*, volume 5, pages 456–463, 2009. → pages 80, 111, 116

[224] Reinhold Schneider and André Uschmajew. Approximation rates for the hierarchical tensor format in periodic Sobolev spaces. *Journal of Complexity*, 30(2):56–71, 2014. → pages 43, 125

[225] Reinhold Schneider and André Uschmajew. Convergence results for projected line-search methods on varieties of low-rank matrices via lojasiewicz inequality. *arXiv preprint arXiv:1402.5284*, 2014. → pages 41

[226] Claude Elwood Shannon. Communication in the presence of noise. *Proceedings of the IRE*, 37(1):10–21, 1949. → pages 2

[227] John Shawe-Taylor and Shiliang Sun. A review of optimization methodologies in support vector machines. *Neurocomputing*, 74(17): 3609–3618, 2011. → pages 63

[228] Peter M Shearer. *Introduction to seismology*. Cambridge University Press, 2009. → pages 122

[229] Chao Shen, Tsung-Hui Chang, Kun-Yu Wang, Zhengding Qiu, and Chong-Yung Chi. Distributed robust multicell coordinated beamforming with imperfect csi: An ADMM approach. *IEEE Transactions on signal processing*, 60(6):2988–3003, 2012. → pages 63

[230] Z J Shi and J Shen. New inexact line search method for unconstrained optimization. *Journal of Optimization Theory and Applications*, 127(2): 425–446, November 2005. → pages 33

[231] Zhen-Jun Shi. Convergence of line search methods for unconstrained optimization. *Applied Mathematics and Computation*, 157(2):393–405, 2004. → pages 33

[232] Naum Zuselevich Shor. *Minimization methods for non-differentiable functions*, volume 3. Springer Science & Business Media, 2012. → pages 6

[233] Kai Siedenburg, Matthieu Kowalski, and Monika Dorfler. Audio declipping with social sparsity. In *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*, pages 1577–1581. IEEE, 2014. → pages 79

[234] M. Signoretto, R. Van de Plas, B. De Moor, and J. AK Suykens. Tensor versus matrix completion: a comparison with application to spectral data. *Signal Processing Letters, IEEE*, 18(7):403–406, 2011. → pages 12

[235] Arnold Sommerfeld. *Partial differential equations in physics*, volume 1. Academic press, 1949. → pages 95

[236] Zhong-Min Song and Paul R Williamson. Frequency-domain acoustic-wave modeling and inversion of crosshole data: Part i 2.5D modeling method. *Geophysics*, 60(3):784–795, 1995. → pages 104, 105

[237] Christiaan C Stolk. A rapidly converging domain decomposition method for the Helmholtz equation. *Journal of Computational Physics*, 241:240–252, 2013. → pages 105

[238] Christiaan C Stolk. An improved sweeping domain decomposition preconditioner for the Helmholtz equation. *Advances in Computational Mathematics*, pages 1–32, 2016. → pages 105

[239] Christiaan C Stolk, Mostak Ahmed, and Samir Kumar Bhowmik. A multigrid method for the Helmholtz equation with optimized coarse grid corrections. *SIAM Journal on Scientific Computing*, 36(6):A2819–A2841, 2014. → pages 105

[240] Dong Sun and William W Symes. Waveform inversion via nonlinear differential semblance optimization. In *75th EAGE Conference & Exhibition-Workshops*, 2013. → pages 96

[241] William W. Symes, Dong Sun, and Marco Enriquez. From modelling to inversion: designing a well-adapted simulator. *Geophysical Prospecting*, 59 (5):814–833, 2011. ISSN 1365-2478. doi: 10.1111/j.1365-2478.2011.00977.x. → pages 92

[242] Min Tao and Junfeng Yang. Alternating direction algorithms for total variation deconvolution in image reconstruction. *Optimization Online*, 2009. → pages 77, 78

[243] Albert Tarantola. *Inverse problem theory and methods for model parameter estimation*. SIAM, 2005. → pages 1

[244] Albert Tarantola and Bernard Valette. Generalized nonlinear inverse problems solved using the least squares criterion. *Reviews of Geophysics*, 20 (2):219–232, 1982. → pages 1

[245] Howard L Taylor, Stephen C Banks, and John F McCoy. Deconvolution with the L1 norm. *Geophysics*, 44(1):39–52, 1979. → pages 4

[246] Fons Ten Kroode, Steffen Bergler, Cees Corsten, Jan Willem de Maag, Floris Strijbos, and Henk Tijhof. Broadband seismic data—the importance of low frequencies. *Geophysics*, 78(2):WA3–WA14, 2013. → pages 91

[247] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288, 1996. → pages 6

[248] C. Tobler. *Low Rank Tensor Methods for Linear Systems and Eigenvalue Problems*. PhD thesis, ETH Zürich, 2012. → pages 36

[249] Ning Tu and Felix J Herrmann. Fast imaging with surface-related multiples by sparse inversion. *Geophysical Journal International*, 201(1):304–317, 2015. → pages 114

[250] Ledyard R Tucker. Some mathematical notes on three-mode factor analysis. *Psychometrika*, 31(3):279–311, 1966. → pages 5

[251] Eli Turkel, Dan Gordon, Rachel Gordon, and Semyon Tsynkov. Compact 2D and 3D sixth order schemes for the Helmholtz equation with variable wave number. *Journal of Computational Physics*, 232(1):272–287, 2013. → pages 93

[252] A. Uschmajew. Local convergence of the alternating least squares algorithm for canonical tensor approximation. *SIAM Journal on Matrix Analysis and Applications*, 33(2):639–652, 2012. → pages 12

[253] A. Uschmajew and B. Vandereycken. The geometry of algorithms using hierarchical tensors. *Linear Algebra and its Applications*, 439(1):133–166, July 2013. → pages 6, 8, 11, 13, 14, 15, 16, 18, 19, 20, 21, 24, 37, 40, 124

[254] A. Uschmajew and B. Vandereycken. Line-search methods and rank increase on low-rank matrix varieties. In *Proceedings of the 2014 International Symposium on Nonlinear Theory and its Applications*, 2014. → pages 42

[255] E. van den Berg and M. P. Friedlander. Probing the pareto frontier for basis pursuit solutions. *SIAM Journal on Scientific Computing*, 31(2):890–912, 2008. doi: 10.1137/080714488. → pages 9, 64

[256] Tristan van Leeuwen. A parallel matrix-free framework for frequency-domain seismic modelling, imaging and inversion in matlab. Technical report, University of British Columbia, 2012. → pages 92

[257] Tristan van Leeuwen and Felix J. Herrmann. 3D frequency-domain seismic inversion with controlled sloppiness. *SIAM Journal on Scientific Computing*, 36(5):S192–S217, 10 2014. doi: 10.1137/130918629. (SISC). → pages 98, 103, 118

[258] Tristan van Leeuwen, Dan Gordon, Rachel Gordon, and Felix J Herrmann. Preconditioning the Helmholtz equation via row-projections. In *74th EAGE Conference and Exhibition incorporating EUROPEC 2012*, 2012. → pages 105

[259] Tristan van Leeuwen, Felix J. Herrmann, and Bas Peters. A new take on FWI: wavefield reconstruction inversion. In *EAGE Annual Conference Proceedings*, 06 2014. doi: 10.3997/2214-4609.20140703. → pages 104

[260] Bart Vandereycken. Low-rank matrix completion by Riemannian optimization. *SIAM Journal on Optimization*, 23(2):1214–1236, 2013. → pages 5

[261] Jean Virieux and Stéphane Operto. An overview of full-waveform inversion in exploration geophysics. *Geophysics*, 74(6):WCC1–WCC26, 2009. → pages 1, 91, 111

[262] Curtis R Vogel. *Computational methods for inverse problems*. SIAM, 2002. → pages 1

[263] Rongrong Wang and Felix Herrmann. Frequency down extrapolation with TV norm minimization. In *SEG Technical Program Expanded Abstracts 2016*, pages 1380–1384. Society of Exploration Geophysicists, 2016. → pages 113

[264] Edmund Taylor Whittaker. XVIII.—on the functions which are represented by the expansions of the interpolation-theory. *Proceedings of the Royal Society of Edinburgh*, 35:181–194, 1915. → pages 2

[265] Robert S Womersley. Local properties of algorithms for minimizing nonsmooth composite functions. *Mathematical Programming*, 32(1):69–89, 1985. → pages 63

[266] John Wright, Arvind Ganesh, Shankar Rao, Yigang Peng, and Yi Ma. Robust principal component analysis: Exact recovery of corrupted low-rank matrices via convex optimization. In *Advances in neural information processing systems*, pages 2080–2088, 2009. → pages 3

[267] Huan Xu, Constantine Caramanis, and Shie Mannor. Robustness and regularization of support vector machines. *Journal of Machine Learning Research*, 10(Jul):1485–1510, 2009. → pages 63

[268] Yangyang Xu and Wotao Yin. A block coordinate descent method for regularized multiconvex optimization with applications to nonnegative tensor factorization and completion. *SIAM Journal on Imaging Sciences*, 6(3): 1758–1789, 2013. → pages 12

[269] Jiyan Yang, Xiangrui Meng, and Michael Mahoney. Quantile regression for large-scale applications. In *Proceedings of The 30th International Conference on Machine Learning*, pages 881–887, 2013. → pages 6

[270] Yi Yang, Jianwei Ma, and Stanley Osher. Seismic data reconstruction via matrix completion. *UCLA CAM Report*, pages 12–14, 2012. → pages 5

[271] Tong Zhang. Solving large scale linear prediction problems using stochastic gradient descent algorithms. In *Proceedings of the twenty-first international conference on Machine learning*, page 116. ACM, 2004. → pages 7

[272] Ciyou Zhu, Richard H Byrd, Peihuang Lu, and Jorge Nocedal. Algorithm 778: LBFGS-B: Fortran subroutines for large-scale bound-constrained optimization. *ACM Transactions on Mathematical Software (TOMS)*, 23(4): 550–560, 1997. → pages 99

Appendices

Appendix A

A 'User Friendly' Guide to Basic Inverse Problems

For our framework, not only do we want to solve (5.2) directly, but allow researchers to explore other subproblems associated to the primary problem, such as the linearized problem [126] and the full-Newton trust region subproblem. As such, we are not only interested in the objective function and gradient, but also other intermediate quantities based on differentiating the state equation $H(m)u(m) = q$. A standard approach to deriving these quantities is the adjoint-state approach, described for instance in [204], but the results we outline below are slightly more elementary and do not make use of Lagrange multipliers.

Rather than focusing on differentiating the expressions in (5.2) directly, we find it useful to consider **directional derivatives** various quantities and their relationships to the gradient. That is to say, for a sufficiently smooth function $f(m)$ that can be scalar, vector, or matrix-valued, the directional dervative in the direction δm, denoted $Df(m)[\delta m]$, is a linear function of δm that satisfies

$$Df(m)[\delta m] = \lim_{t \to 0} \frac{f(m + t\delta m) - f(m)}{t}.$$

The most important part of the directional derivative, for the purposes of the following calculations, is that $Df(m)[\delta m]$ is the same mathematical object as $f(m)$, i.e., if $f(m)$ is a matrix, so is $Df(m)[\delta m]$.

For a given inner product $\langle \cdot, \cdot \rangle$, the gradient of $f(m)$, denoted $\nabla f(m)$, is the unique vector that satisfies

$$\langle \nabla f(m), \delta m \rangle = Df(m)[\delta m] \quad \forall \delta m$$

If we are not terribly worried about specifying the exact vectors spaces in which these objects live and treat them as we'd expect them to behave (i.e., satisfying product, chain rules, commuting with matrix transposes, complex conjugation, linear operators, etc.), the resulting derivations become much more manageable.

Starting from the baseline expression for the misfit $f(m)$ and differentiating, using the chain rule, we have that

$$\begin{aligned} f(m) &= \phi(P_r H(m)^{-1} q, d) = \phi(P_r u(m), d) \\ Df(m)[\delta m] &= D\phi(r(m), d)[Dr(m)[\delta m]] \\ r(m) &= P_r u(m) \\ Dr(m)[\delta m] &= P_r Du(m)[\delta m] \end{aligned}$$

In order to determine the expression for $Du(m)[\delta m]$, we differentiate the state equation,

$$H(m)u(m) = q,$$

in the direction δm using the product rule to obtain

$$\begin{aligned} DH(m)[\delta m]u(m) + H(m)Du(m)[\delta m] &= 0 \\ Du(m)[\delta m] &= H(m)^{-1}(-DH(m)[\delta m]u(m)). \end{aligned} \tag{A.1}$$

For the forward modelling operator $F(m)$, $DF(m)[\delta m]$ is the Jacobian or so-called linearized born-modelling operator in geophysical parlance. Since, for any linear operator A, its transpose satisfies

$$\langle Ax, y\rangle = \langle x, A^T y\rangle$$

for any appropriately sized vectors x, y, in order to determine the transpose of $Dr(m)[\delta m]$, we merely take the inner product with an arbitrary vector y, "isolate" the vector δm on one side of the inner product. The other side is the expression for the adjoint operator. In this case, we have that

$$\begin{aligned} \langle DF(m)[\delta m], y\rangle &= \langle P_r H(m)^{-1}(-DH(m)[\delta m]u(m)), y\rangle \\ &= \langle H(m)^{-1}(-DH(m)[\delta m]u(m)), P_r^T y\rangle \\ &= \langle DH(m)[\delta m]u(m), -H(m)^{-H} P_r^T y\rangle \\ &= \langle T\delta m, -H(m)^{-H} P_r^T y\rangle \\ &= \langle \delta m, T^*(-H(m)^{-H} P_r^T y)\rangle \end{aligned}$$

In order to completely specify the adjoint of $DF(m)[\delta m]$, we need to specify the adjoint of $T\delta m = DH(m)[\delta m]u(m)$ acting on a vector. This expression is particular to the form of the PDE with which we're working. For instance, discretizing the constant-density acoustic Helmholtz equation with finite differences results in

$$\nabla^2 u(x) + \omega^2 m(x)u(x) = q(x)$$

with particular matrices L, A discretizing the Laplacian and identity operators, respectively, yields the linear system

$$Lu + \omega^2 A\text{diag}(m)u = q.$$

Therefore, we have the expression

$$\begin{aligned} T\delta m &:= DH(m)[\delta m]u(m) \\ &= \omega^2 A\text{diag}(\delta m)u(m) \\ &= \omega^2 A\text{diag}(u(m))\delta m, \end{aligned} \tag{A.2}$$

whose adjoint is clearly

$$T^* = \omega^2 \text{diag}(\overline{u(m)})A^H \tag{A.3}$$

with directional derivative

$$DT^*[\delta m, \delta u] = \omega^2 \text{diag}(\overline{\delta u})A^H. \tag{A.4}$$

Our final expression for $DF(m)[\cdot]^* y$ is therefore

$$DF(m)[\cdot]^* y = \omega^2 \text{diag}(\overline{u(m)})A^H(-H(m)^{-H}P_r^T y)$$

Setting $y = \nabla\phi(P_r u(m))$, $v(m) = -H(m)^{-H}P_r^T y$ yields the familiar expression for the gradient of $f(m)$

$$\nabla f(m) = \omega^2 \text{diag}(\overline{u(m)})A^H H(m)^{-H}P_r^T(-\nabla\phi(u(m)))$$

This sort of methodology can be used to symbolically differentiate more complicated expressions as well as compute higher order derivatives of $f(m)$. Let us write $\nabla f(m)$ more abstractly as

$$\nabla f(m) = T(m, u)^* v(m),$$

which will allow us to compute the Hessian as

$$\nabla^2 f(m)\delta m = DT(m, u)^*[\delta m, Du[\delta m]]v(m) + T(m, u)^* Dv(m)[\delta m].$$

Here, $Du[\delta m]$ is given in (A.1) and $DT(m, u)^*[\delta m, \delta u]$ is given in (A.4), for the Helmholtz equation. We compute $Dv(m)[\delta m]$ by differentiating $v(m)$ as

$$\begin{aligned} Dv(m)[\delta m] &= H(m)^{-H}DH(m)[\delta m]^H v(m) - H(m)^{-H}P_r^T(\nabla^2\phi[P_r Du(m)[\delta m]]) \\ &= H(m)^{-H}(DH(m)[\delta m]^H v(m) - P_r^T(\nabla^2\phi[P_r Du(m)[\delta m]])) \end{aligned}$$

which completes our derivation for the Hessian-vector product.

A.1 Derivative expressions for Waveform Reconstruction Inversion

From (5.5), we have that the augmented wavefield $u(m)$ solves the least-squares system

$$\min_u \left\| \begin{bmatrix} P_r \\ \lambda H(m) \end{bmatrix} u - \begin{bmatrix} d \\ \lambda q \end{bmatrix} \right\|_2^2,$$

i.e., $u(m)$ solves the normal equations

$$(P_r^T P_r + \lambda^2 H(m)^H H(m))u(m) = P_r^T d + \lambda^2 H(m)^H q.$$

For the objective $\phi(m,u) = \frac{1}{2}\|P_r u - d\|_2^2 + \frac{\lambda^2}{2}\|H(m)u - q\|_2^2$, the corresponding WRI objective is $f(m) = \phi(m, u(m))$. Owing to the variable projection structure of this objective, the expression for $\nabla_m f(m)$, by [14], is

$$\begin{aligned}\nabla_m f(m) &= \nabla_m \phi(m, u(m)) \\ &= \lambda^2 T(m, u(m))^* (H(m)u(m) - q),\end{aligned}$$

which is identical to the original adjoint-state formulation, except evaluated at the wavefield $u(m)$.

The Hessian-vector product is therefore

$$\begin{aligned}\nabla_m^2 f(m) =& DT(m, u(m))[\delta m, Du(m)[\delta m]]^* (H(m)u(m) - q) + \\ & T(m, u(m))^* (DH(m)[\delta m]u(m) + H(m)Du(m)[\delta m]).\end{aligned}$$

As previously, the expressions for $DH(m)[\delta m]$ and $DT(m,u)[\delta m, \delta u]^*$ are implementation specific. It remains to derive an explicit expression for $Du(m)[\delta m]$ below.

Let $G(m) = (P_r^T P_r + \lambda^2 H(m)^H H(m))$, $r(m) = P_r^T d + \lambda^2 H(m)^H q$, so the above equation reads as $G(m)u(m) = r(m)$.

We differentiate this equation in the direction δm to obtain

$$\begin{aligned} DG(m)[\delta m]u(m) + G(m)Du(m)[\delta m] &= Dr(m)[\delta m] \\ \rightarrow Du(m)[\delta m] &= G(m)^{-1}(Dr(m)[\delta m] - DG(m)[\delta m]u(m)).\end{aligned}$$

Since $DG(m)[\delta m] = \lambda^2 (DH(m)[\delta m]^H H(m) + H(m)^H DH(m)[\delta m])$ and $Dr(m)[\delta m] = \lambda^2 DH(m)[\delta m]^H q$, we have that

$$Du(m)[\delta m] = \lambda^2 G(m)^{-1}(-H(m)^H DH(m)[\delta m]u(m) + DH(m)[\delta m]^H (H(m)u(m) - q)).$$