Nonconvex and Nonsmooth Inverse Problems

Robert Baraldi

A dissertation
submitted in partial fulfillment of the
requirements for the degree of

Doctor of Philosophy

University of Washington

2021

Reading Committee:

Aleksandr Aravkin, Chair

Randall LeVeque

James Burke

Program Authorized to Offer Degree:
Department of Applied Mathematics

University of Washington

Abstract

Nonconvex and Nonsmooth Inverse Problems

Robert Baraldi

Chair of the Supervisory Committee:
Professor Aleksandr Aravkin
Department of Applied Mathematics

Optimization approaches to inverse problems and parameter estimation have wide-ranging applications, from classical physics and biology to recently developed topics in statistical computing. Here, we focus on solving nonsmooth and nonconvex inverse problems. These properties are increasingly prevalent in modern applications yet typically approximated in many settings to simplify analysis. Such difficulties preclude common algorithms, giving rise to approaches that are highly problem specific and in many cases intractable at scale.

Nonsmooth, nonconvex inverse problems arise in a wide range of fields, from PDE-constrained optimization to machine learning applications. These objectives often have composite structure; an objective which minimizes data misfit, and a regularizer that controls model complexity. These regularizers are often nonsmooth or discontinuous, while expensive cost functions must be evaluated inexactly for numerical efficiency. We develop and analyze efficient relaxation algorithms that take advantage of this composite structure, and illustrate their performance on seismic interpolation, denoising, and data-fitting problems.

We deploy algorithms that solve these problems in as general a manner as possible, while allowing us to leverage problem structure. The main route of study is the creation of fast, first order splitting algorithms for approximate subproblems. In particular, we develop a family of splitting methods that first relaxes key parts of the inverse problem, and then solves an augmented problem with improved numerical properties and easier analyzation. Our key

application is seismic inversion, with more additional applications to data interpolation and denoising.

We extend this framework to the trust-region setting for nonlinear objectives. This requires new results that align convergence analysis from splitting methods for nonconvex and nonsmooth models with classic trust region convergence analysis. The practical implementation allows us to use derivative information from smooth problem components, and atomic operators for nonsmooth and nonconvex components, all within the context of general trust region methods for unconstrained and constrained problems. Along with theoretical results, we illustrate the efficacy of the proposed method numerically.

Finally, we address convergence for a large class of splitting methods. These work for a variety of nonconvex and even nonsmooth problems, but a-priori convergence knowledge is limited and in particular requires linear constraints. We attempt to solve both of these issues by guiding convergence with augmented Lagrangian filter methods, and solve a highly nonlinear nonnegative matrix factorizaton problem with applications to chemical spectra determination.

We conclude by proposing new directions that enable large-scale implementation of these algorithms, such as leveraging inexact evaluations of gradients and operators, as well as mixed precision arithmetic in next generation hardware. We also propose extensions to nonlinear least squares algorithms, implicit sampling techniques, and a new way of looking at splitting methods for PDE inverse problems.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

GLOSSARY

BASIC VECTOR NOTATION: Denoted $\mathbb{R}$,

- $\mathbb{R}$ set of real numbers
- $\mathbb{C}$ set of complex numbers
- $\mathbb{R}_0$ set of nonzero real numbers
- $\mathbb{R}_+$ set of nonnegative real numbers
- $\mathbb{R}^n$ set of real column vectors of dimension $n \times 1$
- $\mathbb{R}^{n\times m}$ set of real matrices of dimension $n \times m$
- $\mathbb{R}^n_+$ set of nonnegative real column vectors of dimension $n \times 1$
- $\mathbb{R}^{n\times m}_+$ set of nonnegative real matrices of dimension $n \times m$

NORMS: Denoted $\|\cdot\|$,

- $\|\cdot\|_1$ - ℓ_1 norm, with $\|x\|_1 = \sum_{i=1}^n |x_i|, x \in \mathbb{R}^n$.
- $\|\cdot\|_2$ - ℓ_2 vector norm, with $\|x\|_2 = \sqrt{\sum_{i=1}^n (x_i)^2}, x \in \mathbb{R}^n$.
  $\|\cdot\|_2$ - ℓ_2 matrix norm, with $\|X\|_2 = \max_{s\in\mathbb{R}^n, \|s\|_2=1} \|Xs\|_2 X \in \mathbb{R}^{n\times m}$.
- $\|\cdot\|_\infty$ - ℓ_∞ norm, with $\|x\|_\infty = \max_{1\le i\le n} |x_i|, x \in \mathbb{R}^n$.
- $\|\cdot\|_0$ - ℓ_1 "norm", with $\|x\|_0 = |\{i|x_i \neq 0\}|, x \in \mathbb{R}^n$.
- $\|\cdot\|_F$ - Frobenius norm, with $\|X\|_F = \sqrt{\sum_{i=1}^n \sum_{j=1}^m |X_{ij}|^2}, X \in \mathbb{R}^{n\times m}$.

FUNCTIONS: on matrices or vectors,

- $\langle A, B\rangle$ - Frobenius matrix product $\langle A, B\rangle_F = \sum_{i,j} \bar{A_{ij}} B_{ij} \equiv \mathrm{tr}\Big(\bar{A}^T B\Big)$.
- $\mathrm{tr}(A)$ - trace of a matrix $\mathrm{tr}(A) = \sum_i A_{ii}$.
- $(\cdot)^T$ - transpose of matrix $(A)^T_(ij) = A_{ij}$.
- $(\cdot)_{i,:}$ or $A(i,:)$ - i^{th} row selection.

- $(\cdot)_{:,j}$ or $A(:,j)$ - j^{th} column selection.
- $(\cdot)_{i,j}$ or $A(i,j)$ - i^{th} row selection and j^{th} column selection.
- $\sigma_i(\cdot)$ - i^{th} singular value of a matrix, decreasing order.
- $\cdot$ - componentwise multiplication $A \cdot B = A_{ij}B_{ij}$
- $\otimes$ - Kronecker product $A \otimes B = \begin{bmatrix} A_{1,1}B & \dots & A_{1,n}B \\ \vdots & \ddots & \vdots \\ A_{m,1}B & \dots & A_{m,n}B \end{bmatrix}$
- $\mathbf{1}_n$ - vector of all ones of dimension n
- I - identity matrix

COMMON OPERATORS/FUNCTIONS: usually on a function

- $\nabla\phi$ - gradient
- $\nabla^2\phi$ - Hessian
- $\partial\phi$ - subdifferential
- $|\cdot|$ - cardinality
- inf - infimum

ACRONYMS: denoted capital letters,

- prox - proximal operator
- PG - proximal gradient descent
- FISTA - Fast Iterative Shrinkage Thresholding Algorithm
- ADMM - Alternating Direction Method of Multipliers
- BFGS - Broyden Fletcher Goldfarb Shannon algorithm
- LBFGS - Limited memory BFGS
- SR1 - Symmetric Rank 1
- LSR1 - Limited memory Symmetric Rank 1
- TR - Trust Region method
- QR - Quadratic Regularization Method
- SVD - Singular Value Decomposition
- BPDN - Basis Pursuit Denoise

- NMF - Nonnegative Matrix Factorization
- NMFC - Nonnegative Matrix Factorization and Completion
- LM - Levenberg Marquardt method
- HPC - High Performance Computing

ACKNOWLEDGMENTS

I would like to thank my advisor, Aleksandr Aravkin. Ever since I mistook him for a student on visit day, I've been treated with nothing but kindness and support as his student. I'm grateful for his balance of independence and supervision, and career guidance through the quagmire applications and project proposals.

I'd like to thank my collaborators in Earth Science - Ken Creager and Carl Ulberg - for taking on a project proposed by a first year student and teaching me how bad mathematicians can be in geophysics. I would also like to thank my twice-coauthor Rajiv Kumar, who helped me set up and subsequently fry my computer with 5D seismic model forward runs. I also thank Dominique Orban, who greeted me warmly when I showed up in Montreal for no reason and continues to put up with me as I fumble around with Julia and trust-regions. I would also like to thank Randy LeVeque, who has helped me throughout grad school with fellowships, projects, and collaborations, despite being "retired". I am grateful to my committee, Randy LeVeque, Jim Burke, and Mehran Mesbahi for the support, as well as making sure I know at least something about optimization before I graduate. I am also very grateful for my main source of funding, the Department of Energy Computational Science Graduate Fellowship, and all the collaborators I have met through it: Matthew Zahr, Sven Leyffer, and Stefan Wild.

I have had the pleasure of interacting with many talented individuals in grad school, who have helped me in various ways and interacted with this work. I am very thankful to have met Donsub Rim, who arbitrarily hosted me on visit day and continues to collaborate despite doing most of the work. I am grateful to have met

Daniel Shapero, who continues to be inspiringly creative and teach me about my own field despite being in PDEs and ice flow. I am also grateful to have met and collaborated with fellow AMath students: my cohort (Megan Morrison, Yang Zhou, Ryan Creedon, and Nathan Lee) who suffered through first year together, friends (Brian de Silva, Andreas Freund, Chris Liu [also a collaborator]), and the admins who have helped me navigate UW (Lauren Lederer and Derek Franz).

This experience could not have been what it was without the help and hilarity of Erin Wagner, Weston Barger and Diya Sashidar, who helped get me through the toughest part of my life. I would also like to thank my roommates and friends Adam Elder, Nathan Lee, Walter Cai, Sheridan Grant, and Mo Zhao. I'm also thankful for my friends from other parts of the country: Ethan Baker, Amanda Cruz, and Patrick Aoude, who I met when I still thought I'd be doing bio and also helped me deal with all the nonsense. I also want to acknowledge Neel Kuila and Trey Moore, who rightfully questioned why I went to grad school in the first place. I am grateful for discovering Club Northwest, coach Tom Cotner, and my training partner Tyler van Dooren, who provided an incredible outlet and helped me somehow greatly improve as a runner.

Finally, I'd like to thank my family. Michael and Kelly Baraldi gave me unwavering support while I left to go study esoteric symbols on the other side of the country. From listening to me complain about self-imposed work to helping me through all the unexpected trauma, I appreciate all the love and patience. Thomas Baraldi gave me the grounded judgement, encouragement, and jokes to help get through this thing; you won't have to write a thesis like this, but you do have to pass the bar. I'd also like to thank my grandparents for all of the love, support, and patience, as well as being far calmer than everyone.

DEDICATION

To my family: Mike, Kelly, and Thomas Baraldi; and my grandparents, Robert and Patricia Heter. To my friend that visited me when I was recovering - Weston Barger. To my other friend that decided to come with me on the next adventure - Erin Wagner. Also to my neuroticism.

Funding Acknowledgments This material is based upon work supported by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research, Department of Energy Computational Science Graduate Fellowship under Award Number DE-FG02-97ER25308. This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

Chapter 1

INTRODUCTION

1.1 Motivation

Optimization and intelligent systems allow researchers to devise technologies that make up fundamental parts of our lives, including search engines, recommendation platforms, speech recognition, medical imaging, and full waveform seismic inversion [28]. Following major computational advances in statistics and numerical linear algebra, they have thrived by capitalizing on vast computing platforms and immense datasets. Increases in computational power, such as the rise of exascale clusters, and data availability have massively impacted statistical optimization and machine-learning (ML), and subsequent accessibility further increases the scientific and societal impacts. Despite optimization's increasing popularity, large-scale environments are distinct from the settings of traditional nonlinear techniques, which often falter due to problem size and efficiency.

This numerical cost is often compounded by the construction of inverse problems. Inverse problem solving is a fundamental technique in modern mathematics concerned with numerically computing parameters by minimizing an objective function. These parameters are then used to make decisions on unobserved data [28]. A plethora of optimization algorithms exist to tackle inverse problems in seismology [12, 13, 131], physics [137], and many others. Within the national lab and university systems, optimization algorithms are used to solve inverse problems in biology [1], quantum computing (QOALAS/FAR-QC) and physics modeling [140], and train machine-learning (ML) models [110]. Objective functions are often *composite* in that one minimizes the sum of a smooth function with a nonsmooth function. The nonsmooth term is typically a *regularizer*, and promotes sparsity for ill-conditioned problems. Vast datasets and problem size encourage overfitting, but ill-posedness can be tempered with regularization functions that often subvert the traditional notion of derivatives and optimality. Examples include Total-Variation regularization for PDEs [4, 131], low-rank minimization, and matrix completion [116]. Almost all optimization routines use some form of derivative information, but this cannot be computed where objective functions are discontinuous, and costly functions must be evaluated inexactly if one is to be efficient.

Nonconvex and nonsmooth functions are notorious difficult to solve despite their commonality in physical problems. Researchers have often avoided these problems by solving smoothed or convex versions of the cost function. Classic approaches to these problems, such as least squares and direct decompositions, require high data quality and often erroneous assumptions on noise and error. To account for this, regularizers, priors, and constraints are often implemented to promote some degree of model simplicity in ill-posed or high-dimensional settings. These regularizers are often chosen carefully, as special function structures are exploited in the solution of a given problem [142].

In the most general form, we consider nonconvex composite problems with regularizers and constraints

$$\min_x f(x) + h(x) \quad s.t.\ c(x) \in \mathcal{X} \tag{1.1}$$

where $\mathcal{X}$ is some set and $x \in \mathbb{R}^n$ are the decision variables. We take $h : \mathbb{R}^c \to \mathbb{R}$ may be nonsmooth and nonconvex, and $f : \mathbb{R}^n \to \mathbb{R}$ is smooth. Largely, the only enforced constraint is that our functions have proximal and projection operators.

We approach these problems primarily with proximal gradient and relax and split methods, the latter of which seeks to augment (1.1) in such a way that minimizes the difference between dummy variable(s) w and more difficult components of the cost function. The gap between these primal variables is controlled via some tuning parameter η that may be augmented. The goal is to choose relaxation in such a way that makes (1.1) amenable to first-order proximal-gradient type methods. We later combine this approach with trust region methods to tackle highly nonlinear regularized problems. Many trust region methods rely on a degree of smoothness, which we try to dispense. This also allows for the proximal gradient methods to be more efficient for computationally intense models, since they are solving a quadratic approximation. The motivation of this work is hence twofold:

1. develop algorithms for a class of regularized problems that have remained largely separate in the literature;

2. increase efficiency and generalizability in current methods.

As modern scientific computing continues to depend on extracting information from often prohibitively large datasets, algorithms must adapt in kind. As data availability increases, algorithms must scale and regularize in turn while simultaneously retaining speed and robustness to low-fidelity measurements.

Roadmap: A chart of chapters, problems dealt with, and methods used is depicted in Figure 1.1. Chapter 2 deal with notational norms, definitions, and algorithms upon whose analysis this work rests. Chapter 3 establishes a scheme for relax and split methods for seismic data interpolation around Mount St. Helens. The goal in this particular application is to use existing travel time data from real stations to interpolate station data in new locations while simultaneously accounting for noisy measurements. We show that our algorithm is faster and more accurate than competitors. Chapter 4 presents a new algorithm for the classic Basis Pursuit Denoise (BPDN) problem that can handle a variety of cost functions and constraints for different noise types. We show that this algorithm converges rapidly and accurately to the true solution with both Gaussian and non-Gaussian noise. Chapter 5 expands this theme to nonlinear, nonsmooth trust region problems. Here we tackle regularized inverse problems and develop a novel method of computing trust region subproblem solutions for these nonsmooth objective functions. Chapter 6 solves a slightly different class of problems; we assume (5.1) has $h = 0$ and $f \in \mathcal{C}^2$, but f is separable and has nonlinear constraints; the latter of which is a known issue for ADMM convergence. We attempt to resolve this with filter methods, which allow for increases in the penalty parameter via conditions on the feasibility and first-order error relationships. Chapter 7 pivots to future directions regarding this work as well as upcoming projects.

1.2 Strategies and Objectives

The primary objective in this body of work is to develop optimization methods to solve nonsmooth regularized inverse problems. We start slowly with a particular example in Chapter 3, and slowly build out to a more general problem class as the work proceeds; see Figure 1.1. Regularizers often confound standard solvers; hence, our focus is to develop generalized

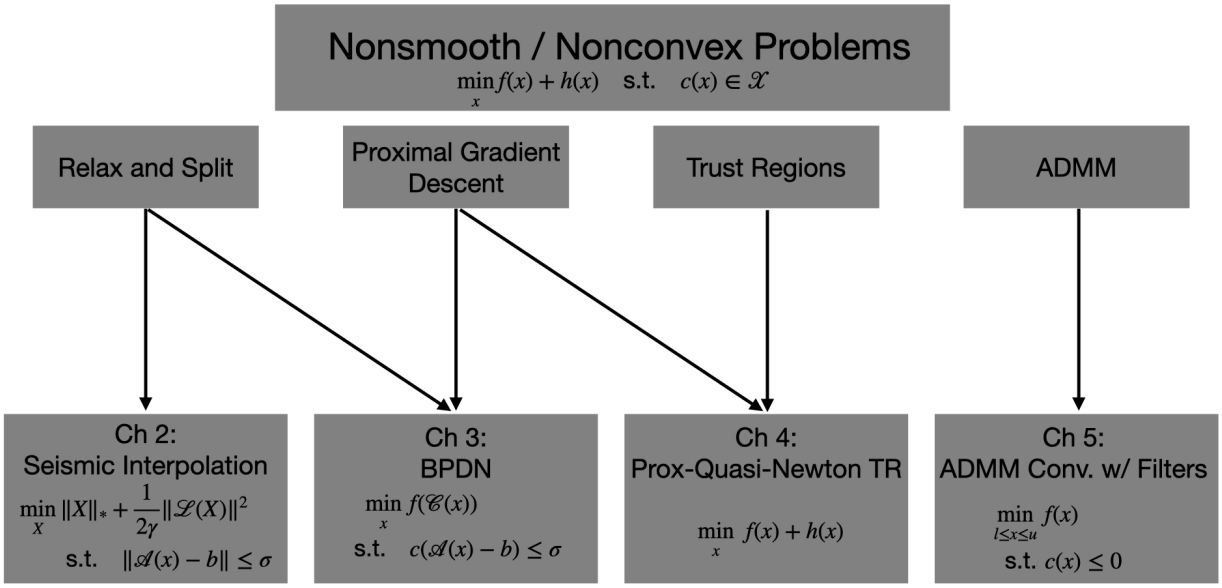


Figure 1.1: Structure of thesis - basic optimization methods with respective developments and applications.

methods for nonsmooth, nonconvex regularizers. The strategy taken is twofold, but built upon the same basic tool: proximal gradient descent. Therefore, the methods developed in this work are first order in nature, which is the most one can hope for in virtually every regularizer. The first strategy taken is relax and split: this effectively decomposes the problem into digestable parts. We analyze algorithm behavior of this method in nonsmooth, nonconvex contexts via proximal gradient analysis and show convergence to stationary points. The second strategy is combining the proximal gradient based analysis with trust region methods. This effectively pushes the regularizer into the subproblem, which can be solved with variations on analytic solutions to standard proximal operators. These technologies yield and efficient solver for problems hitherto untouched by current state of the art. With regards to future goals, this work seeks to expand the analysis to inexact solves and stochastic regimes.

1.3 Accomplishments and Outline

This thesis develops algorithms for nonsmooth and nonconvex inverse problems. The primary contribution is twofold: 1) an algorithm for a general class of basis pursuit denoise problems with nonsmooth constraints; 2) a fusion of trust region and proximal methods for nonlinear, nonsmooth regularized problems. These contributions were built on the fundamental properties of proximal operators and trust regions, and the implications of this work extend beyond the scope of the inverse problem applications discussed here.

In Chapter 3 of this work, we apply proximal gradient and splitting techniques to a novel regularized matrix factorization problem with applications to travel time tomography. Travel time tomography is used to infer the underlying three-dimensional wavespeed structure of the Earth by fitting seismic travel time data collected at surface stations. Data interpolation and denoising techniques are important pre-processing steps that use prior knowledge about the data, including parsimony in the frequency and wavelet domains, low-rank structure of matricizations, and local smoothness. We show how local smoothness structure can be combined with low rank constraints using level-set optimization formulations, and develop a new relaxation algorithm that can efficiently solve these joint problems. In the seismology

setting, we use the approach to interpolate missing stations and de-noise observed stations. The new approach is competitive with alternative algorithms, and offers new functionality to interpolate observed data using both smoothness and low rank structure in the presence of data fitting constraints. Relevant works: [13]

- Robert Baraldi, Carl Ulberg, Rajiv Kumar, Kenneth Creager, and Aleksandr Aravkin. Relaxation algorithms for matrix completion, with applications to seismic travel-time data interpolation. *Inverse Problems*, 35(10):105009, 2019.

- Paper figure code: https://github.com/rjbaraldi/IMUSH-Interpolation

In Chapter 4, we extend the above formulation to a more general set of problems that occur in basis pursuit denoise problems. Level-set optimization formulations with data-driven constraints minimize a regularization functional subject to matching observations to a given error level. These formulations are widely used, particularly for matrix completion and sparsity promotion in data interpolation and denoising. The misfit level is typically measured in the ℓ_2 norm, or other smooth metrics. This is a new flexible algorithmic framework that targets *nonsmooth* level-set constraints, including ℓ_1, ℓ_∞, and even ℓ_0 norms. These constraints give greater flexibility for modeling deviations in observation and denoising, and have significant impact on the solution. Measuring error in the ℓ_1 and ℓ_0 norms makes the result more robust to large outliers, while matching many observations exactly. We demonstrate the approach for basis pursuit denoise (BPDN) problems as well as for extensions of BPDN to matrix factorization, with applications to interpolation and denoising of 4D seismic data. The new methods are particularly promising for seismic applications, where the amplitude in the data varies significantly, and measurement noise in low-amplitude regions can wreak havoc for standard Gaussian error models. The relevant paper being [12], with code:

- Robert Baraldi, Rajiv Kumar, and Aleksandr Aravkin. Basis pursuit denoise with nonsmooth constraints. *IEEE T. Signal Proces.*, 67(22):5811–5823, 2019.

- Paper figure code:
  https://github.com/rjbaraldi/bpdn-with-nonsmooth-constraints

In Chapter 5, we combine the nonsmooth analysis developed earlier with trust region methods. We develop a trust-region method for minimizing the sum of a smooth term f and a nonsmooth term h, both of which can be nonconvex. Each iteration of our method minimizes a possibly nonconvex model of $f + h$ in a trust region. The model coincides with $f + h$ in value and subdifferential at the center. We establish global convergence to a first-order stationary point when f satisfies a smoothness condition that holds, in particular, when it has Lipschitz-continuous gradient, and h is proper and lower semi-continuous. The model of h is required to be proper, lower-semi-continuous and prox-bounded. Under these weak assumptions, we establish a worst-case $O(1/\epsilon^2)$ iteration complexity bound that matches the best known complexity bound of standard trust-region methods for smooth optimization. We detail a special instance in which we use a limited-memory quasi-Newton model of f and compute a step with the proximal gradient method, resulting in a practical proximal quasi-Newton method. We establish similar convergence properties and complexity bound for a quadratic regularization variant, and provide an interpretation as a proximal gradient method with adaptive step size for nonconvex problems. We describe our Julia implementations and report numerical results on inverse problems from sparse optimization and signal processing. Our trust-region algorithm exhibits promising performance and compares favorably with linesearch proximal quasi-Newton methods based on convex models.

- Robert Baraldi, Aleksandr Aravkin, and Dominique Orban. A Proximal Quasi-Newton Trust-Region Method for Nonsmooth Regularized Optimization. *SIAM Journal of Optimization - in revision*, 2020.

- ArXiv preprint: https://arxiv.org/pdf/2103.15993.pdf

- Code: https://github.com/rjbaraldi/ShiftedProximalOperators and https://github.com/UW-AMO/TRNC

Our last problem concerns solving separable problems in Chapter 6. Large, nonconvex, and nonlinear problems arise naturally in physical inverse problems and machine learning applications. The Alternating Direction Method of Multipliers (ADMM) is one of the workhorses for these applications. Unfortunately, it is not known whether ADMM converges for many nonconvex applications, resulting in slow or uncertain convergence behavior. We study filter methods for promoting global convergence of ADMM scheme. Filter methods have proved particularly promising for enforcing convergence for sequential quadratic programming methods and interior point methods. We develop an ADMM filter method for highly nonlinear and nonconvex problems, and in particular focus on nonlinear non-negative matrix factorization and completion problems. We explore the convergence properties of an ADMM filter method with the intention of guaranteeing speedy convergence to at least local minima.

- Robert Baraldi, Stefan Wild, and Sven Leyffer. Guiding Convergence of the Alternating Direction Method of Multipliers with Augmented Lagrangian Filters. *in preparation*, 2021.

In Chapter 7, we discuss the implications and extensions of the work developed in prior chapters, primarily with regards to nonsmooth regularized least squares, barrier and penalty methods, and inexact evaluations. We also discuss the novel application of developed methods to PDE-constrained optimization and ice sheet flow.

- Robert Baraldi, Aleksandr Aravkin, and Dominique Orban. A Levenberg-Marquardt Method for Nonsmooth Regularized Least Squares. *In preparation*, 2021.

Chapter 2

PRELIMINARIES

2.1 Notation

Sets are represented by calligraphic letters. The cardinality of set $\mathcal{S}$ is represented by $|\mathcal{S}|$. So to are operators, which we use as functions with particular properties and transform to specific domains. Examples of these are the sampling operator $\mathcal{A}$ and the Curvelet transformation operator $\mathcal{C}$. We use $\|\cdot\|$ to denote a generic norm on $\mathbb{R}^n$. The symbols ν, λ, σ, and Δ are scalars. Vector variables are denoted by lower-case letters; typically $s, x, y, z, \ldots$. Scalar variables are *typically* denoted by Greek letters, ν, Δ, but be aware of context. Matrix variables are by capital English alphabet letters: A, B, C etc. Variables that are being decomposed into blocks in a specific context are indicated by **boldness**: such as $\mathbf{x}, \mathbf{s}$, or $\mathbf{A}$. $\mathbb{B}(0, \Delta)$ is the ball centered at 0 with radius $\Delta > 0$ defined by a norm that should be clear from the context. We use the shorthands $\mathbb{B} = \mathbb{B}(0, 1)$ and $\Delta\mathbb{B} = \mathbb{B}(0, \Delta)$. When necessary, we write $\mathbb{B}_p$ to indicate that the ℓ_p-norm is used. Functional symbols f, g, h, as well as ϕ, φ and ψ are used for functions. Typically we use f, φ, ϕ to denote smooth functions and h, ψ to denote nonsmooth functions. $\chi(\cdot; A)$ represents the indicator function of $A \subseteq \mathbb{R}^n$. In particular, the indicator of $\mathbb{B}(0, \Delta)$ is denoted $\chi(\cdot; \Delta\mathbb{B})$ or just $\chi(\cdot; \Delta)$ when the norm is clear from the context. We use the alternative notation $\chi(\cdot; \Delta\mathbb{B}_p)$ to emphasize that the ℓ_p-norm is used to define the ball. If $A \subseteq \mathbb{R}^n$ is closed and convex, $\mathrm{proj}_A(x)$ denotes the projection of x into A. Finally, j and k are iteration counters.

2.2 Definitions and Operators

We summarize some basic defintions from [17, 120]. We denote $\overline{\mathbb{R}} = \mathbb{R} \cup \{\pm\infty\}$. We call $\phi : \mathbb{R}^n \to \overline{\mathbb{R}}$ *proper* if $\phi(x) > -\infty$ for all x and $\phi(x) < \infty$ for at least one x, and *lower semi-continuous*, or *lsc*, at $\bar{x}$ if $\liminf_{x\to\bar{x}} \phi(x) = \phi(\bar{x})$. Similarly, ϕ is *upper semi-continuous*, or *usc*, at $\bar{x}$ if $\limsup_{x\to\bar{x}} \phi(x) = \phi(\bar{x})$. If ϕ is both lsc and usc at $\bar{x}$, it is continuous at $\bar{x}$. We say that ϕ is *(lower-)level bounded* if all its level sets are bounded. If ϕ is proper, lsc and level bounded, then $\arg\min \phi$ is nonempty and compact [120, Theorem 1.9].

2.2.1 Nonsmooth operators

First, we define the indicator function to be

Definition 1 (Indicator Function)**.**

$$\chi(x;\mathbb{B}) = \begin{cases} 0, & x \in \mathbb{B} \\ \infty, & else \end{cases}.$$

and then the projection and proximal operator:

Definition 2 (Projection Operator)**.**

$$\underset{\mathbb{B}}{\text{proj}}(y) := \underset{x\in\mathbb{B}}{\arg\min}\, \tfrac{1}{2}\|y-x\|^2$$

Definition 3. *For a proper lsc function $\phi : \mathbb{R}^n \to \overline{\mathbb{R}}$ and a parameter $\nu > 0$, the* Moreau envelope $e_{\nu\phi}$ *and the* proximal mapping $\text{prox}_{\nu\phi}$ *are defined by*

$$e_{\nu\phi}(x) := \inf_w \tfrac{1}{2}\nu^{-1}\|w-x\|^2 + \phi(w) = \nu^{-1} \inf_w \tfrac{1}{2}\|w-x\|^2 + \nu\phi(w), \tag{2.1a}$$

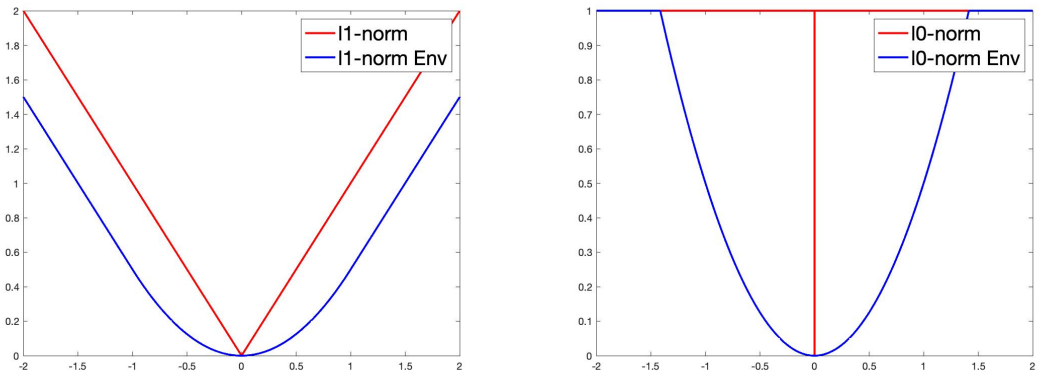
$$\underset{\nu\phi}{\text{prox}}(x) := \underset{w}{\arg\min}\, \tfrac{1}{2}\nu^{-1}\|w-x\|^2 + \phi(w) = \underset{w}{\arg\min}\, \tfrac{1}{2}\|w-x\|^2 + \nu\phi(w). \tag{2.1b}$$

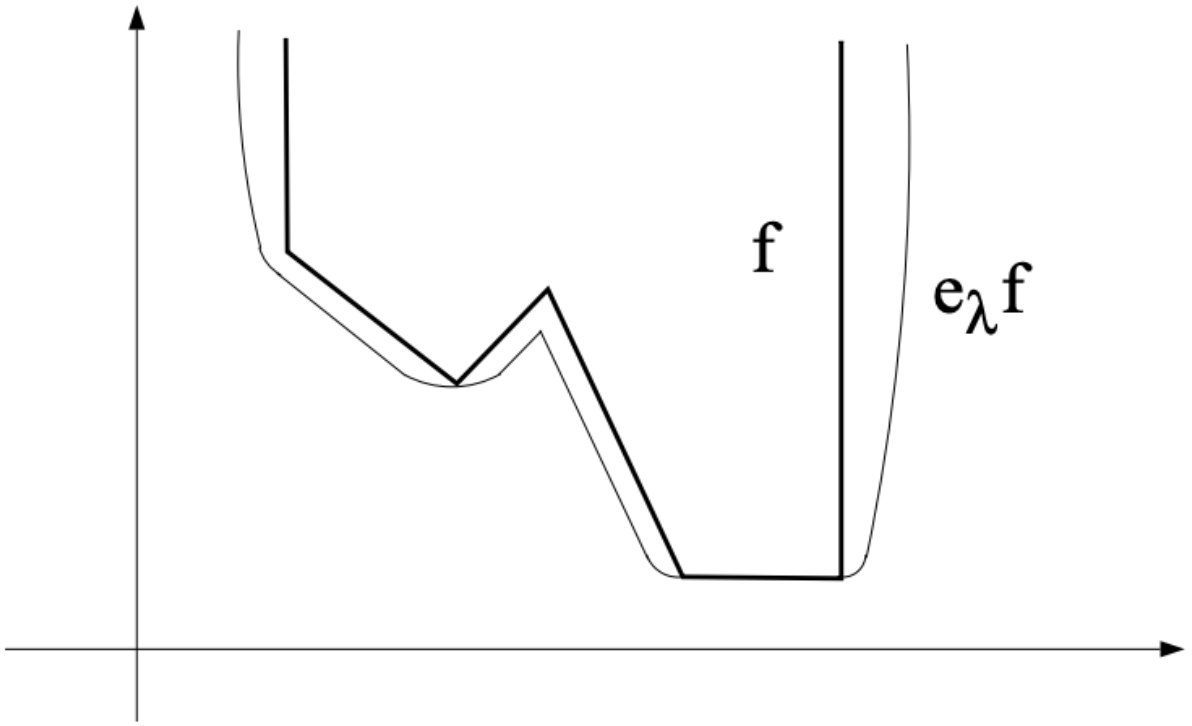
Under certain assumptions, including strong convexity of the objective of (2.1b), the set $\text{prox}_{\nu\phi}(x)$ is a singleton. However, in general, the set-valued mapping $\text{prox}_{\nu\phi}$ may assume empty values or values that contain multiple elements. For a given ϕ, the range of parameter values for which the Moreau envelope assumes a finite value is given by the following definition. Some examples can be seen in Figure 2.1.

Definition 4 (Prox-Boundedness)**.** *The proper lsc function $\phi : \mathbb{R}^n \to \overline{\mathbb{R}}$ is* prox-bounded *if there exists $\nu > 0$ and at least one $x \in \mathbb{R}^n$ such that $e_{\nu\phi}(x) > -\infty$. The* threshold of prox-boundedness ν_ϕ *of ϕ is the supremum of all such $\nu > 0$.*

We can write $\nu_\phi = \min(\infty, 1/\rho_\phi)$, where $\rho_\phi := \inf\{\rho \in \mathbb{R} \mid x \mapsto \phi(x) + \frac{1}{2}\rho\|x\|^2\}$ is bounded below, with $\nu_\phi = \infty$ if ϕ is bounded below. If ϕ is level bounded, then so is

Figure 2.1: Two common proximal operators and their envelopes ($\nu = \lambda = 1$), including a third from [120].





$w \mapsto \frac{1}{2}\nu^{-1}\|w-x\|^2 + \phi(w)$ for all $x \in \mathbb{R}^n$ and all $\nu > 0$, so $e_{\nu\phi}(x) > -\infty$ [120, Theorem 1.9] and ϕ is prox-bounded. The following result summarizes some properties of (2.1a)–(2.1b). Further properties appear in [120, Theorem 1.25].

Proposition 2.2.1. *Let $\phi : \mathbb{R}^n \to \overline{\mathbb{R}}$ be proper lsc and prox-bounded with threshold $\nu_\phi > 0$. For every $\nu \in (0, \nu_\phi)$ and all $x \in \mathbb{R}^n$,*

1. $\operatorname{prox}_{\nu\phi}(x)$ *is nonempty and compact;*

2. $e_{\nu\phi}(x)$ *depends continuously on (ν, x) and $e_{\nu\phi}(x) \nearrow \phi(x)$ as $\nu \searrow 0$.*

3. *if $\{x_k\} \to \bar{x}$ and $\{\nu_k\} \searrow 0$ in such a way that $\{\|x_k - \bar{x}\|/\nu_k\}$ is bounded, then $\{e_{\nu_k\phi}(x_k)\} \to \phi(\bar{x})$;*

4. *if $\{x_k\} \to \bar{x}$ and $\{\nu_k\} \to \bar{\nu} \in (0, \nu_\phi)$ and for each k, $w_k \in \operatorname{prox}_{\nu_k\phi}(x_k)$, then $\{w_k\}$ is bounded and all its limit points are in $\operatorname{prox}_{\bar{\nu}\phi}(\bar{x})$.*

From here, it is pretty easy to show that Moreau Envelope is smooth (but we just state it).

Theorem 2.2.2 (Smoothness of Moreau Envelope)**.** *Let $\phi : \mathbb{R} \to (-\infty, \infty]$ be a proper closed and convex function and have $\nu > 0$. Then $e_{\nu\phi}$ is $\frac{1}{\nu}$-smooth over $\mathbb{R}$ and for any $x \in \mathbb{R}^n$*

$$\nabla e_{\nu\phi}(x) = \frac{1}{\nu}\left(x - \operatorname*{prox}_{\nu\phi}(x)\right)$$

Before we proceed, we need to define the concept of the subdifferential and optimality for nonsmooth problems.

2.2.2 Optimality conditions

We use the following notions of subgradient and subdifferential [120, Definition 8.3].

Definition 5 (Limiting subdifferential)**.** *Consider $\phi : \mathbb{R}^n \to \overline{\mathbb{R}}$ and $\bar{x} \in \mathbb{R}^n$ with $\phi(\bar{x}) < \infty$. We say that $v \in \mathbb{R}^n$ is a* regular subgradient *of ϕ at $\bar{x}$, and we write $v \in \hat{\partial}\phi(\bar{x})$ if*

$$\liminf_{x \to \bar{x}} \frac{\phi(x) - \phi(\bar{x}) - v^T(x - \bar{x})}{\|x - \bar{x}\|} \geq 0.$$

The set of regular subgradients is also called the Fréchet subdifferential*. We say that v is a* general subgradient *of ϕ at $\bar{x}$, and we write $v \in \partial\phi(\bar{x})$, if there are sequences $\{x_k\}$ and $\{v_k\}$ such that $x_k \to \bar{x}$, $\phi(x_k) \to \phi(\bar{x})$, $v_k \in \hat{\partial}\phi(x^k)$ and $v^k \to v$. The set of general subgradients is called the* limiting subdifferential*.*

If ϕ is convex, the Fréchet and limiting subdifferentials coincide with the subdifferential of convex analysis. If ϕ is differentiable at x, $\hat{\partial}\phi(x) = \partial\phi(x) = \{\nabla\phi(x)\}$. For nonconvex or convex ϕ, we can simplify Definition 5 to local behavior:

Definition 6. *Consider $\phi : \mathbb{R}^n \to \bar{\mathbb{R}}$ and $\bar{x} \in \mathbb{R}^n$ with $\phi(\bar{x}) \leq \infty$. We say that $v \in \mathbb{R}^n$ is a local subgradient of ϕ at $\bar{x}$*

$$\phi(x) \geq \phi(\bar{x}) + \langle v, x - \bar{x} \rangle + o(\|x - \bar{x}\|) \tag{2.2}$$

$$\phi(x) \geq \phi(\bar{x}) + \langle v, x - \bar{x} \rangle \tag{2.3}$$

for nonconvex and convex functions, respectively. Note that for convex functions, (2.3) is global.

In the following, we do not make use of the precise definition of the relevant subdifferential, but merely rely on the following criticality property.

Proposition 2.2.3 (120, Theorem 10.1)**.** *If $\phi : \mathbb{R}^n \to \overline{\mathbb{R}}$ is proper and has a local minimum at $\bar{x}$, then $0 \in \hat{\partial}\phi(\bar{x}) \subseteq \partial\phi(\bar{x})$. If ϕ is convex, the latter condition is also sufficient for $\bar{x}$ to be a global minimum. If $\phi = \varphi + \psi$ where φ is differentiable on a neighborhood of $\bar{x}$ and ψ is finite at $\bar{x}$, then $\partial\phi(\bar{x}) = \nabla\varphi(\bar{x}) + \partial\psi(\bar{x})$.*

It is also conducive to show some proximal operator properties, which we denote the Prox-subgradient theorem.

Theorem 2.2.4 (Prox-subgradient Theorem)**.** . *Let $\phi : \mathbb{R} \to (-\infty, \infty]$ be a proper closed and convex function. Then for any $x, s \in \mathbb{R}^n$, the following claims are equivalent:*

1. $s = \text{prox}_{\nu\phi}(x)$
2. $x - s \in \partial\phi(s)$
3. $\langle x - s, y - s\rangle \leq \phi(y) - \phi(s)$ *for any* $y \in \mathbb{R}^n$.

Proof. By definition $s = \text{prox}_{\nu\phi}(x)$ iff s is the minimizer of

$$\min_x \left\{ \phi(x) + \frac{1}{2}\|s - x\|^2 \right\}$$

which is equivalent to

$$0 \in \partial\phi(s) + s - x.$$

This is the same as 1 and 2. Finally, the subgradient definition implies the equivalence of 3 to 2. □

We also note that the proximal operator is firmly not expansive and nonempty if the prox is closed and coercive. Another useful relationship is the Moreau Decomposition. First, we have to define the Fenchel conjugate of a function.

Definition 7. *We let $\mathbb{X}$ be a topological real vector space and let $\mathbb{X}^*$ be the dual space to $\mathbb{X}$. Denote*

$$\langle \cdot, \cdot \rangle : \mathbb{X}^* \times \mathbb{X} \to \mathbb{R}$$

the canonical dual pairing, defined by $(x^*, x) \mapsto x^*(x)$. For a function $\phi : X \to \mathbb{R} \cup \{-\infty, \infty\}$, the convex conjugate is

$$\phi^* : \mathbb{X}^* \to \mathbb{R} \cup \{-\infty, \infty\}$$

whose value at $x^* \in \mathbb{X}^*$ is defined to be the supremum

$$\phi^*(x^*) := \sup_{x \in \mathbb{X}} \langle x^*, x\rangle - f(x).$$

This definition is the convex hull of the function's epigraph in terms of its supporting hyperplanes. Now, we can talk about the Moreau Decomposition.

Theorem 2.2.5 (Moreau Decomposition)**.** *Let $\phi : \mathbb{R} \to (-\infty, \infty]$ be proper closed and convex. Then for any $s \in \mathbb{R}^n$,*

$$\underset{\nu\phi}{\text{prox}}(x) + \underset{\nu\phi^*}{\text{prox}}(x) = x$$

and

$$e_{\nu\phi}(s) + e_{\nu\phi^*}(s) = \frac{1}{2}\|s\|^2.$$

Proof. For the first part: Let $x \in \mathbb{R}^n$ and denote $s = \text{prox}_{\nu\phi}(x)$. By the prox-subgradient theorem 2.2.4, we have that $x - s \in \partial\phi(s)$, which is equivalent to

$$s \in \partial\phi^*(x - s).$$

Using Theorem 2.2.4 again, we have $x - s = \text{prox}_{\nu\phi^*}(x)$. Hence,

$$\underset{\nu\phi}{\text{prox}}(x) - \underset{\nu\phi^*}{\text{prox}}(x) = s + (x - s) = x$$

□

One can think of the proximal operator is the strict generalization of the projection operator, while the Moreau envelope is the strict generalization of the distance function. The proximal operator is uniquely determined since the squared norm is strongly convex. Taking the derivative shows that

$$\underset{\nu\phi}{\text{prox}} = (I + \nu\partial\phi)^{-1}$$

namely $s = \text{prox}_{\nu\phi}(x) \Leftrightarrow \ x \in x + \nu\partial\phi(x)$. Before we conclude, we note that the Moreau Envelope is differentiable.

Theorem 2.2.6 (Differentiability)**.** *The Moreau envelope is Frechét differentiable, with* $\nabla e_{\nu\phi} = I - \text{prox}_{\nu\phi} = \text{prox}_{\nu\phi^*}$

and that a useful relationship between the envelope and its function exists.

Theorem 2.2.7 (Relationship between Envelope and Function)**.** *For any* $\nu > 0$, $s \in \arg\min \phi \Leftrightarrow s \in \arg\min e_{\nu\phi} \Leftrightarrow s = \text{prox}_{\nu\phi}(x)$

Proof.

$$0 \in \partial\phi(s) \Leftrightarrow s \in s + \nu \cdot \partial\phi(s) \Leftrightarrow s - \underset{\nu\phi}{\text{prox}}(s) = 0 \Leftrightarrow \nabla e_{\nu\phi}(s) = 0$$

□

You can think of a projection as the proximal operator of the level set (ie proximal operators are the generalization of a projection).

Theorem 2.2.8 (Projection Theorem)**.** *Let* $\phi : \mathbb{R}^n \to (-\infty, \infty]$ *be given by* $\phi(s) = \chi(x; C)$ *where* C *is a nonempty set. Then*

$$\underset{\nu\phi}{\text{prox}}(x) = \underset{s\in\mathbb{R}^n}{\arg\min} \left\{ \chi(s; C) + \frac{1}{2}\|s - x\|^2 \right\} = \underset{s\in C}{\arg\min} \|s - x\|^2 = \underset{C}{\text{proj}}(x)$$

Hence, we have the proximal mapping of the indicator function of a given set is the orthogonal projection operator onto the same set. If C is closed and convex, in addition to being nonempty, the indicator function $\chi(\cdot; C)$ is proper closed and convex, and hence the first prox Theorem 2.2.8 the orthogonal projection mapping exists and is unique. Now, with the basic theory out of the way, we can proceed to describe the fundamental algorithms we will be using in this work.

2.3 Basic Algorithms

We use many algorithmic tools in the applications and algorithms developed in this work. These algorithms are general to a certain degree; all assume some particular structure in the cost function; namely that it comprises of the sum of a smooth and nonsmooth term (1.1). The main workhorse in this analysis is the proximal operator (2.1b), and more broadly proximal gradient methods. Note that we typically pick optimality conditions to be the norm of the subdifferential (5).

2.3.1 The proximal gradient method

Consider the generic nonsmooth regularized problem

$$\underset{s}{\text{minimize}} \;\; \varphi(s) + \psi(s), \tag{2.4}$$

where φ is continuously differentiable and ψ is proper, lower semi-continuous and prox-bounded (Definition 4). A natural method to solve (2.4) that generalizes the gradient method of smooth optimization is the *proximal gradient method* [15, 91]. This has been a very popular method for a long time, and the basic method is still subject to ongoing vigorous research [9, 26]. When initialized from $s_0 \in \mathbb{R}^n$ where ψ is finite, it generates iterates according to

$$s_{j+1} \in \underset{\nu\psi}{\text{prox}}(s_j - \nu\nabla\varphi(s_j)), \quad j \geq 0, \tag{2.5}$$

where $\nu > 0$ is a step size. This algorithm is generated by taking a gradient step in the smooth part φ, and then regularizing the Moreau Envelop Equation (2.1a) of the nonsmooth part ψ and then completing the square in the first-order Taylor expansion of φ. If ψ is the indicator of a closed convex set, the proximal gradient method reduces to the projected gradient method.

The first-order optimality conditions of (2.5) are

$$0 \in s_{j+1} - s_j + \nu\nabla\varphi(s_j) + \nu\partial\psi(s_{j+1}). \tag{2.6}$$

The proximal literature primarily focuses on the generalized gradient

$$G_\nu(s) := \nu^{-1}(s - \operatorname*{prox}_{\nu\psi}(s - \nu\nabla\varphi(s))), \tag{2.7}$$

with $G_\nu(0) = \nabla\varphi(0)$ in the case of smooth optimization. The following result gives conditions under which the proximal gradient method is monotonic.

Proposition 2.3.1 (26, Lemma 2)**.** *Let φ be continuously differentiable, $\nabla\varphi$ be Lipschitz continuous with constant $L > 0$ and ψ be proper, lsc and bounded below. For any $0 < \nu < 1/L$, any s_0 where ψ is finite, the iteration (2.5) is such that*

$$(\varphi + \psi)(s_{j+1}) \leq (\varphi + \psi)(s_j) - \tfrac{1}{2}(\nu^{-1} - L)\|s_{j+1} - s_j\|^2, \quad j \geq 0.$$

Stronger assertions exist when ψ is convex, but we focus on general ψ. It is possible to remove the assumption that ψ is bounded below from Proposition 2.3.1 and replace it with the weaker assumption that ψ is prox-bounded and that ν is chosen smaller than the threshold of prox-boundedness of ψ.

In the smooth case, where $\psi = 0$, we have $s_1 = -\nu\nabla\varphi(s_0)$ and the decrease is

$$\varphi(s_1; x) \leq \varphi(s_0) - \tfrac{1}{2}\nu^2(\nu^{-1} - L)\|\nabla\varphi(s_0)\|^2. \tag{2.8}$$

Algorithm 1 Proximal Gradient Algorithm.

1: Choose $s_0 \in \mathbb{R}^n$ where ψ is finite.
2: **for** $j = 0, 1, \ldots$ **do**
3: Choose $0 < \nu_j$ such that $\nu_j < 1/L$.
4: Update the the proximal-step by computing the point (2.5)

$$s_{j+1} \in \operatorname*{prox}_{\nu_j\psi}(s_j - \nu_j\nabla\varphi(s_j))$$

The convergence rate for smooth, convex φ and convex ψ PG is given by

$$\psi(s_j) - \psi_* \leq \frac{1}{2j\nu}\|s_0 - s_*\|^2 \tag{2.9}$$

where ψ_* is the finite optimal value of ψ attained at s_*. The conclusion is that $\psi(s_j) - \psi_* \leq \epsilon$ after $\mathcal{O}(1/\epsilon)$ iterations. Later, we develop a convergence criteria independent of convexity. We note that there are many versions of PG; the most popular of which is using a linesearch method to determine ν instead of basing it off of the typically unknown Lipschitz constant L. Much like how proximal gradient can be viewed as an analog for steepest descent, there is also a counterpart corresponding to the optimal gradient method.

2.3.2 FISTA

While proximal gradient descent is fundamental to the function of many first order algorithms, there have been many advancements and improvements. Accelerated gradient methods introduced by [104] are as simple as plain gradient descent, but have a much faster convergence rate. These methods are optimal for smooth and convex problems with Lipschitz-continuous gradient. Their worst-case complexity is proportional to the theoretical lower complexity bound of first-order methods for this class of problems. These methods are connected to the Störmer-Verlet method for discretizing second-order ODES [107]. The Fast-Iterative-Shrinkage-Thresholding Algorithm (FISTA) was first developed to solve image-deblurring problems [18].

Algorithm 2 FISTA.

1: Choose $s_0 \in \mathbb{R}^n, y_1 = s_0, t_1 = 1.0$ where ψ is finite.

2: **for** $k = 0, 1, \ldots$ **do**

3: Choose $0 < \nu_j$ such that $\nu_j < 1/L$.

4: Update the the proximal-step by computing the point (2.5)

$$s_{j+1} \in \underset{\nu_j \psi}{\text{prox}}(y_j - \nu_j \nabla \varphi(y_j))$$

5: $t_{j+1} = \frac{1+\sqrt{1+4t_j^2}}{2}$

6: $y_{j+1} = s_j + \frac{t_j - 1}{t_{j+1}}(s_j - s_{j-1})$

Convergence analysis for smooth, convex φ and convex ψ gives us

$$\psi(s_j) - \psi_* \leq \frac{2\|s_0 - s_*\|^2}{\nu(j+1)^2} \tag{2.10}$$

which is faster than standard PG, as it reaches ϵ error in $\mathcal{O}(1/\sqrt{\epsilon})$ (or $1/j^2$) iterations. We pause note here that backtracking linesearch implementation for ν_j does not affect convergence for both FISTA and PG. We also note that FISTA accelerates at the sacrifice of monotonicity; monotonic versions of the algorithm exist, and are implemented by simply taking the prox step when the momentum step does not decrease the objective. Next, we expand our nonsmooth algorithms to multiple variables by detailing relax-and-split.

2.3.3 Relax and Split

Relaxation and splitting methods have around for some time and are reminiscent of the Alternating Direction Method of Multipliers, albeit entirely in the primal space. In [152], the authors consider the composite problem

$$\underset{s}{\text{minimize}}\, \phi(s) := \psi(As) + \varphi(s) \tag{2.11}$$

where $s \in \mathbb{R}^n$ are the decision variables, $A = [a_1, \ldots, a_m]^T \in \mathbb{R}^{m \times n}$ is a linear mapping, $\psi : \mathbb{R}^m \to \mathbb{R}$ is nonsmooth, nonconvex, and separable $\psi = \sum_{i=1}^m \psi_i(\langle a_i, s \rangle)$, and φ is convex. The key step is to introduce an auxiliary variable w and then use partial minimization over the original variables. This relaxed version of (2.11) is

$$\underset{w,s}{\text{minimize}}\, \phi_\nu(s, w) := \psi(w) + \tfrac{1}{2\nu}\|As - w\|^2 + \varphi(s) \tag{2.12}$$

where $w \approx As$. This allows for a partial minimization scheme with

$$\varphi_\nu(w) := \min_s \tfrac{1}{2\nu}\|As - w\|^2 + \varphi(s)$$

which means that (2.12) is equivalent to

$$\min_w \phi_\nu(w) := \psi(w) + \varphi_\nu(w).$$

Algorithm 3 Proximal Gradient Descent for Relax and Split (2.12).

1: Choose $s_0, w_0 \in \mathbb{R}^n$ where ψ is finite.

2: **for** $j = 0, 1, \ldots$ **do**

3: Choose $0 < \nu_j$ such that $\nu_j < 1/L$.

4: Update the the proximal-step by computing the point (2.5)

$$w_{j+1} \in \operatorname*{prox}_{\nu_j \psi}(w_j - \nu_j \nabla \varphi_{\nu_j}(s_j))$$

5: Pick $\nu_{j+1} \leq \nu_j$ such that $\nu_j \to 0$.

Technically many algorithms can be used to solve this formulation, especially if ψ is simple. We detail proximal-gradient descent in Algorithm 3 without explict ν updates. These updates can be written explicitly as

$$\operatorname*{prox}_{\nu\psi}(w_j - \nu \nabla \varphi_\nu(w_j)) = \operatorname*{prox}_{\nu\psi}(As_j)$$
$$s_j(w_j) \in \arg\min_s \ \varphi(s) + \frac{1}{2\nu}\|As - w_j\|^2.$$

Solutions for convex φ can be easily established and convergence results exist in [152]. For basis pursuit problems, more analysis is done in [153]. Note that generally, we have to take $\nu \to 0$ to make sure the problems end up being the same. Our work in Chapter 4 extends this method to Morozov-type formulations of basis-pursuit denoise problems, with double splitting.

2.3.4 Block Coordinate Descent

Block coordinate descent is a popular method for minimizing real-valued continuosly differentiable functions of n real variables. These functions can also be subject to bound constraints. This method has the coordinates partitioned into p blocks where ϕ is minimized with respect to one of the coordinate blocks while other coordinates are fixed. In this way, it is similar to Gauss-Seidel or SOR methods for equation solving [139]. Convergence requires strict con-

vexity, quasiconvexity, or hemivariate differentiable functions, along with bounded level sets. [139] relaxes these assumptions; the authors consider the nondifferentiable and nonconvex case where the nondifferentiable part is separable. This takes the form of

$$\phi(s_1, \ldots, s_p) = \phi_0(s_1, \ldots, s_p) + \sum_{i=1}^{p} \phi_i(s_i) \tag{2.13}$$

for some $\phi_0 : \mathbb{R}^{n_1+\ldots+n_p} \to \mathbb{R} \cup \{\infty\}$ and some $\phi_i : \mathbb{R}^{n_k} \to \mathbb{R} \cup \{\infty\}$. This ϕ must also be proper. [139] shows that each cluster point of the iterates generated by BCD is a stationary point of ϕ, provided that ϕ_0 is smooth, ϕ is continuous on a compact set, and either ϕ is pseudoconvex in every pair of coordinate blocks from among the $p-1$, or ϕ has at most one minimum in each of the $p-2$ blocks. Our variable is $\mathbf{s} := [s_0; \ldots; s_p] \in \mathbb{R}^n$ where $n = \sum_{i=0}^{p} n_i$. We let $s_{<i} := [s_0; \ldots s_{i-1}] \in \mathbb{R}^{n_0+n_1+\ldots+n_{i-1}}$ and $s_{>i} := [s_{i+1}; \ldots; s_p] \in \mathbb{R}^{n_{i+1}+\ldots+n_p}$ with $s_{<0}$ and $s_{>p}$ are null variables. Subvectors $s_{\leq i} := [s_{<i}; s_i]$ and $s_{\geq i}$ are defined similarly. This algorithm is given by Note that we use arg min rather vaguely in this context; ϕ can take

Algorithm 4 Block Coordinate Descent.

1: Choose $\mathbf{s} = [s_0, \ldots, s_p] \in \mathrm{dom}(\phi)$.

2: **for** $j = 0, 1, \ldots$ **do**

3: Given $\mathbf{s}_j = [s_{0,j}, \ldots, s_{p,j}] \in \mathrm{dom}(\phi)$, choose index $k \in \{1, \ldots, p\}$.

4: Compute new iterate

$$\mathbf{s}_{k,j+1} \in \arg\min_{s_k} \phi(s_{<k,j}, s_k, s_{>k,j})$$

5: $\mathbf{s}_{i,j+1} = \mathbf{s}_j \ \forall\ i \in \{1, \ldots, p\}, i \neq k$.

on many forms, and in upcoming works we use it as the solution of the proximal operator and direction inversion.

2.3.5 Alternating Direction Method of Multipliers

To round out our preview of nonsmooth algorithms, we touch on the primal-dual method ADMM. The Alternating Direction Method of Multipliers primarily is concerned with prob-

lems of the form

$$\underset{\mathbf{s}\in\mathcal{S},z\in\mathcal{Z}}{\text{minimize}}\,\phi(\mathbf{s},z) := \varphi(\mathbf{s}) + \psi(z) \quad \text{s.t. } \mathbf{A}\mathbf{s} + Bz = c \tag{2.14}$$

or in a multi-block sense

$$\min_{s\in\mathcal{S},z\in\mathcal{Z}} \varphi(s_0, s_1, \ldots, s_p) + \psi(z) \quad \text{s.t. } A_0 s_0 + \ldots + A_p s_p + Bz = c. \tag{2.15}$$

for simple constraints $\mathcal{S}, \mathcal{Z}$ (e.g. bounds). For reference, we refer to the minimum general ADMM algorithm for problems in (2.14) with

$$\phi_* = \underset{\mathbf{s},z}{\text{minimize}}\{\phi := \varphi(\mathbf{s}) + \psi(z) \ |\mathbf{A}\mathbf{s} + Bz - c = 0\}$$

and augmented Lagrangian

$$\mathcal{L}_\rho(\mathbf{s}, z, y) = \varphi(\mathbf{s}) + \psi(y) + y^T(\mathbf{A}\mathbf{s} + Bz - c) + (\rho/2)\|\mathbf{A}\mathbf{s} + Bz - c\|_2^2 \tag{2.16}$$

where $\mathbf{s}, z$ are the primal variables, ρ is the penalty parameter, and y represents the dual variables. From this, we construct the generic ADMM for two blocks Algorithm 5; we denote general iterates as s_k and indices as $s_{i,k}$ (i.e. blocks come first). The final step is

Algorithm 5 Basic ADMM 2-Block splitting

1: **Input:** functions φ, ψ, matrices $\mathbf{A}, B$, vector c, and augmented Lagrangian parameter $\rho > 0$.
2: Choose $\mathbf{s}_0, z_0, y_0$.
3: **for** $k = 0, 1, \ldots$ **do**
4: $\quad \mathbf{s}_{k+1} = \arg\min_{\mathbf{s}} \mathcal{L}_\rho(\mathbf{s}, z_k, y_k)$
5: $\quad z_{k+1} = \arg\min_{z} \mathcal{L}_\rho(\mathbf{s}_{k+1}, z, y_k)$
6: $\quad y_{k+1} = y_k + \rho(\mathbf{A}\mathbf{s}_{k+1} + Bz_{k+1} - c)$

simply a first-order update on the dual variables. However, a slightly more useful way of writing Algorithm 5 is with scaling. Here, we rewrite the feasibility to be $r = \mathbf{A}\mathbf{s} + Bz - c$

and consider a *scaled* dual variable $u = \frac{y}{\rho}$, which enables us to complete the square of the augmented Lagrangian part

$$\begin{aligned} y^T r - \frac{\rho}{2}\|r\|_2^2 &= (\rho/2)\|r + y/\rho\|_2^2 - \frac{1}{2\rho}\|y\|_2^2 \\ &= \frac{\rho}{2}\|r + u\|_2^2 - \frac{\rho}{2}\|u\|_2^2 \end{aligned}$$

and we can express our new algorithm as Algorithm 6 where if we let the feasibility at

Algorithm 6 Scaled, Basic ADMM 2-Block splitting

Input: functions φ, ψ, matrices $\mathbf{A}, B$, vector c, and augmented Lagrangian parameter $\rho > 0$.

Choose $\mathbf{s}_0, y_0, z_0$.

for $k = 0, 1, \ldots$ **do**

$\mathbf{s}_{k+1} = \arg\min_{\mathbf{s}} \varphi(\mathbf{s}) + \frac{\rho}{2}\|\mathbf{A}\mathbf{s} + Bz_k - c + u_k\|^2$

$z_{k+1} = \arg\min_{z} \psi(z) + \frac{\rho}{2}\|\mathbf{A}\mathbf{s}^{k+1} + Bz - c + u_k\|^2$

$u_{k+1} = u_k + \mathbf{A}\mathbf{s}^{k+1} + Bz_{k+1} - c$

iteration k be $r_k = \mathbf{A}\mathbf{s}_k + Bz_k - c$, then $u_k = u_0 + \sum_{j=1}^{k} r_k$ is the running feasibility sum. In general, ADMM converges for convex problems regardless of smoothness. ADMM is a very popular subject of research, and our literature review primarily refers to [29], [148], and [63] for a summary of exactly what types of convergence have been proved.

Convex and Closed

At first, we list the assumptions for established problem classes and their convergence results from [29]. If extended, real-valued functions $\varphi : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$, $\psi : \mathbb{R}^m \to \mathbb{R} \cup \{+\infty\}$ are closed, proper, and convex, then ADMM converges. The split variable updates $\mathbf{s}, z$ are solvable; i.e. there exists $\mathbf{s}, z$ that minimize the Lagrangian but are not necessarily unique (without further assumptions on $\mathbf{A}$ and B). Note that φ, ψ need not be differentiable or smooth, and at this point there are no assumptions on $\mathbf{A}, B$ matrices. If there exists a

$\mathbf{s}_*, z_*, y_*$, not necessarily unique, for which

$$\mathcal{L}_0(\mathbf{s}_*, z_*, y) \leq \mathcal{L}_0(\mathbf{s}_*, z_*, y_*) \leq \mathcal{L}_0(\mathbf{s}, z, y_*)$$

holds for all x, y, z, then we can make a stronger convergence statement:

1. *Feasibility Convergence* - $\lim_{k\to\infty} r_k \to 0$, this means the iterates approach feasibility.

2. *Objective Convergence* - $\lim_{k\to\infty} \varphi(\mathbf{s}_k) + \psi(z_k) \to p_*$ the objective function of the iterates approaches the optimal value.

3. *Dual Variable Convergence* - $\lim_{k\to\infty} y_k \to y_*$, for the dual optimal point y_*.

Note here that these results claim nothing about rate and make strong assumptions on the nature of the problem, but not on the penalty parameter. However, [56] and [21, Chapter 3] both state that $\rho \geq \rho_{\min} > 0$ are required for linear convergence. This is reiterated in [56], where required ρ values are shown graphically for both linear and nonlinear constraints. There are several update schemes available to ρ [29], but its exact value typically remains unknown. Generally speaking, ρ affects the dual feasibility (small ρ) at the expense of the primal feasibility and subsequently the primal feasibility. Taking $\rho \to \infty$ with each subsequent iteration has been analyzed as well as lower bounds on ρ. In fact, most of the summation below provides only lower bounds for the penalty parameter itself.

Nonconvex Block ADMM

In [148], the authors consider Equation (2.15) where ADMM may have multiple blocks. In this paper, the authors make assumptions on the coercivity, feasibility, and Lipschitz continuity of the functions but nothing on convexity or smoothness.

Definition 8 (Coercivity)**.** *Define the feasible set* $\mathcal{F} : \{(\mathbf{s}, z) \in \mathbb{R}^{n+q} : \mathbf{A}\mathbf{s} + Bz = c\}$. *The objective function* $\varphi(\mathbf{s}) + \psi(z)$ *is coercive over this set. This means* $\varphi(\mathbf{s}) + \psi(z) \to \infty$ *if* $(\mathbf{s}, z) \in \mathcal{F}$ *and* $\|(\mathbf{s}, z)\| \to \infty$.

If the feasible set of $(\mathbf{s}, z)$ is bounded, then this assumption holds trivially for any continuous objective function. $\text{Ima}(\mathbf{A}) \subseteq \text{Ima}(B)$ where $\text{Ima}(\cdot)$ returns the image of a matrix. This assumption might be stronger than feasibility since it implies there exists an $\mathbf{s}$ feasible for every z. Algorithm 5 with p blocks converges for sufficiently large ρ (with lower bound based on restricted prox-regularity parameter) and any starting point $(\mathbf{s}, z, y)$. It generates a bounded sequence with at least one limit point which is a stationary point for the augmented Lagrangian (2.16). In [56], the authors derive a minimum $\rho_{\min} > 0$ that depends on the eigenvalues of the Hessian of the objective and the singular values of the Jacobian $\mathbf{A}$. Additionally, if (2.16) is a Kurdyka-Łojasiewicz (KŁ) function (see [10]), then it converges globally to the unique limit point. If the whole function $\varphi(\mathbf{s}) + \psi(y)$ is Lipschitz differentiable, then all of the above still holds. d Wang et al. [148] also give iterate difference convergence results of $\mathcal{O}(1/k)$ and subgradient convergence results of $\{\|d^{k+1}\|\} \sim o(1/\sqrt{k})$ where $d \in \partial\mathcal{L}_\rho$ from (2.16). Also, the variables do not have to be updated in order, similar to block-coordinate-descent.

The authors of [63] extend the nonconvex formulation to also include bilinearly constrained optimization problems in the form of nonnegative matrix factorization. This is a slightly different problem, as it focuses on the problem

$$\min_{X,Y,Z} \phi(X, Y, Z) := \varphi(Z) + \psi_1(X) + \psi_2(Y) \quad \text{s.t. } Z = XY, \tag{2.17}$$

where ψ_i are regularizers or conditions on the matrices X, Y. The authors make several assumptions on the functions described in (2.17):

1. $\varphi : \mathbb{R}^{P\times Q} \to \mathbb{R}$ is smooth (possibly nonconvex), and gradient Lipschitz continuous with constant L.

2. $\phi(X, Y, Z)$ is bounded below; e.g., $\exists\tilde{\phi}$ such that $\phi(X, Y, Z) \leq \tilde{\phi}$ for all $X, Y, Z \in \text{dom}(\phi)$.

3. For $i = 1, 2$, $\psi_i : \mathbb{R}^{P\times K} \to \mathbb{R}$ is either a convex or concave, possibly non-smooth, lower semi-continuous regularizer. These have the form $\psi_i(X) = h_i(l_i(X))$ where l_i

are convex possibly non-smooth functions and h_i is a continuously differentiable and monotone non-decreasing function over the open set containing $\text{dom}(l_i)$.

In later work, we build upon this framework to guide ADMM convergence with Filter Methods.

2.3.6 Quasi-Newton Methods

Another algorithm we frequently use for comparison in this piece is the Broyden-Fletcher-Goldfarb-Shanno algorithm. Unlike perhaps the previous routines, this method is primarily used on unconstrained nonlinear problems. BFGS (or L for limited memory versions) is a first-order method that determines descent directions by preconditioning the gradient with curvature information obtained via generalized secant method approximation of the objective Hessian [106]. This Hessian is low rank, and limited memory applications can influence the number of gradient approximations used in the Hessian approximation. It is known as a computationally cheap method due to the lack of matrix inversion, with $O(n^2)$ operations. Modifications to constraints and projects have also been implemented widely in the literature [124].

Before we discuss Quasi-Newton Methods like LBFGS, we first speak briefly about Newton Methods. Overall, both methods are solving (2.4) with $\psi(x) = 0$. Essentially, this method constructs iterates $\{x_k\}$ that converges towards x_*. Newton methods solve a function $\phi : \mathbb{R}^n \to \mathbb{R}$ with $\phi \in \mathcal{C}^2$ by exploiting the Taylor Series

$$\phi(x_k + s) \approx \phi(x_k) + \nabla\phi(x_k)^T(s) + \tfrac{1}{2}s^T\nabla^2\phi(x_k)s \tag{2.18}$$

where the next iterate minimizes the quadratic approximation in the descent step $s = x_k - x_{k+1}$. Hence, can take the derivative of (2.18) with respect to s to get an update scheme, given by Algorithm 7. Note that there are many extensions of this method that enable linesearch methods, ν choice, etc [106].

A downside of Newton's method is that Hessian computation can be prohibitively expensive for large problems. In BFGS methods, this constraint is negated by only using first

Algorithm 7 Basic Newton's Method.

1: Choose $x_0 \in \mathbb{R}^n$ for $\phi \in \mathcal{C}^2$, stepsize $\nu \in (0, 1]$.
2: **for** $j = 0, 1, \ldots$ **do**
3: $\quad x_{j+1} \leftarrow x_j - \nu \nabla^2 \phi(x_j)^{-1} \nabla \phi(x_j)$

order information to construct Hessian approximations. This makes use of the Sherman-Morrison-Woodbury formula to update the Hessian approximation and take it's inverse. It also utilizes a linesearch method that can be exact or satisfy the Wolf conditions [106]. This routine is given in Algorithm 8.

Algorithm 8 BFGS.

1: Choose $x_0 \in \mathbb{R}^n$ for $\phi \in \mathcal{C}^1$, initial Hessian guess B_0.
2: **for** $j = 0, 1, \ldots$ **do**
3: $\quad d_j \leftarrow -B_k^{-1} \nabla \phi(x_j)$.
4: $\quad$ Perform linesearch to find acceptable step-size ν_j; can be $\arg\min_\nu \phi(x_j + \nu d_j)$, or satisfy Wolfe Conditions.
5: $\quad s_j \leftarrow \nu_j d_j$
6: $\quad x_{j+1} \leftarrow x_j + s_j$
7: $\quad y_j \leftarrow \nabla \phi(x_{j+1}) - \nabla \phi(x_j)$
8: $\quad B_{j+1} = B_j + \frac{y_j y_j^T}{y_j^T s_j} - \frac{B_j s_j s_j^T B_j^T}{s_j^T B_j s_j}$

Note that stopping criteria is typically given by the norm of the gradient being larger than a user-defined value: $\|\nabla \phi(x_j)\|_2 \geq \varepsilon$. It is also very useful to form the inverse Hessian approximation via

$$B_{j+1}^{-1} = B_j^{-1} + \frac{(s_j^T y_j + y_j^T B_j^{-1} y_j)(s_j s_j^T)}{(s_j^T y_j)^2} - \frac{B_j^{-1} y_j s_j^T + s_j y_j^T B_j^{-1}}{s_j^T y_j}.$$

L-BFGS methods compute the descent direction for a fixed number of previous gradient computations. This allows for only local or relevant curvature information to be used, as well

as limiting the memory requirement of storing a potentially large Hessian approximation. We do not list that algorithm here, but essentially the only modification is how B_j^{-1} is computed in Algorithm 8. A fuller description occurs in [106], as well as similar quasi-Newton methods like the SR1/LSR1 methods.

2.3.7 Trust Region Methods

Trust region methods are popular and robust, ensuring that any starting point attains a local minimum. While not as efficient as linesearch strategies [39, 102] (sometimes), they are flexible enough to handle many nonlinear functions that arise in inverse problems. We assume here that $\phi(\cdot)$ is smooth and $\nabla\phi(\cdot)$ are expensive to compute. At the core of trust region methods is the replacement of the expensive cost function

$$\underset{x}{\text{minimize}}\, \phi(x)$$

with the inexpensive quadratic program centered around x_k

$$\underset{s}{\text{minimize}}\, m_k(s) := \phi(x_k) + \nabla\phi(x_k)^T s + \tfrac{1}{2} s^T B_k s \tag{2.19a}$$

$$\text{subject to} \quad \|s\| \leq \Delta_k \tag{2.19b}$$

where B_k is a symmetric matrix approximation of the Hessian $\nabla^2\phi(x_k)$. In some instances, it can be the true Hessian, but these are expensive to compute generally; hence this usually becomes an L-BFGS or LSR1 approximation [40, 102]. The solution of the trust region subproblem provides a candidate for the new trust region center. Trust region methods keep track of the quadratic model performance via

$$\rho_k = \frac{\phi(x_k) - \phi(x_k + s)}{m_k(0) - m_k(s)}$$

which is the ratio of actual to predicted model performance. If the model predicts the actual step in the function well, then x_k is updated and the trust region is either kept or increased. If the quadratic model predicts the actual function decrease poorly, then the step is rejected and the trust region Δ_k is decreased. This behavior is articulated in Algorithm 9.

Algorithm 9 Basic Trust-Region Algorithm.

Choose constants

$$0 < \eta_1 \leq \eta_2 < 1, \quad 0 < \gamma_1 \leq \gamma_2 < 1 < \gamma_3 \leq \gamma_4$$

Choose $x_0 \in \mathbb{R}^n$ where ϕ is finite, $\Delta_0 > 0$, compute $\phi(x_0)$.

for $k = 0, 1, \ldots$ **do**

Define $m_k(s; x_k)$ with norm $\|\cdot\|$ and continuously differentiable/L-smooth that matching $\phi(\cdot)$ at x_k.

Compute an approximate solution s_k of (2.19) with model $m_k(s; x_k)$ that attains sufficient decrease [40, 6.3.1].

Compute the ratio

$$\rho_k := \frac{\phi(x_k) - (\phi(x_k + s_k))}{m_k(0) - m_k(s_k)}.$$

If $\rho_k \geq \eta_1$, set $x_{k+1} = x_k + s_k$. Otherwise, set $x_{k+1} = x_k$.

Update the trust-region radius according to

$$\Delta_{k+1} \in \begin{cases} [\gamma_3 \Delta_k, \gamma_4 \Delta_k] & \text{if } \rho_k \geq \eta_2, \\ [\gamma_2 \Delta_k, \Delta_k] & \text{if } \eta_1 \leq \rho_k < \eta_2, \\ [\gamma_1 \Delta_k, \gamma_2 \Delta_k] & \text{if } \rho_k < \eta_1. \end{cases}$$

It can be shown [40] that the sequence of points that are trust-region centers converges to a first-order critical point

$$\lim_{k\to\infty} \|\nabla\phi(x_k)\| = 0.$$

[102] show that fractional Cauchy decrease produces convergence. This has opened up a rich vein of research that has led to many innovations seen in [36, 40–42, 59]. However, limited work has been done for nonsmooth trust region algorithms. [36] look at composite trust region optimization, but rely on at least some smoothness in the problem. [40] describe a (partially) nonsmooth interpretation but still require at least Lipschitz continuity. More literature review is completed on this particular topic when we approach it later in Chapter 5. With the majority of the basic algorithmic building blocks covered, we can now proceed to our first objective: developing relax and split techniques for regularized inverse problems.

Chapter 3

RELAXATION ALGORITHMS FOR MATRIX COMPLETION, WITH APPLICATIONS TO SEISMIC TRAVEL-TIME DATA INTERPOLATION

3.1 Introduction & Problem Motivation

Travel-time tomography is used to determine the underlying structure of the earth and how seismic waves propagate through that structure. This weakly nonlinear problem is formulated as a data-fitting inverse problem and solved using iterative optimization techniques. Data quality and availability are key constraints and can drastically influence the merit of the results [83, 84, 112]. Hence, researchers often use prior information to denoise and interpolate the data prior to inversion. Parsimonious representations [33, 117] of the data in transform domains such as Fourier [122] and Curvelet [69] have been used in exploration seismology [66, 81], along with low-rank representations [7, 44].

We focus on regional seismology — in particular, we want to analyze data obtained by the Imaging Magma Under St. Helens (iMUSH) project, a multi-year effort to image and infer the architecture of the greater Mount St. Helens, WA, magmatic system [76, 141]. The iMUSH project uses a variety of geophysical and petrological methods, including active source, local earthquake, and ambient noise seismic tomography to study the most active volcano in the Cascades arc. For the passive source seismic portion of the iMUSH project, 70 broadband seismometers were deployed from 2014 to 2016 within a 100km diameter circle around the mountain; these have an average station spacing of 10km, and are supplemented by permanent stations maintained by the Pacific Northwest Seismic Network (PNSN) and a temporary array of 20 broadband seismometers deployed by AltaRock Energy in June to November of 2016.

The data collected from this experiment comprises P-wave travel times from 23 active borehole explosions [77] and over 400 local earthquakes [141] recorded at the iMUSH broadband array. Collected data has a range of fidelities, and different subsets inform different parameters for 3D P-wave velocities and hence geometries of the structure underlying Mount St. Helens. Travel times are inverted to obtain 3D seismic velocity models and image complex subsurface structures, including low velocity zones. This is standard practice in seismology and much has been done in the literature [68, 131] to explore this facet of the problem.

The efficacy of inverse problems depends on the quality of the data, which is affected by noise from roads, streams, ocean waves, and wind. The signal to noise ratio (SNR) of the data depends on the source size, distance from the seismometer, and attenuation structure of the earth. Data is scarce because many landscape features are impassable; typical station spacing is 10km. Arrival times are chosen by operators, who assign uncertainties for particular observations based on confidence in seismic readings. Low-magnitude events in particular are often difficult to distinguish from noise. Installing more sensors may overcome data sparsity problems as well as some noise issues; however, the issue of impassable terrain remains, as well as sensor placement. In addition, the iMUSH project has been completed, so no more sensors can be installed or used (outside of the permanent ones). Running the inverse problem at a higher resolution may resolve some resolution issues, but that does not necessarily mean results are correct in some locations. Usually a checkerboard test is done to see where exactly the model has good resolution versus bad resolution by examining how sensitive it is to perturbations, but this does not help in resolving the areas of low-resolution. Running the model may also result in needless computational expense; `struct3DP` takes several days to solve the standard problem.

Our goal here is to increase the raypath coverage for use in earthquake tomography by simulating seismometer locations and observations by interpolating noisy data. We plan to do this via an optimization problem that utilizes low-rank matrix completion and denoising. The interpolating and smoothing results for generated sensors would then be used to run a standard inverse problem to possibly better clarity; this is left for future work.

Our approach is particularly useful for areas within the study region that did not have seismometers installed or which had seismometers out of service for extended periods of time. Low-rank matrix completion and denoising is attractive because it allows the enunciation of complex data features with a few components. For example, [67, 96] show that wavelet and curvelet transforms can be expressed with few principal components, thereby lending missing/noisy data to be re-expressed with linear combinations of known/trusted data. Correct matricization of data can naturally lead to low-rank interpolation schemes [7, 44], as

storing data in matrix form phrases it in the context of a linear basis. Since we know the data is corrupted by uncertainty and noise, the problem is a good match for level-set optimization formulations [8] that minimize a regularizer subject to a prescribed level of data fit. We propose a relaxation formulation that allows misfit constraints while (1) penalizing rank of a tessellation that captures redundancy of features across sources, and (2) enforcing smooth features consistent with the underlying physical model. We develop an efficient block-coordinate algorithm for this formulation, and compare the approach with a variety of competing formulations and algorithms.

This chapter proceeds as follows. In section 3.2 we review relevant formulations and algorithms for interpolation and denoising. In section 3.3, we develop the extended model formulation and the relaxation method to solve it. Section 3.4 describes the data used and how it fits into a low-rank interpolation scheme. In section 3.5 we evaluate the approach and compare it against alternatives for denoising and interpolating Mount St. Helens data.

3.2 Seismic Notational Preliminaries and Data Formulation

In this section, we set up the notation, review prior information used in interpolation and denoising, and discuss different types of standard optimization formulations needed to implement such approaches.

3.2.1 Notation

We use the following notation conventions this section. The terms (R_x, R_y) represent a receiver coordinate grid. The variables $n_d, n_{R_x}, n_{R_y}, n_s$ represent the number of observed data points, the number of points in the x-direction of the receiver grid, the number of points in the y-direction of the receiver grid, and the number of sources. The variable Ω is used to represent the entire 4-dimensional source/receiver space and generically the interpolation space.

3.2.2 Formulations for Low-rank Interpolation

The goal of low-rank matrix completion is to accurately estimate the missing data entries of a matrix $X \in \mathbb{R}^{m\times n}$ from observed entries. Low-rank structure is often inferred if completed X has few non-zero singular values and experiences entry repetition. The observed entries are given by the vector $b \in \mathbb{R}^{n_d}$. We let $\mathcal{A} : \mathbb{R}^{n\times m} \to \mathbb{R}^{n_d}$ be the restriction/interpolation operator that maps elements from the regular grid X to the observed values b, which can be written as $b = \mathcal{A}(X) + \epsilon$ for $\epsilon \in \mathbb{R}^{n_d}$. Here, ϵ represents the data corruption of observed entries via some noise distribution. In formulating the problem, the nuclear norm

$$\|X\|_* = \sum_{j=1}^{\min(n,m)} \hat{\sigma}_j(X),$$

with $\hat{\sigma}_j$ the singular values of X, is used as a proxy for rank. The classic formulations that balance data fit with regularization are

$$\min_{X\in\mathbb{R}^{n\times m}} \|X\|_* + \tfrac{1}{\sigma}\|\mathcal{A}(X) - b\|_2 \tag{3.1}$$

$$\min_{X\in\mathbb{R}^{n\times m}} \quad \|\mathcal{A}(X) - b\|_2 \quad \text{subject to} \quad \|X\|_* \le \tau \tag{3.2}$$

$$\boxed{\min_{X\in\mathbb{R}^{n\times m}} \|X\|_* \quad \text{subject to} \quad \|\mathcal{A}(X) - b\|_2 \le \sigma}. \tag{3.3}$$

These formulations are known as Tichonoff, Ivanov, and Morozov regularization [108], respectively. The misfit-constraint Morozov variant (3.3) is best suited for situations where a good estimate of the uncertainty, σ, is available. To simplify exposition, we will focus on Morozov-type formulations.

3.2.3 Smoothness constraints

In travel time tomography, smoothness and continuity between gridpoints is a reasonable prior, since the geological structure of the crust exhibits the traits of approximately homogeneous media. Smoothness is enforced by introducing a penalty term $\|\mathcal{L}(X)\|_2^2$, with $\mathcal{L}$ the discretization of the Laplacian operator. In 1D, it is a tridiagonal difference matrix operating on the vectorized X; in 2D, it has four off-diagonal elements.

Enforcing smoothness and low-rank structure combines local and global information. A Morozov formulation, with γ balancing the regularizers, is given by

$$\min_X \|X\|_* + \frac{1}{2\gamma}\|\mathcal{L}(X)\|_2^2 \quad \text{subject to} \quad \|\mathcal{A}(X) - b\|_2 \leq \sigma. \tag{3.4}$$

3.2.4 Factorized Formulations

Theoretical properties of matrix completion via nuclear-norm minimization have been extensively studied [31, 32]. The theoretical appeal of convex formulations is tempered by computational considerations — algorithms that optimize $\|X\|_*$ require a full matrix decision variable, and full or partial singular value decompositions (SVDs) at each iteration. An efficient alternative is to use matrix factorization formulations [7], writing $X = LR^T$, with $L \in \mathbb{R}^{n\times k}, R \in \mathbb{R}^{m\times k}$. From [117], we have the characterization

$$\|X\|_* = \inf_{L,R:X=LR^T} \frac{1}{2}(\|L\|_F^2 + \|R\|_F^2),$$

which allows us to replace $\|X\|_*$ in any formulation by $\frac{1}{2}(\|L\|_F^2 + \|R\|_F^2)$ for any factorization $X = LR^T$. For example, the three formulations (3.1)-(3.3) become

$$\min\nolimits_{L\in\mathbb{R}^{n\times k},\ R\in\mathbb{R}^{m\times k}} \tfrac{1}{2}\|L\|_F^2 + \tfrac{1}{2}\|R\|_F^2 + \tfrac{1}{\sigma}\|\mathcal{A}(LR^T) - b\|_2 \tag{3.5}$$

$$\min\nolimits_{L\in\mathbb{R}^{n\times k},\ R\in\mathbb{R}^{m\times k}} \quad \|\mathcal{A}(LR^T) - b\|^2 \quad \text{subject to} \quad \|L\|_F^2 + \|R\|_F^2 \leq 2\tau \tag{3.6}$$

$$\boxed{\min_{L\in\mathbb{R}^{n\times k},\ R\in\mathbb{R}^{m\times k}} \frac{1}{2}\|L\|_F^2 + \frac{1}{2}\|R\|_F^2 \quad \text{subject to} \quad \|\mathcal{A}(LR^T) - b\|_2 \leq \sigma}, \tag{3.7}$$

where $k \ll \min(n, m)$, and the memory requirements are reduced from mn to $k(n + m)$. No SVDs are required; formulation (3.5) is smooth, formulation (3.6) requires simple projections onto the Frobenius-norm ball, and formulation (3.7) can be solved using (3.6) via root-finding as described by [7].

Our primary technical goal here is to solve the factorized Morozov formulation corresponding to (3.4):

$$\min_{L,R} \frac{1}{2}\|L\|_F^2 + \frac{1}{2}\|R\|_F^2 + \frac{1}{2\gamma}\|\mathcal{L}(LR^T)\|_2^2 \quad \text{s.t.} \quad \|\mathcal{A}(LR^T) - b\|_2 \leq \sigma. \tag{3.8}$$

In particular, this formulation incorporates both local and global structure, and gives a misfit target σ. It requires a new algorithm, since (3.8) cannot be solved by the level-set approach of [7]. The only available alternative is to use Tichonoff-type formulations (see section 3.3.1). Our main technical contribution is a nonconvex splitting algorithm for problem (3.8), developed in the next section.

3.3 Relaxed Joint Inversion

The main challenge of the factorized Morozov formulation (3.8) is the data-misfit constraint. To solve the problem, we propose a *relaxation* following the ideas of [152]. In particular, we introduce an auxiliary variable $W \approx LR^T$:

$$\min_{L,R,W} \frac{1}{2}\|L\|_F^2 + \frac{1}{2}\|R\|_F^2 + \frac{1}{2\gamma}\|\mathcal{L}(W)\|_2^2 + \frac{1}{2\eta}\|W - LR^T\|_F^2 \quad \text{s.t.} \quad \|\mathcal{A}(W) - b\|_2 \leq \sigma. \tag{3.9}$$

Problem (3.9) is a relaxation for problem (3.8), since W approximates $X = LR^T$; in particular $\|W - LR^T\| = \mathcal{O}(\eta)$. The salient modeling features of (3.8) are still preserved. We can now design a simple block-coordinate descent algorithm by iteratively optimizing in each of (L, R, W), detailed in Algorithm 10, which is a version of Algorithm 4.

Algorithm 10 Block-Coordinate Descent for (3.9).

1: **Input:** W_0, L_0, R_0

2: Initialize: $i = 0$.

3: **while** not converged **do**

4: $\quad L_{i+1} \leftarrow \left(I + \eta R_i^T R_i\right)^{-1} (\eta R_i^T W_i^T)$ ▷ Solves $\min_L \frac{1}{2}\|L\|_F^2 + \frac{1}{2\eta}\|W_i - LR_i^T\|_F^2$

5: $\quad R_{i+1} \leftarrow (\eta W_i^T L_{i+1}) \left(I + \eta L_{i+1}^T L_{i+1}\right)^{-1}$ ▷ Solves $\min_R \frac{1}{2}\|R\|_F^2 + \frac{1}{2\eta}\|W_i - L_{i+1}R^T\|_F^2$

6: $\quad W_{i+1} \leftarrow \arg\min_W \frac{1}{2\gamma}\|\mathcal{L}(W)\|_2^2 + \frac{1}{2\eta}\|W - L_{i+1}R_{i+1}^T\|_F^2 \quad \text{s.t.} \quad \|\mathcal{A}(W) - b\|_2 \leq \sigma$

7: $\quad i \leftarrow i + 1$

8: **Output:** W_i, L_i, R_i

Steps 4 and 5 of Algorithm 10 are simple least squares updates; each minimizes (3.9) in

L and R respectively, with the remaining variables held fixed. Algorithm 10 converges to a stationary point of (3.9) by [139, Theorem 4.1]. In particular, $f(L, R, W)$ has a unique minimum in each coordinate block with the remaining blocks held fixed, which satisfies condition (c) of the theorem. The uniqueness of the minima are clear from the closed form solutions in steps 4 and 5, and from the strong convexity of the W subproblem in step 6. This step 6 is solved using an efficient root-finding method. First, we describe the equivalent penalized problem with penalty parameter λ, given by

$$w(\lambda) := \arg\min_{w} \frac{1}{2\gamma}\|L_\nabla w\|_2^2 + \frac{1}{2\eta}\|w - d_{i+1}\|_F^2 + \frac{\lambda}{2}\|Aw - b\|^2, \tag{3.10}$$

where $w = \mathrm{vec}(W)$, L_∇ is a sparse matrix that encodes the action of the Laplacian on w, A is a linear operator that sends w to b, and $d_{i+1} = \mathrm{vec}(L_{i+1}R_{i+1}^T)$. The root finding method obtains the smallest value of λ satisfying $\|\mathcal{A}w(\lambda) - b\|_2 \leq \sigma$.

Taking the gradient of the objective defining $w(\lambda)$ in (3.10) and setting it equal to 0, we find an explicit formula

$$w(\lambda) = \left(\frac{1}{\gamma}(L_\nabla^T L_\nabla) + \frac{1}{\eta}I + \lambda A^T A\right)^{-1}\left(\frac{1}{\eta}d_{i+1} + \lambda A^T b\right). \tag{3.11}$$

We need only find the smallest $\lambda \geq 0$ so that $\|Aw(\lambda) - b\|_2 = \sigma$. The special case $\lambda = 0$ occurs when the constraint is satisfied at the least squares solution $w(0)$ in (3.11):

$$\left\| A\left(\frac{1}{\gamma}(L_\nabla^T L_\nabla) + \frac{1}{\eta}I\right)^{-1}\left(\frac{1}{\eta}d_{i+1}\right) - b\right\| \leq \sigma,$$

In all other cases, we have

$$f(\lambda) := \sigma - \|Aw(\lambda) - b\|_2, \quad f'(\lambda) = -\frac{\langle A^T Aw(\lambda) - b, \nabla_\lambda w(\lambda)\rangle.}{\|Aw(\lambda) - b\|_2}$$

We compute the quantity $\nabla_\lambda w(\lambda)$ required to evaluate $f'(\lambda)$ using the complex step method [99], instead of differentiating (3.11) directly. The root-finding update is given by

$$\lambda^+ := \lambda - \frac{f(\lambda)}{f'(\lambda).}$$

The expensive step (3.11) is implemented using Cholesky factors of the sparse matrix

$$\left(\frac{1}{\gamma}(L_\nabla^T L_\nabla) + \frac{1}{\eta}I + \lambda A^T A\right).$$

This system only changes when η is updated. Updating W in Algorithm 10 has a relatively high computational cost. Table 3.1 gives a detailed arithmetic complexity analysis, with $L \in \mathbb{R}^{n\times k}$, $R \in \mathbb{R}^{m\times k}$, $W \in \mathbb{R}^{n\times m}$. For our data, $n = m$ and W is square.

Table 3.1: Numerical complexity for Algorithm 10. Key steps are Cholesky factorization (CF), back-substitution (BS), and root finding.

Line	$\mathcal{O}(\cdot)$	Explanation
Line 4	$\frac{k^3}{3} + 2k^2 m$	CF of $k \times k$ matrix with BS of $k \times m$ matrix
Line 5	$\frac{k^3}{3} + 2k^2 n$	CF of $k \times k$ matrix with BS of $k \times n$ matrix
Line 6 $\sigma = 0$	$\frac{(nm)^3}{3} + 2(nm)^2$	CF of $nm \times nm$ matrix and BS
Line 6 $\sigma > 0$	$\left(k_r(2k_p) + 1 + k_l\right)(nm)^2$	Rootfinding algorithm with $k_l, k_p, k_r \leq 100$.

Line 6 of Algorithm 10 requires a root-finding algorithm or another Cholesky decomposition, depending on the value selected for σ. If $\sigma = 0$, then we have the Cholesky decomposition of $\left(\frac{1}{\gamma}(L_\nabla^T L_\nabla) + \frac{1}{\eta}I + \lambda A^T A\right)$ from (3.11) which is order $\frac{(nm)^3}{3} + 2(nm)^2$; the update is required when η or λ change. If $\sigma > 0$, then we implement the root finding algorithm with inputs $L_\nabla^T L_\nabla \in \mathbb{R}^{nm\times nm}$, $L_\nabla A^T b = q \in \mathbb{R}^{nm}$, and $A \in \mathbb{R}^{n_d \times nm}$. This requires three steps. The first is matrix-vector multiply $L_\nabla^T q$. Next, we apply LSQR iterations of order $\mathcal{O}(nm)^2$ where $k_l \leq 100$ is the number of LSQR iterations. Finally, we use k_r iterations of PCG for $k_p \leq 100$ iterations. Here, the major cost is $\mathcal{O}(nm)^2$. Hence, in the total root finding algorithm with k_r iterations, we have one mat-vec, k_l LSQR, and k_p PCG iterations, with $k_l, k_p, k_r \leq 100$.

3.3.1 Alternative Approaches for Low-Rank & Smooth Inversion

There are alternative ways to model problem (3.9). We consider two formulations and algorithms to compare with Algorithm 10.

Nuclear-norm formulation using FISTA. A simple convex formulation that uses smoothness, data misfit, and a rank proxy (nuclear norm) is given by

$$\min_X \frac{\lambda}{2}\|\mathcal{A}(X) - b\|^2 + \frac{1}{2\gamma}\|\mathcal{L}(X)\|^2 + \|X\|_* \tag{3.12}$$

Formulation (3.12) is the sum of a smooth and a simple function, and can be solved using projected gradient or Fast Iterative Shrinkage-Thresholding Algorithm (FISTA) [18], detailed in Algorithm 11 and based on Algorithm 2. This class of algorithms can be viewed as an extension of the classical gradient algorithm and is attractive due to its simplicity. The step size α is the reciprocal of the largest singular value of $(\lambda\mathcal{A}^*\mathcal{A} + \gamma^{-1}\mathcal{L}^*\mathcal{L})$, and the operator S_α is the soft-thresholding operator:

$$S_\alpha(\Sigma)_{jj} = \max(0, \Sigma_{jj} - \alpha).$$

Algorithm 11 FISTA for (3.12).

Input: $X^0 = X^{-1} \in \mathbb{R}^{m\times n}$, $t^0 = t^{-1} = 1$
Initialize: $i = 0$
while not converged **do**
  $Y_i \leftarrow X_i + \frac{t_{i-1}-1}{t_i}(X_i - X_{i-1})$
  $G_i \leftarrow Y_i - \alpha\left((\lambda\mathcal{A}^*\mathcal{A} + \gamma^{-1}\mathcal{L}^*\mathcal{L})\text{vec}(Y_i) - \lambda\mathcal{A}^*b\right)$
  $U, \Sigma, V^T \leftarrow \texttt{svd}(G_i)$
  $X_{i+1} \leftarrow US_\alpha(\Sigma)V^T$
  $t_{i+1} \leftarrow \frac{1+\sqrt{1+4(t^i)^2}}{2}$
  $i \leftarrow i + 1$
Output: X_i

Algorithm 11 uses gradients of the smooth terms, which requires appying $\mathcal{A}$, $\mathcal{L}$ and their adjoints, and the prox operator of $\|\cdot\|_*$, which requires thresholding on singular values computed via SVD (steps 6,7). The most expensive computational step here is the `svd`, which requires $\mathcal{O}(2mn^2+2n^3)$ arithmetic operations [136]. These steps become prohibitively expensive as the dimensions of X grow.

Smooth Factorized Formulation with L-BFGS. To avoid the SVD steps of Algorithm 11, we use the factorization strategy described in section 3.2.4:

$$\min_{L,R} \frac{\lambda}{2}\|\mathcal{A}(LR^T)-b\|^2+\frac{1}{2\gamma}\|\mathcal{L}(LR^T)\|^2+\frac{1}{2}\|L\|_F^2+\frac{1}{2}\|R\|_F^2. \tag{3.13}$$

Formulation (3.13) is smooth with respect to the decision variables L and R, and at larger scales, the limited memory BFGS (L-BFGS) [106] algorithm is a reasonable choice. The number of operations per iteration is $\mathcal{O}(k_L nm)$, with k_L the L-BFGS history size.

Convergence Complexity An apples to apples complexity analysis is difficult for the algorithms presented. We have no complexity result for Algorithm 10; only a convergence guarantee (eventual stationarity of iterates). The accelerated proximal gradient method used for matrix completion in [134] has a rate of convergence $\mathcal{O}\left(1/\sqrt{\varepsilon}\right)$ or $\mathcal{O}\left(\sqrt{\frac{L_f}{k}}\right)$, where L_f is the Lipschitz constant of the least squares component of formulation (3.12). However, this section addresses the *exact* problem, while FISTA and L-BFGS are solving relaxations. In the context of our application, Algorithm 10 converges in an order of magnitude fewer iterations to a higher root-mean-squared error (for both observed and unobserved). For example, this is 2000 iterations of FISTA relative to our 200 iterations (in Algorithm 10, both $\sigma > 0$ and $\sigma = 0$), see section 3.5.

In the next section, we describe the travel-time interpolation problem for regional seismology, evaluate low rank and smooth regularization, and compare the performance of Algorithm 10 for (3.9) to those of FISTA (Algorithm 11) on (3.12) and L-BFGS on (3.13), in terms of computational efficiency and quality of reconstruction.

3.4 Interpolation of synthetic travel time iMUSH data

This experiment interpolates *synthetic* travel time residuals, which are calculated with respect to travel times predicted by a 1D velocity model. Travel times vary on the order of tens of seconds, while travel time residuals are generally between -1 and 1 seconds. We obtain synthetic travel times by using the forward modeling portion of the `struct3DP` code, which uses a finite difference 3D eikonal equation solver [71, 146]. These travel times are calculated for the best-fit 3D model from the iMUSH local earthquake tomography and compared to travel times through the PNSN S4 [97] 1D velocity model to obtain the synthetic residual. Similarly, for observed travel times, we subtract the travel time predicted through the 1D model from the observed travel time to obtain the observed residual.

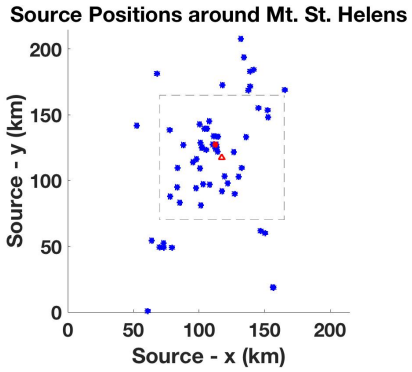
Here, we define the experimental data. While raw iMUSH/PNSN data exists at station locations around the mountain (see blue dots in Figure 3.1(b)), these iMUSH/PNSN stations are not gridded. To deal with this problem, we include an interpolation operator into the standard linear map $\mathcal{A}$, as discussed in detail in section 3.6. In the following synthetic experiments, we solve a forward problem with the 3D eikonal equation solver and use the best-fit 3D model to generate synthetic results for uniformly gridded stations in a square of 70km to 165km (with origin being at latitute 45.2, longitude -123.7). For the rest of this paper, we refer to the travel-time residuals between these synthetic travel times and those predicted by the 1D PNSN S4 model as the *true* data, or X_{true}. To generate observations for the interpolation schemes, we subsample this data down to 15% of total grid coverage over all sources by picking synthetic gridded stations near raw iMUSH/PNSN stations. This is a subset of synthetic stations meant to represent the distribution of the real-world stations that recorded the event. This subsampled data is then corrupted with noise. For each station, we generate a standard deviation parameter from the uniform distribution (0.03-0.15)(s); this captures each synthetic station's inherent uncertainty. The deviation range is based on raw iMUSH uncertainties recorded over the two year period. Then, for each station's data, we add zero mean random gaussian noise generated using that station's deviation parameter to

create the *observed* data, or X_{obs}. Unobserved entries of X_{obs} are zero, while observed entries are given by $\mathcal{A}(X_{obs}) = b$. The operator $\mathcal{A}(\cdot) : \mathbb{R}^{n\times m} \to \mathbb{R}^{n_d}$ takes the tensor we wish to interpolate, X, and selects the observed entries to return a vector the same size as $b \in \mathbb{R}^{n_d}$. This is represented by a binary matrix $A \in \mathbb{R}^{n_d \times nm}$ that selects the observed entries out of a vectorized X.

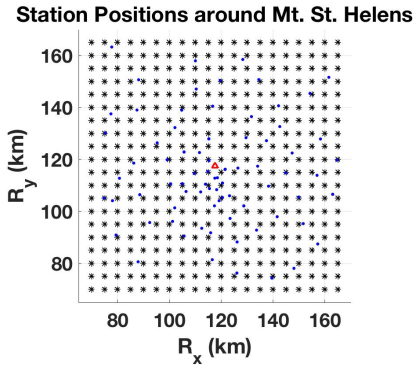
3.4.1 Description of source tensor construction

In order to apply the methods of section 3.3, we have to specify a matricization of the data. The matricization we use is derived from the receiver grid, which is represented as a 95km-wide mesh with 5 kilometer spacing centered near Mount St. Helens. Again, the station grid is centered near the mountain and roughly coincides with the 70 stations deployed for 2 years. Each entry in the matrix represents a point on this uniform receiver grid. Missing entries are designated by zeros, while observed entries are represented with the travel time residuals relative to the 1D model. Our experiments in this paper focus mainly on synthetic residuals of the nonlinear 3D modeled data relative to the 1D model, which provide a 'ground truth' dataset we use to evaluate and compare interpolation techniques.

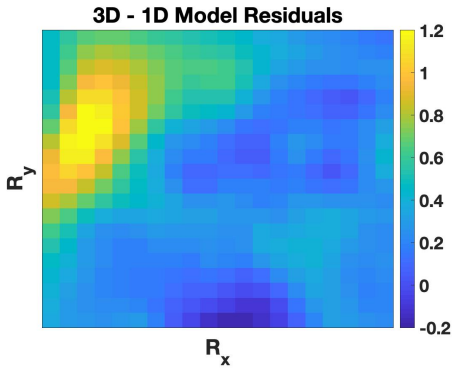
Each source represents a wave moving through the same media. The underlying physical model suggests enforcing smoothness between station gridpoints. We further improve interpolation with low-rank methods by finding a tessellation of source-receiver grid matrices in which full data exhibits fast decay of singular values, while subsampled data does not. Intuitively, such a tessellation reflects the redundancy of features across sources. Our combined goal is to interpolate X from a subset of observations by penalizing rank across sources, and nonsmooth local features. We parameterize the time picks into 4-tensors Ω_{ijkl} where (i, j) are the receiver index pairs and k, l are the source index pairs. The observed residual data is recorded in this 4D tensor format with dimension $(i, j, k, l) \in (1 \dots n_{R_x}, 1 \dots n_{R_y}, 1 \dots \sqrt{n_s}, 1 \dots \sqrt{n_s})$, where $n_{R_x} = 20, n_{R_y} = 20, n_s = 64$ for the experiments. Each source (k, l) has an associated receiver grid of observations $(R_x, R_y) \in (70, 165\text{km}) \times (70, 165\text{km})$ wherein each receiver coordinate pair (i, j) lies. Each receiver axis



(a) Spatial locations of the sources ('*') around Mount St. Helens (Δ). The box represents the edges of the uniform 20×20 station grid. The grid panel (c) is taken from the source marked with a red '*'.

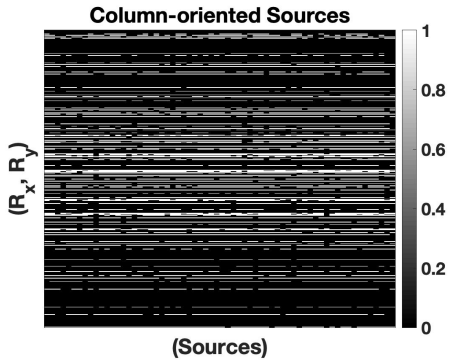


(b) Receiver grid for a single source with the most datapoints. Black '*' represent synthetic stations on a grid, while blue '.' represent iMUSH/PNSN stations off-grid.

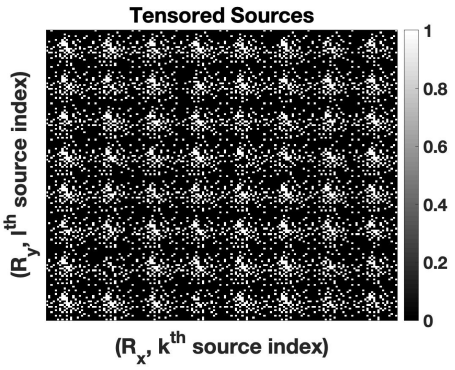


(c) *True* residual data (in seconds) for the receiver grid in panel (b).

Figure 3.1: Data information for synthetic test.



(a) Subsampled matricization Form 1 $X_{(n_{R_x}-1)i+j,(\sqrt{n_s}-1)k+l}$; the missing data (zeros) go across the rows.



(b) Subsampled matricization Form 2 $X_{i+(n_{R_x}-1)k,j+(n_{R_y}-1)l}$; missing entries (zeros) are are interwoven throughout the matrix.

Figure 3.2: Different tensor formulations for low-rank interpolation.

lies between 70km and 165km with 5km spacing. The observation grid was chosen to lie close to the mountain and to contain a relatively large number of sensors (see Figure 3.1(b)). Initially, we observed 64 sources, where each source recorded at most 80 receivers (out of the potential 400). Source locations for the 64 sources are shown in Figure 3.1(a), and a sample receiver grid for a particular source is given in Figure 3.1(b) and Figure 3.1(c) is the *true* data for that grid. Since observed residual data is a tensor, we have to choose a matricization to exploit the induced low-rank structure.

We consider two such matricizations: Form 1 $X_{(n_{R_x}-1)i+j,(\sqrt{n_s}-1)k+l} = \Omega_{ijkl}$, where we group receivers along columns and sources along rows, and Form 2 $X_{i+(n_{R_x}-1)k,j+(n_{R_y}-1)l} = \Omega_{ijkl}$, where each receiver grid is block-inserted into the underlying matrix. On a 20x20 receiver grid with 64 sources, the first matricization $X \in \mathbb{R}^{400\times 64}$, depicted in Figure 3.2(a), is obtained by letting $\Omega_{ij11} \in \mathbb{R}^{R_x,R_y} = \mathbb{R}^{400\times 1}$ be the vectorized receiver grid for single source $k = l = 1$; the sources are then arranged column-wise for $k, l = 1, \ldots, \sqrt{n_s}$. The second matricization $X \in \mathbb{R}^{160\times 160}$, depicted in Figure 3.2(b), is obtained by letting $\Omega_{ij11} \in \mathbb{R}^{R_x\times R_y} = \mathbb{R}^{20\times 20}$ be the nested receiver grid for single source $k = l = 1$. The sources are

then nested together for $k, l = 1, \ldots, \sqrt{n_s}$.

The subsampling scheme is crucial for the approach. The ideal situation is for the subsampled data to have high rank (slow decay of singular values), while the full data has low rank (fast decay of singular values). Then, it is possible to recover the full volume by penalizing rank while matching observed data. For the low-rank penalty to be effective, the subsampled data must have high rank; otherwise, the low-rank penalty will have no effect on the problem. That is, the matricized X_{obs} (plotted in Figure 3.2(b)) should have a higher rank than the fully sampled matricized volume we want to recover. The difference between tensor respresentations in Figure 3.2 is that the columns of Form 1 (Figure 3.2(a)) have high mutual coherence, while the columns of Form 2 (Figure 3.2(b)) have low mutual coherence. The coherence or mutual coherence of a matrix is defined as the maximum absolute value of the cross-correlations between the columns of that matrix. According to Theorem 1.3 in [32], low coherence implies that few entries of X are required for recovery of the full matrix. Hence, we require that X_{obs} have high rank, and the tensor representation plays an important role in the success of the algorthim. The first matricization does not satisfy this simple requirement: the subsampled matrix is itself low-rank and has rows and columns made up of zeros. Therefore we use the second matricization with tessellated sources, which indeed satisfies the requirement, see section 3.4.1.

In both matricizations, the sources are organized from largest in magnitude to smallest in magnitude. In Form 1/Figure 3.2(a), the sources are arranged such that the source with the greatest absolute residual values is in the first column and the least absolute travel-time residuals is in the last. In Form 2/Figure 3.2(b), the source with the greatest energy is in the top left, and the least in the bottom right; the energy then decays by source column (i.e. the next highest energy is the source immediately below in the row of sources). This was done to promote an even slower decrease in singular values for the observed matrix, and makes the low-rank approach far more effective.

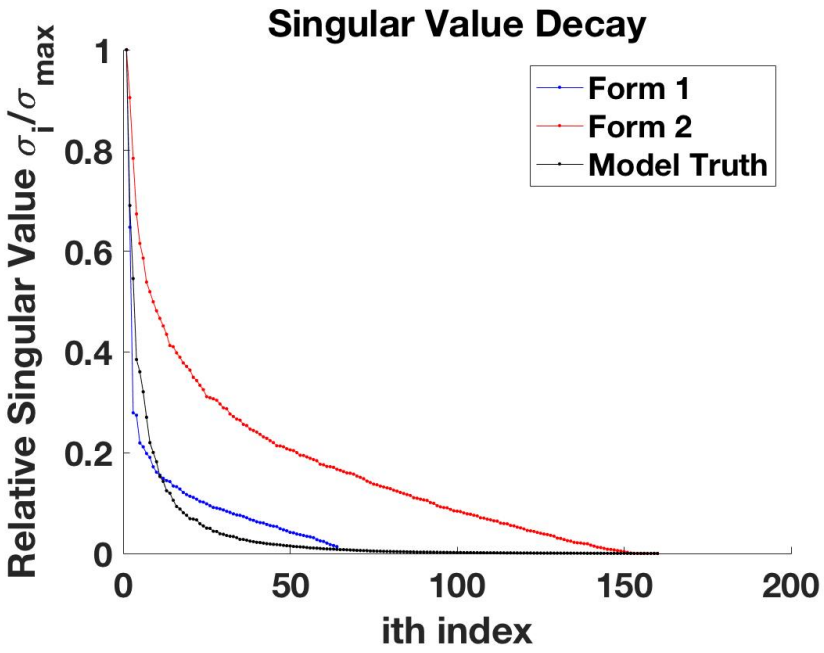


Figure 3.3: Singular value decay for matricization formulations 1 and 2 of the interpolation tensor.

3.5 Application to Synthetic Data

First, we evaluate the accuracy of using both smoothness and low-rank, compared to using either property alone. Then, we compare the speed and robustness of the new algorithm with competitors: FISTA (Algorithm 11, solving (3.12)) and L-BFGS (solving (3.13)). All tests use a tensored grid with the number of sources $n_s = 64$, and grid sizes $n_{R_x}, n_{R_y} = 20$ for $R_x, R_y \in (70, 165)$ evenly spaced at 5km. For all plots, north is up. The subsampling rate for every test is approximatestely 15%. We set the value k (for the LR^T formulation) to be 40 for all algorithms. The value of k was chosen to be significantly smaller than the total number of sources; it is 1/4th of the total number of columns in our experiment. A systemic method for chosing k is currently a driving question within the community; in practice, one can choose a larger k, if there is sufficient computational budget, since penalizing $\|L\|_F^2 + \|R\|_F^2$ controls

the nuclear norm (and hence rank) through the relationship

$$\|LR\|_* \leq \frac{1}{2}\|L\|_F^2 + \frac{1}{2}\|R\|_F^2.$$

These ideas are discussed at length in [7], where numerical tests confirm that driving k to be higher does not result in overfitting the data.

First, we compare the 'low-rank only' formulation (3.4) with the 'smoothing only' formulation

$$\min_W \|\mathcal{L}(W)\|_2^2 \quad \text{s.t.} \quad \|\mathcal{A}(W) - b\|_2 \leq \sigma. \tag{3.14}$$

For this test, we use the inequality constraint $\|\mathcal{A}(W) - b\|_2 \leq \sigma$; the equality constraint ($\mathcal{A}(W) = b$) is infeasible. This requires a root-finding algorithm nearly identical to the W-update of Algorithm 10 in section 3.3. For the low-rank formulation, we use the matricization in Figure 3.2(b). The convergence criteria (l_2-norm of minimized variable(s) iterate difference) for all algorithms is 10^{-10}. All model hyper-parameters are shown in Table 3.2.

Table 3.2: Model hyper-parameters. We set $\gamma = 6.45 \times 10^{-7}$ for all algorithms except low-rank and smoothing only (where it is not used). The smoothing only formulation requires a root-finding problem for the inequality constraint, and has no Max Iteration value corresponding to block-coordinate-descent portion. For FISTA, the step-size $\alpha = \|\mathcal{L}\|_2^{-2}$, the reciprocal of the largest singular value of $\mathcal{L}$. In the joint formulation, we update η every 30 iterations; in the low-rank only, we update every 100 iterations.

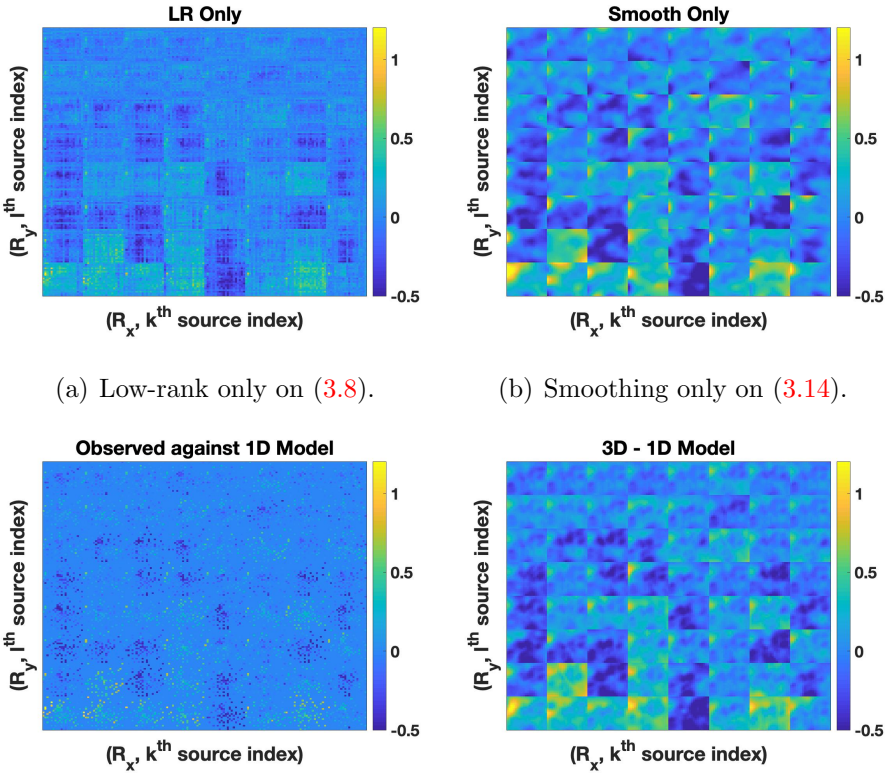
Alg	λ	η	η_f	Max Iterations	σ
Combined - VR Exact	0.0111	0.5	4.17	90	0
Combined - VR Noise	0.0111	0.5	4.17	90	3.719
FISTA	2.2222×10^{-4}	—	—	1500	—
L-BFGS	1.1111×10^{-4}	—	—	1500	—
Smooth only	—	—	—	—	3.719
Low-rank only	—	1.0	4.17	500	3.719

For the proposed relaxation algorithm, η is updated by $\eta_+ = \eta_f \eta$ for $\eta_f = \frac{\sum_j \hat{\sigma}_j}{k}$ at every 30th iteration. Here, $\hat{\sigma}_j$ are the singular values of the pre-interpolated matrix, which is the observed data on the grid and zeros where no data was observed. For low-rank only (with the l_2-norm), we update η ever 100 iterations. Increasing η can accelerate the algorithm [152]. We control η starting with a value that allows a large difference between W and LR^T, and then drive the value down such that $W \approx LR^T$. Recall that W captures smoothness and LR^T the low-rank properties of sources; the scheme drives them closer so that the final solution is both smooth and close to low rank. While Algorithm 10 is guaranteed to converge with any η value [152], the η path affects the behavior of the overall algorithm. In this work, η was updated when the feasibility measure and $\|W - LR^T\|_2$ stopped changing, i.e. when each problem was solved. A systematic approach for continuation is part of ongoing work.

The interpolated results for these two schemes and the *true* data are shown in Figure 3.4 with root-mean-squared (RMS) errors for both observed and interpolated data listed in Table 3.3. The RMS error is calculated with respect to *true* data, and is split into two categories: RMS for *true* data at locations that were observed (designated $\mathcal{A}(X_{true})$ or obs), and RMS for *true* data for locations that were not observed (given by the complement $\mathcal{A}^{\mathsf{c}}(X_{true})$ or int). Recall that in this context, *true* data refers to the synthetic residuals of the 3D model compared to the PNSN S4 1D velocity model uncorrupted by noise. The RMS is not weighted by uncertainties. While the low-rank only interpolation can accurately capture the observed data, it fails to reproduce the missing data.

In Figure 3.4(a), the low-rank interpolation produces mono-color bands across the tensor: when entries for a receiver coordinate are missing in every source, the low-rank mechanism places zeros in the entire row or column. Likewise, if there is a single observed receiver in an entire row or column of the matrix, that value is propagated through every other entry in that row or column. The sparsity of the data impairs low-rank only interpolation, which gives good results at sampling ratio of 70% and above (these results not shown), but is less effective in our context. Smoothing alone, shown in Figure 3.4(b) can capture major model dynamics, yet overestimates observed data worse than other methods and does not overall

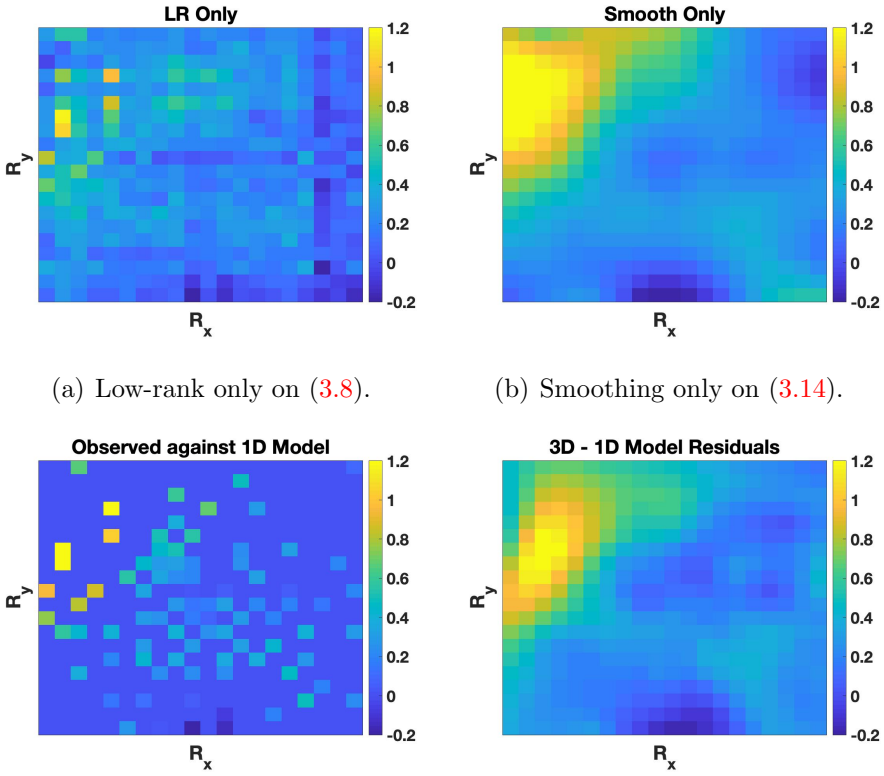
match data as well as other methods (see Table 3.3). While the scale on Figure 3.4(b) breaches is capped at 1.2 (s), smoothing only actually has values larger than 1.5 (s) in the bright yellow section, and the observed differences are also larger (Figure 3.8(f)). It also makes sense that smoothing alone would out-perform low-rank alone, since the underlying model used to generate data is inherently smooth. Some of the overall residual patterns are matched (larger residual energies are correctly placed around larger observed synthetic residuals), but the magnitude of the synthetic residuals is incorrect across all sources. A blown-up version of Figure 3.4 for the source with the highest number of observed data ponts is shown in Figure 3.5. Both low-rank only and smoothing only formulations are inadequate for our application, especially with sparse data.



(a) Low-rank only on (3.8). (b) Smoothing only on (3.14).

(c) Observed synthetic residuals, X_{obs}. (d) *True* synthetic residuals, X_{true}.

Figure 3.4: Full tensor residuals (s) results for Low-rank and smoothing only.



(a) Low-rank only on (3.8).

(b) Smoothing only on (3.14).

(c) Observed synthetic residuals.

(d) *True* synthetic residuals.

Figure 3.5: Travel time residual results (in seconds) for a single source with Low-rank and smoothing only.

In contrast, we show in Figure 3.6 that using both types local and global information can meaningfully interpolate the data. This can be done with the new joint formulation as well as unconstrained formulations that use FISTA and L-BFGS algorithms; so we also test the efficacy of the new approach against these two competitors. We consider two different choices of our variable relaxation algorithm: $\sigma = 0$ and $\sigma > 0$. In the latter case, we assume that we do not know the true data misfit, and use available uncertainties to set $\sigma = \sqrt{\sum_{i=1}^{n} 0.06^2} \approx 3.72(s)$ for n being the size of b. The true data misfit is actually $\|b - \mathcal{A}(X_{true})\|_2 = \sigma_{true} = 5.85(s)$ where $\mathcal{A}(X_{obs}) = b$ is our observed data. The results for tensored and single-source matrix-completion are shown in Figure 3.6 and Figure 3.7, and are

summarized in Table 3.3. The new approach (in bold) achieves better results than competing methods in similar amounts of compute time. Setting $\sigma > 0$ in the variable relaxation scheme produces smaller RMS values, and in particular recovers missing data with higher accuracy.

Table 3.3: Different formulations for synthetic residuals with sampling rate of 15%. Terminal feasibility is $\|\mathcal{A}(X) - b\|_2 - \sigma$ at the algorithm's termination (which would mean l_2-norm data misfit where $\sigma = 0$). Note that the feasibility is calculated against observed data while RMS is calculated against *true* data. Recall that (obs) signifies $\mathcal{A}(X_{true})$, while (int) stands for $\mathcal{A}^c(X_{true})$.

Alg	Terminal Feasibility	Time (s)	RMS (obs)	RMS (int)
Combined - VR Exact	**0.0031**	**10.81**	**0.09**	**0.110**
Combined - VR Noise	**1.18e-08**	**19.84**	**0.06**	**0.100**
FISTA	0.081	12.30	0.09	0.119
L-BFGS	0.074	57.22	0.09	0.128
Smooth only	0.0016	5.42	0.06	0.125
Low-rank only	0.058	1.27	0.08	0.216

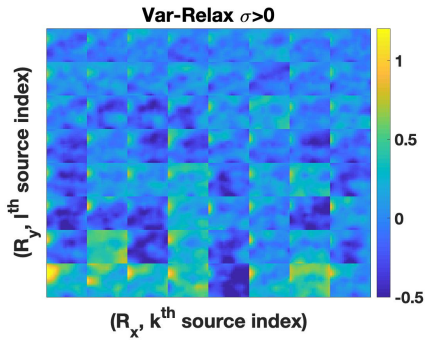
Table 3.3 also shows the degree to which algorithms match the feasibility constraint. FISTA and L-BFGS use penalties rather than constraints, so the precise data-misfit level is hard to control. The terminal feasibility is 0.08(s) for FISTA and 0.075(s) for L-BFGS, which means for each individual point, the values are close to fitting the observed entries. FISTA's feasibility level settles at approximately 0.08(s) and does not change after about 1000 iterations; similar behavior is seen in L-BFGS. Variable relaxation schemes can match the feasibility constraint to a high accuracy, with the explicit inequality constraint matching close to numerical precision.

With both smoothness and low-rank regularization, fitting observed data inexactly yields a better interpolation for missing data. Figure 3.6 shows that variable relaxation inexact

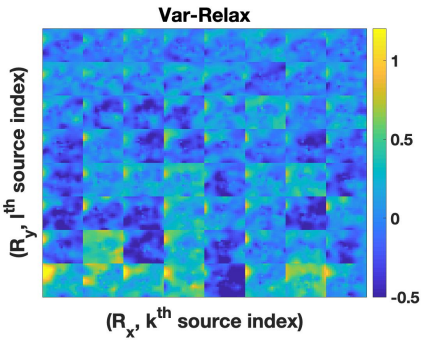
data fit (Figure 3.6(a)) has fewer contrasts overall when compared to the fits obtained using exact fit and competing algorithms. Focusing on a single source in Figure 3.7, we can see that the inequality constraint produces a smoother image for each particular source. While each algorithm effectively captures high energy area in the northwest corner of the plot, most algorithms overestimate the amount of energy that is actually present; variable relaxation with $\sigma > 0$ gets the closest values. All algorithms tend to smooth out the observed entries, and tend to be less accurate the fewer entries there are in a designated space, notably around the corners of the receiver grid. Generally speaking, the combination of smoothness and low-rank works much better for interpolating the interior station grid rather than extrapolation near the edges. Figure 3.8 (full tensor) and Figure 3.9 (single source) depict the absolute values of the results for each algorithm and the true value. Figure 3.8 shows that some sources are estimated very poorly, with the main error contributions coming further away from the mountain.

Zooming in to a particular source, Figure 3.9 shows how much each algorithm overestimates the high-energy in the northwest corner of the map, primarily because the observed data in that corner is also very large. Certain artifacts of the interpolation schemes are seen more vividly in the difference plots. For instance, slightly northeast of the center, there is a higher energy region where very few data points are collected. This is very pronounced in the L-BFGS case (Figure 3.9(d)) but less pronounced in the variable relaxation case (Figure 3.9(a)). Overall, the amount of error present in the $\sigma > 0$ variable relaxation case is lower than all the other schemes. This is particularly significant given the relative magnitude of the problem. We can also solve the problem **exactly** for a variety of general functions; the $\|\mathcal{L}(\cdot)\|_2^2$ smoothing-penalty is pertinent for the seismic iMUSH data, but one can select different functions for different datasets. Competing algorithms, FISTA and L-BFGS, solve approximate variants of the target problem. The simplest Formulation ((3.9)) solved via FISTA ((3.12)) is essentially the Tikhonov Formulation (3.1) with a Laplacian penalty. This solution performs worse overall, especially with respect to interpolating unobserved data. Hence, the approach implemented with Algorithm 10 is preferable, as it solves the tar-

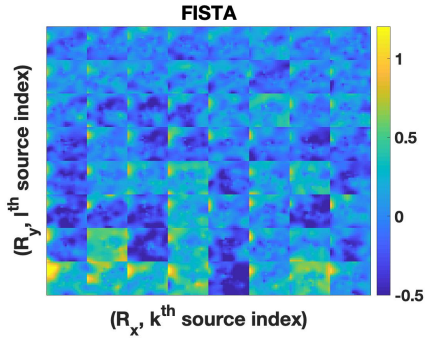
get problem, achieving results with greater accuracy, while requiring similar computational resources as the alternatives.



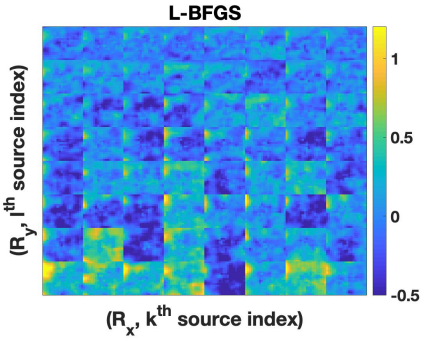
(a) Algorithm 10 ($\sigma > 0$), on (3.9).



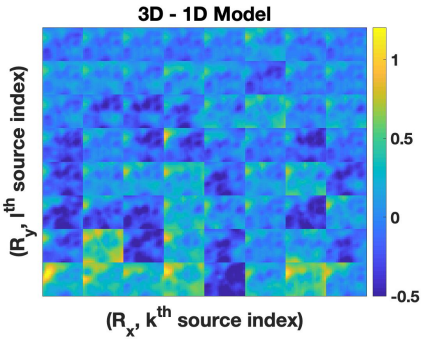
(b) Algorithm 10 ($\sigma = 0$), on (3.9).



(c) Algorithm 11, on (3.12).

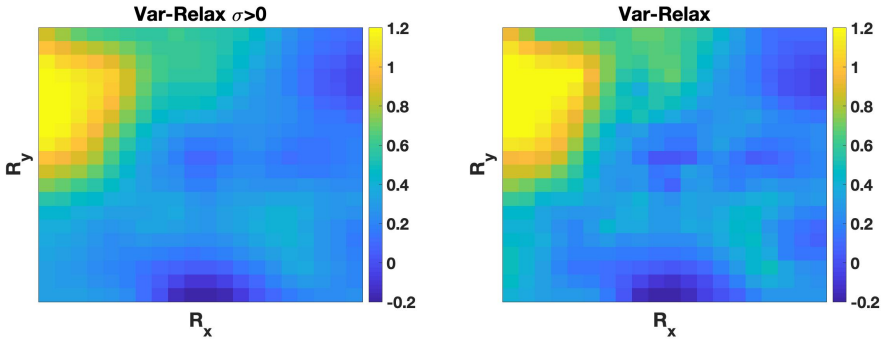


(d) L-BFGS, on (3.13).

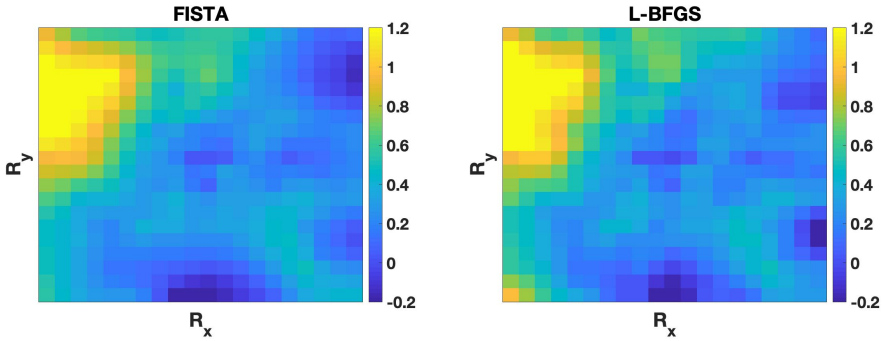


(e) *True* model residuals, X_{true}.

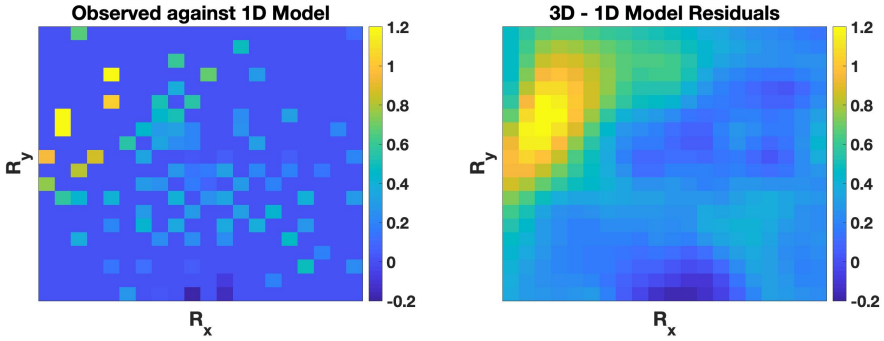
Figure 3.6: Full tensor travel time residual results (s) for different algorithms and formulations.



(a) Algorithm 10 ($\sigma > 0$), on (3.9). (b) Algorithm 10 ($\sigma = 0$), on (3.9).

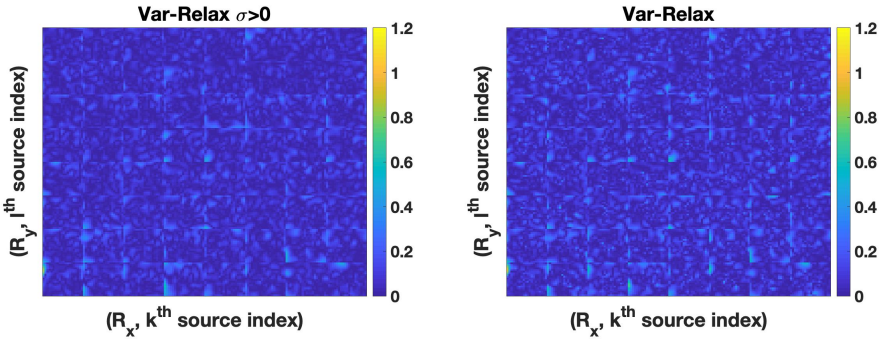


(c) Algorithm 11, on (3.12) (d) L-BFGS, on (3.13).

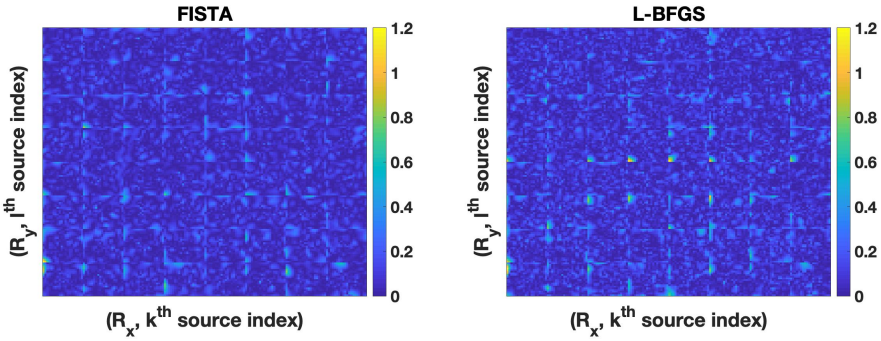


(e) Observed synthetic residuals. (f) *True* synthetic residuals.

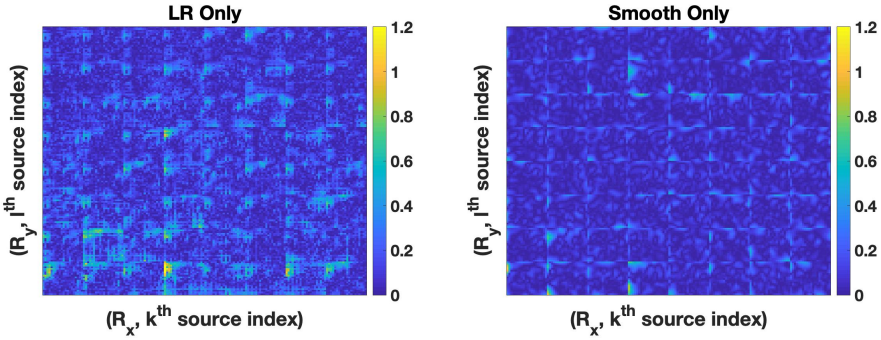
Figure 3.7: Single source synthetic residuals (s) for the different algorithms.



(a) Algorithm 10 ($\sigma > 0$), on (3.9). (b) Algorithm 10 ($\sigma = 0$), on (3.9).

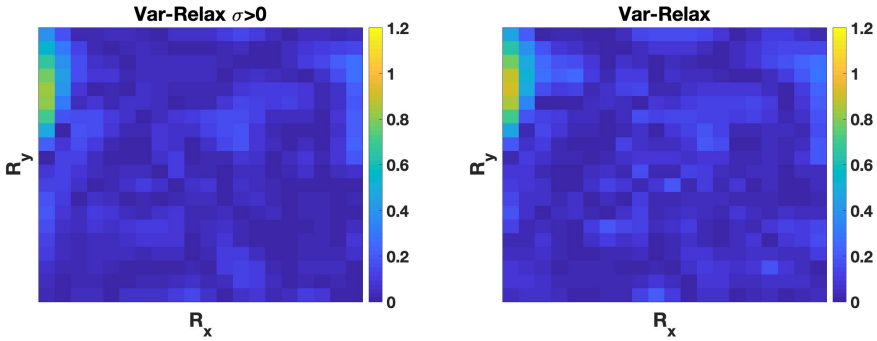


(c) Algorithm 11, on (3.12) (d) L-BFGS, on (3.13).

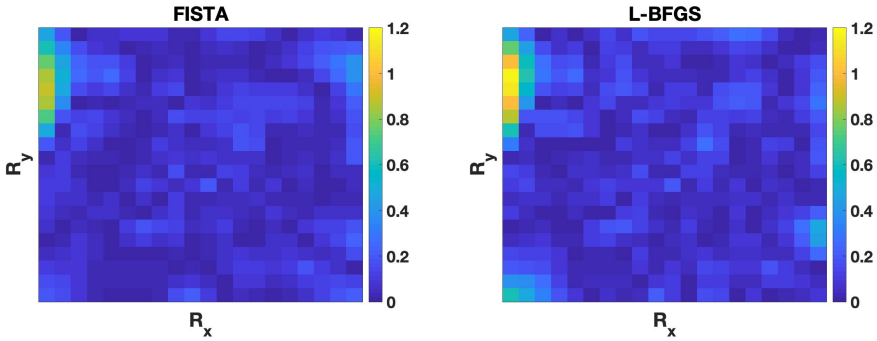


(e) Low-Rank only, on (3.8). (f) Smoothing only, on (3.14).

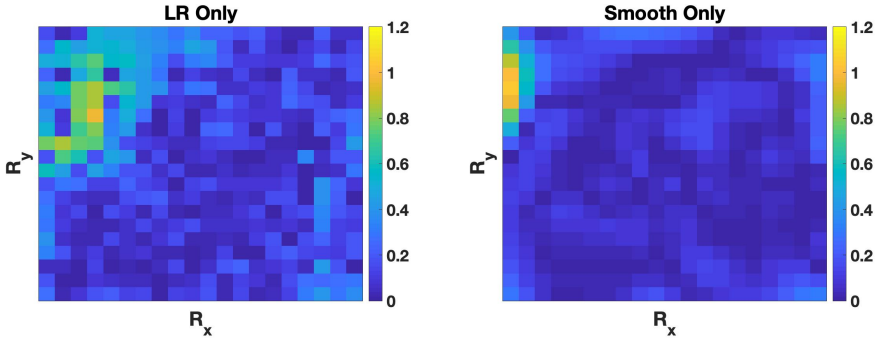
Figure 3.8: Full tensor $|X - X_{true}|$ for all algorithms. Note the significant scaling difference in the FISTA result.



(a) Algorithm 10 ($\sigma > 0$). (b) Algorithm 10 ($\sigma = 0$).



(c) Algorithm 11, on (3.12) (d) L-BFGS, on (3.13).



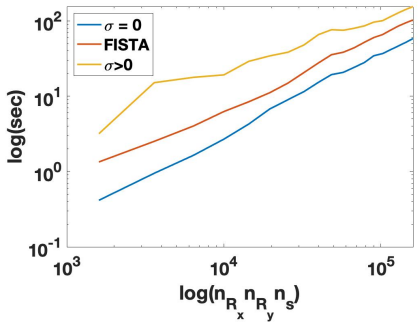
(e) Low-Rank only, on (3.8). (f) Smoothing only, on (3.14).

Figure 3.9: $|X - X_{true}|$ for all algorithms in a single source.

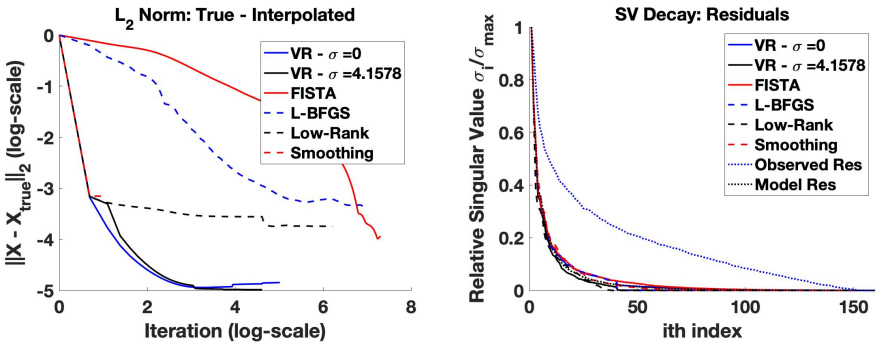
Next, we turn to singular value decay and convergence history. The singular value decay is shown in Figure 3.10(c), and a log-log convergence plot appears in Figure 3.10(b). With the

exception of the smoothing-only implementation, all algorithms match the SVD decay well. In Figure 3.10(b), L-BFGS and FISTA have similar convergence histories but L-BFGS has relatively slow compute time compared to the FISTA (and indeed all other algorithms) when achieving a satisfactory answer. The proposed approach converges faster than competing methods, and also matches the SVD decay of the ground truth datasets.

Finally, we address the scalability of our algorithm and problem. Section 3.5 shows our results in computational time and root mean-squared error as we increase the number of sources from 2^2 to 20^2, the latter of which being the nearest perfect square to the total number of sources observed in the iMUSH experiments. The number of iterations for each algorithm remains the same, as do the hyperparameters for these algorithms. The only change made is that for $\sigma > 0$; here, we make it so the average root mean-squared error is 0.06. Much beyond 20^2, the matrix storage for the smoothing operator becomes infeasible for a laptop computer, and source-separation parallelization would have to occur [82]. From this, we can see that the algorithm, both with and without root-finding, scales in similar computational time to FISTA.



(a) Log-log plot of time (s) as number of sources increases.



(b) Data misfit decay.

(c) SVD decay for interpolated matrices.

Figure 3.10: Convergence information for different algorithms.

3.6 Application to Real Data

Algorithm 10 can be easily modified to fit real data by changing the constraint, $\mathcal{A}(\cdot)$, to another operator that manages the fit between interpolation data and the observed stations. The formulation (3.9) is represented then as

$$\min_{L,R,W} \frac{1}{2}\|L\|_F^2 + \frac{1}{2}\|R\|_F^2 + \frac{1}{2\gamma}\|\mathcal{L}(W)\|_2^2 + \frac{1}{2\eta}\|W - LR^T\|_F^2 \quad \text{s.t.} \quad \|\mathcal{D}(W) - b\|_2 \leq \sigma \quad (3.15)$$

where the only difference between Formulation (3.9) and Formulation (3.15) is the substitution of the restriction operator $\mathcal{A}(\cdot)$ with an interpolation operator $\mathcal{D}(\cdot) : \mathbb{R}^{n_{R_x}\sqrt{n_s}\times n_{R_y}\sqrt{n_s}} \to \mathbb{R}^{n_d}$. The only requirement on the interpolation operator is that it is linear; i.e. can be represented as a matrix. In this problem, the interpolation operator takes the tensor W, permutes it and then uses a 2D linear Lagrange interpolation function to map it to the observed field data for all sources, which is in $b \in \mathbb{R}^{n_d}$. Hence, step 6 of Algorithm 10 becomes

$$W_{k+1} = \arg\min_W \|\mathcal{L}(W)\|_2^2 + \frac{1}{2\eta}\|W - L_{k+1}R_{k+1}^T\|_F^2 \quad \text{s.t. } \|\mathcal{D}(W) - b\|_2 \leq \sigma \quad (3.16)$$

which collapses to

$$w(\lambda) := \arg\min_w \frac{1}{2\gamma}\|L_\nabla w\|_2^2 + \frac{1}{2\eta}\|w - d_{k+1}\|_F^2 + \frac{\lambda}{2}\|Dw - b\|^2, \quad (3.17)$$

where $D \in \mathbb{R}^{n_d \times n_{R_x} n_{R_y} n_s}$ is the matrix representing our interpolation operator. Hence our solution for w is

$$w(\lambda) = \left(\frac{1}{\gamma}(L_\nabla^T L_\nabla) + \frac{1}{\eta} I + \lambda D^T D\right)^{-1} \left(\lambda D^T b + \frac{1}{\eta} d_{k+1}\right).$$

Another way of formulating this problem is to take the analytic perspective; that is, we let $w \in \mathbb{R}^{n_d}$ and modify the problem as such

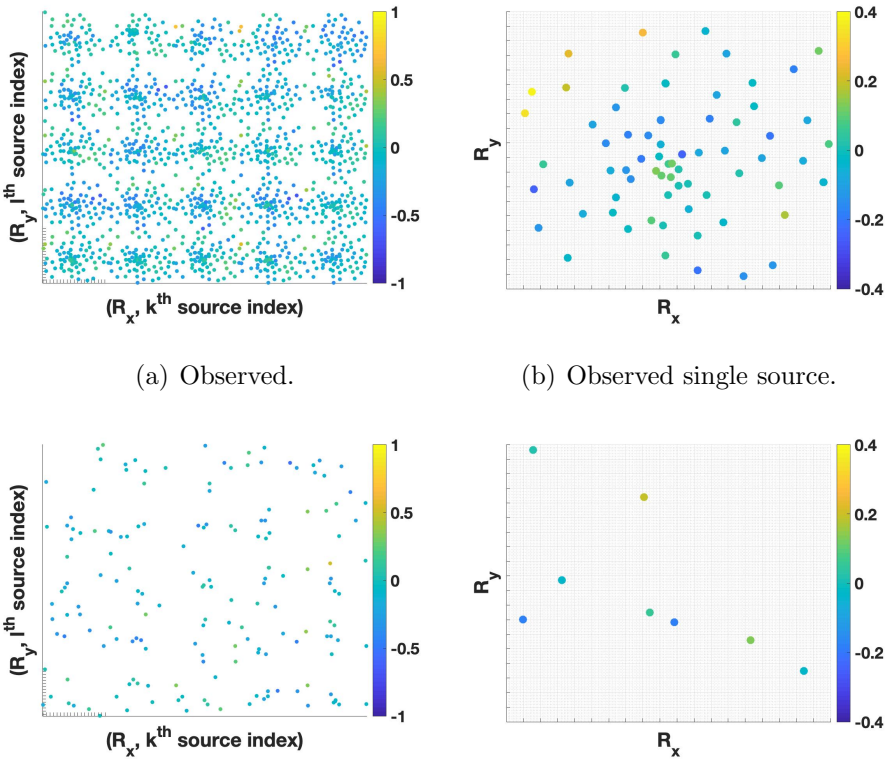
$$\min_{L,R,w} \frac{1}{2}\|L\|_F^2 + \frac{1}{2}\|R\|_F^2 + \frac{1}{2\gamma}\|\mathcal{L}\left(\mathcal{D}(w)\right)\|_2^2 + \frac{1}{2\eta}\|\mathcal{D}(w) - LR^T\|_F^2 \quad \text{s.t.} \quad \|w - b\|_2 \leq \sigma \tag{3.18}$$

where $\mathcal{D}(\cdot) : \mathbb{R}^{n_d} \to \mathbb{R}^{n_{R_x}\sqrt{n_s} \times n_{R_y}\sqrt{n_s}}$; this makes our root-finding process a bit easier to implement. However, both approaches yield the same results.

For real data, the number of sources is reduced to $n_s = 25$, while the grid remains the same. We withhold 10% of the observed data for cross-validation purposes, and perform Algorithm 10 with (3.15) for $\sigma = 0$. The results for this are shown in Figure 3.12 and Figure 3.13 and are summarized in Table 3.4. These results show that our accuracy is greater the closer the points are to the center of the mountain. This makes sense, as there are more sensors closer to the mountain than in the surrounding area. However, we see an overall smoothness in the interpolated solution. This is a route for further exploration, and possibly the makings of another paper.

Table 3.4: Results for real data with a sub-sampling rate of 10%. We ran the modification of Algorithm 10 with (3.15) with $\sigma = 0$. Since we don't have *true* data for comparison, this is simply compared to withheld data to measure accuracy, despite uncertainty in observed data.

Data	Mean $\lvert X - X_{obs}\rvert$	RMS	Median $\lvert X - X_{obs}\rvert$
Observed	0.14	1.3e-3	0.115
Withheld	0.15	1.4e-3	0.133



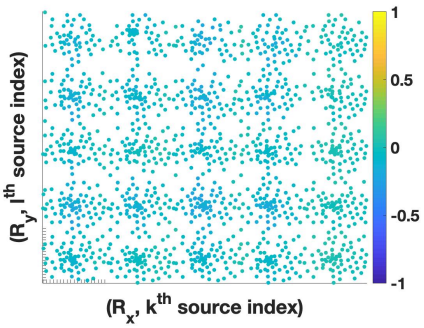
(a) Observed.

(b) Observed single source.

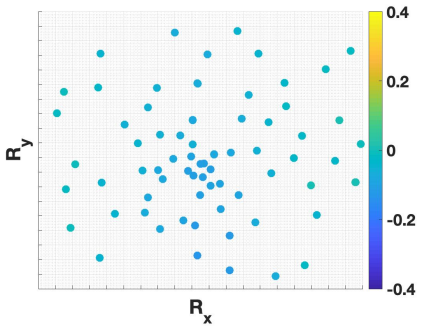
(c) Omitted.

(d) Omitted single source.

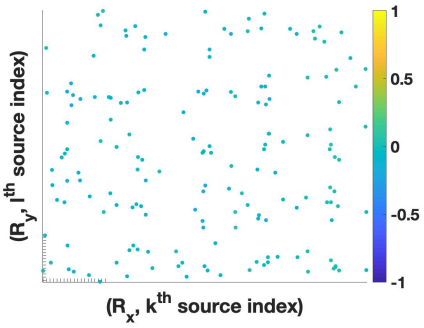
Figure 3.11: Observed and Omitted data



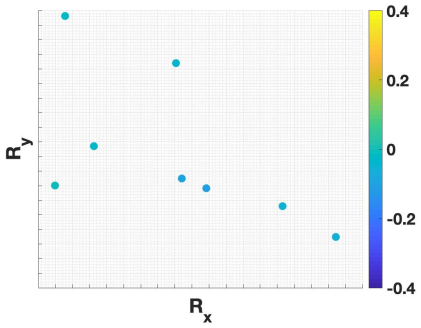
(a) Interpolated at observed receivers.



(b) Interpolated at observed receivers for a single source.

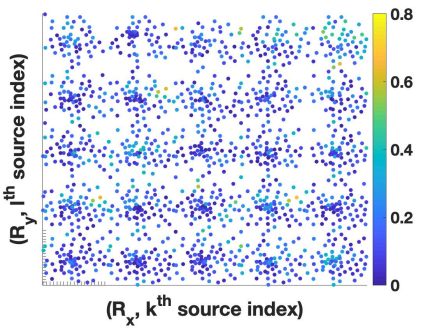


(c) Interpolated at hidden receivers.

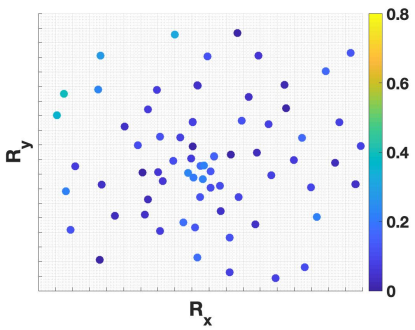


(d) Interpolated at hidden receivers for a single source.

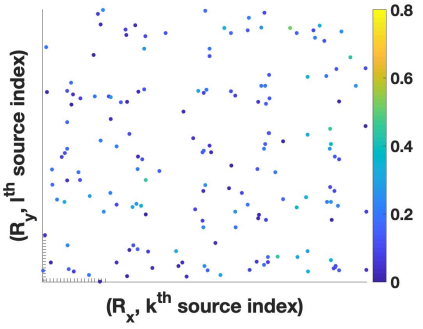
Figure 3.12: Interpolation results for travel time residuals (s) for Algorithm 10 with real IMUSH data.



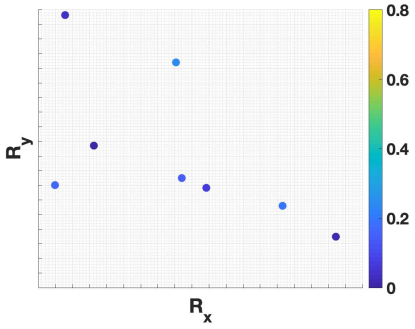
(a) Difference in observed



(b) Difference in observed for a single source.



(c) Difference at hidden receivers.



(d) Difference at hidden receivers for a single source.

Figure 3.13: Receiver Differences for real IMUSH data.

3.7 Conclusions and Future Directions

Travel time tomography suffers from data collection constraints, reducing model resolution. We proposed an interpolation scheme that combines both local smoothness and low-rank information from a carefully chosen tessellation of the data to estimate residual values at prospective stations. To implement the scheme, we developed a new relaxation approach that is flexible enough to allow multiple regularizers, and efficient in practice. We used it to estimate data from missing stations with a relatively high degree of accuracy (measured against a synthetic ground-truth dataset), in the presence of observation noise in available

data. The algorithm is more flexible than available alternatives, and in particular can fit available data to a prescribed error level. The new approach is competitive with standard alternatives, and offers new functionality to interpolate with both local and global structure over data fitting constraints. It is important to note that none of the methods do well estimating where there are very few stations. This can be observed in Figure 3.9, where the top left corner of each graph has a high residual and no stations present towards border of the receiver space. The proximity of at least one station increases the accuracy of our scheme.

It is also important to address the proposed future work of this problem, in which we would use these interpolated/denoised stations in the inversion. The goal for future work would to analyze the impact of these new stations on the direct 3D structure approximation and compare it to the same inversion with the available observations. It is certainly possible that the interpolated observations would bias the inversion; given that the procedure described in this section does poorly in spaces with little coverage, it is likely that biases would show up in the overall inversion. There may be something to say about increasing the quality of the station data by denoising with our procedure; this would then have to be compared against other regularization techniques. One can also use our procedure, which is fairly cheap, as a method of determining station reliability, station redundance, or where to put new stations. For instance, if one were to omit certain stations that were close to each other and the interpolation/denoising routine was able to accurately recapture this value, it may be likely that this station produces information already captured by other stations. In that case, one could move this redundant station to a new location to capture more information. However, IMUSH is already completed, and this could be used for future work in the region around Mount St. Helens. We leave the question of biasing the inversion with a preliminary routine to future work.

In the next chapter, we expand on the themes developed in this chapter to accomodate general ℓ_p norm constraints. To do this, we actually relax and split twice, which offers and algorithm similar to Algorithm 10 and Algorithm 3 but simpler in practice and with better

convergence guarantees.

Chapter 4

BASIS PURSUIT DENOISE WITH NONSMOOTH CONSTRAINTS

4.1 Introduction & Motivation

Chapter 3 focused on a particular seismic application which featured a regularized, nonsmooth cost function and level-set constraint (3.3). In this chapter, we generalize this to general nonsmooth level-set constrained problems, but disregard the regularizer for the time being. The main contribution of this chapter is to provide a fast, easily adaptable algorithm to solve non-smooth and nonconvex data constraints in general level-set formulations including BPDN, and illustrate the efficacy of the approach using large-scale interpolation and denoising problems. To do this, we extend the universal regularization framework of [153] to level-set formulations with nonsmooth/nonconvex constraints. We develop a convergence theory for the optimization approach, and illustrate the practical performance of the new formulations for data interpolation and denoising in both sparse recovery and low-rank matrix factorization.

Basis Pursuit Denoise (BPDN) seeks a sparse solution to an under-deterimined system of equations that have been corrupted by noise. The classic level-set formulation [8, 142] is given by

$$\min_x \|x\|_1 \quad \text{s.t.} \quad \|\mathcal{A}(x) - b\|_2 \leq \Delta \tag{4.1}$$

where $\mathcal{A} : \mathbb{R}^{m\times n} \to \mathbb{R}^d$ is a linear functional taking unknown parameters $x \in \mathbb{R}^{m\times n}$ to observations $b \in \mathbb{R}^d$. Problem (4.1) is also known as a Morozov formulation (in contrast to Ivanov or Tikhonov [108]). The functional $\mathcal{A}$ can include a transformation to another domain, including Wavelets, Fourier, or Curvelet coefficients [37], as well as compositions of these transforms with other linear operators such as restriction in interpolation problems. Typically, the system is underdetermined with $d \ll nm$. The inequality constraint (as opposed to the equality constraint) is due to the fact that with noisy data, fitting the data exactly may be less accurate overall. The parameter Δ controls the error budget, and is based on an estimate of noise level in the data.

Problems with cardinality or sparsity constraints, such as BPDN and the closely related LASSO formulation, have applications to compressed sensing [33, 50, 117], image process-

ing [95], sparse controller design [90], and machine learning [52, 58], as well as to more applied domains including MRI [94] and seismic inversion [131]. Indeed, the total variation norm, as well as other sparsity-inducing functions, has been used to denoise images since the 1980s [46, 121]. Seismic data is a key use case [7, 44, 80], and the primary application for the techniques developed in this paper. Data acquisition is prohibitively expensive in seismology and interpolation techniques are used to fill in data volumes by promoting parsimonious representations in the Fourier [122] or Curvelet [69] domains. Matricization of the data leads to low-rank interpolation schemes [7, 44, 80, 150]. The types of errors encountered in seismic inversion motivate the technology developed here. In our numerics section, we show how using the ℓ_0 norm for the data misfit constraints can help deal with the situation where uniformly large errors applied across a range of scales creates a variable SNR that stops more common losses from effectively recovering the signal.

Related Work. Theoretical recovery guarantees for classes of operators $\mathcal{A}$ are developed in [33] and [138]. While BPDN uses nonsmooth regularizers (including the ℓ_1 norm, nuclear norm, and elastic net), the inequality constraint is ubiquitously smooth, and often taken to be the ℓ_2 norm as in (4.1). Indeed, the ℓ_1 norm is particularly useful in that it is a convex proxy for sparsity [127] and much work has been done to remain in this convex regime, despite some disadvantages of the ℓ_1 norm as a sparsity-inducing metric [126]. In most applications of BPDN, the ℓ_2 norm in the context of Gaussian noise takes on the interpretation that Δ in Problem (4.1) is the variance of the noise [49]. In almost all contexts, this noise is assumed to be Gaussian [55, 126, 127]. This stems from the very nature of denoising/image reconstruction being a very ill-posed problem predicated on *a-priori* information about the data itself. Prior work, including [7, 8, 44, 142], exploits the smoothness of the inequality constraint in developing algorithms for the problem class. These smooth constraints work well when errors are Gaussian, but this assumption fails for seismic data (explored in [131]) and is often violated in many applications, from sparse controller design [90] to compressed sensing [50].

The use of cardinality constraints (ie the ℓ_0 norm) to enforce sparsity has been studied

in depth in the case of convex cost functionals [11] or continuously differentiable cost functionals [16]. Nonsmooth data fidelity terms have been explored by Nikolova in [105], whose work underscores the utility of using these terms in modeling. We develop a broad class that captures any nonsmooth nonconvex regularizer and/or misfit, and gives a way to set an error-budget (Δ in (4.1)). We also explore the matrix variant of the problem, which can be viewed as an extension of robust PCA [6, 34, 62]. Robust PCA is equivalent to minimizing a rank functional subject to a Huber data misfit [51], and hence misfit constrained versions of the problem use smooth misfit constraints, see e.g. [7]. Our formulation extends all these approaches to nonsmooth data misfit constraints, including cardinality constraints, just as in the vector case.

Roadmap. Section 4.2 develops the general relaxation framework and approach. Section 4.3 specifies this framework to the BPDN setting with nonsmooth, nonconvex constraints. In section 4.4 we apply the approach to sparse signal recovery problem and sparse Curvelet reconstruction. In section 4.5, we extend the approach to a low-rank interpolation framework, which embeds matrix factorization within the BPDN constraint. In section 4.6 we test the low-rank extension using synthetic examples and data extracted from a full 5D dataset simulated on complex SEG/EAGE overthrust model.

4.2 Nonsmooth, nonconvex level-set formulations.

We consider the following problem class:

$$\min_x \varphi(\mathcal{C}(x)) \quad \text{s.t.} \quad \psi(\mathcal{A}(x) - b) \leq \Delta, \tag{4.2}$$

where φ and ψ may be nonsmooth, nonconvex, but have well-defined proximity (2.1b) and projection Definition 2 operators, with the latter taking the form

$$\underset{\psi(\cdot)\leq\Delta}{\text{proj}} = \arg\min_{\psi(x)\leq\Delta} \frac{1}{2\eta}\|x - y\|^2. \tag{4.3}$$

Here, $\mathcal{C} : \mathbb{C}^{m\times n} \to \mathbb{R}^c$ is typically a linear operator that converts x to some transform domain, while $\mathcal{A} : \mathbb{C}^{m\times n} \to \mathbb{R}^d$ is a linear observation operator also acting on x. In the context of interpolation, $\mathcal{A}$ is often a restriction operator.

Algorithm 12 Prox-gradient for (4.4).

Input: $x_0, w_{0,1}, w_{0,2}$

Initialize: $k = 0$

while not converged **do**

$$x_{k+1} \leftarrow x^k - \beta\left(\frac{1}{\nu_1}\mathcal{C}^T(\mathcal{C}(x) - w_{k,1}) + \frac{1}{\nu_2}\mathcal{A}^T(\mathcal{A}(x) - w_{k,2} - b)\right)$$

$$w_{k+1,1} \leftarrow \text{prox}_{\beta\varphi}\left(w_{k,1} - \tfrac{\beta}{\nu_1}(w_{k,1} - \mathcal{C}(x_{k+1}))\right)$$

$$w_{k+1,2} \leftarrow \text{proj}_{\Delta\mathbb{B}_\psi}\left(w_{k,2} - \tfrac{\beta}{\nu_2}(w_{k,2} - (\mathcal{A}(x_{k+1}) - b))\right)$$

$k \leftarrow k + 1$

Output: $w_{k,1}, w_{k,2}, x_k$

This setting significantly extends that of [8], who assume ψ and φ are convex, $\mathcal{C} = I$, and use the *value function*

$$v(\tau) = \min_x \psi(\mathcal{A}(x) - b) \quad \text{s.t.} \quad \varphi(x) \le \tau$$

to solve (4.2) using root-finding to solve $v(\tau) = \Delta$. Variational properties of v are fully only understood in the convex setting, and efficient evaluation of $v(\tau)$ requires ψ to be smooth, so that efficient first-order methods are applicable.

Here, we develop an approach to solve any problem of type (4.2), including problems with nonsmooth and nonconvex ψ, φ, using only matrix vector products with $\mathcal{A}, \mathcal{A}^T, \mathcal{C}, \mathcal{C}^T$ and simple nonlinear operators. In special cases, the approach can also use equation solves to gain significant speedup.

The general approach uses the relaxation formulation proposed in [152, 153]. We use relaxation to split φ, ψ from the linear map $\mathcal{A}$ and transformation map $\mathcal{C}$, extending (4.2) to

$$\begin{aligned} \min_{x,w_1,w_2} \ & \varphi(w_1) + \frac{1}{2\nu_1}\|\mathcal{C}(x) - w_1\|^2 + \frac{1}{2\nu_2}\|w_2 - \mathcal{A}(x) + b\|_2^2 \\ \text{s.t.} \ & \psi(w_2) \le \Delta. \end{aligned} \tag{4.4}$$

with $w_1 \in \mathbb{R}^c$ and $w_2 \in \mathbb{R}^d$. In contrast to [153], we use a continuation scheme to force $\nu_i \to 0$, in order to solve the original formulation (4.2). Thus the only external algorithmic parameter the scheme requires is Δ, which controls the error budget for ψ.

There are two algorithms readily available to solve (4.4). The first is prox-gradient descent, detailed in Algorithm 12. We let $\mathbf{s} = [x, w_1, w_2]^T$, and define

$$\Phi(\mathbf{s}) = \varphi(w_1) + \delta_{\psi(\cdot)\leq\Delta}(w_2),$$

where the *indicator function* $\delta_{\psi(\cdot)\leq\Delta}$ takes the value 0 if $\psi(w_2) \leq \Delta$, and infinity otherwise. Problem (4.4) can now be written as

$$\underset{\mathbf{s}}{\text{minimize}} \underbrace{\frac{1}{2}\left\| \begin{bmatrix} \frac{1}{\sqrt{\nu_1}}\mathcal{C} & -\frac{1}{\sqrt{\nu_1}}I & 0 \\ \frac{1}{\sqrt{\nu_2}}\mathcal{A} & 0 & -\frac{1}{\sqrt{\nu_2}}I \end{bmatrix} \mathbf{s} - \begin{bmatrix} 0 \\ b \end{bmatrix} \right\|^2}_{f(\mathbf{s})} + \Phi(\mathbf{s}). \tag{4.5}$$

Applying the prox-gradient descent iteration (2.5) with step-size β

$$\mathbf{s}_{k+1} = \underset{\beta\Phi}{\text{prox}}(\mathbf{s}_k - \beta\nabla f(\mathbf{s}_k)) \tag{4.6}$$

gives the coordinate updates in Algorithm 12.

Prox-gradient has been analyzed in the general nonconvex setting by [9, 10]. Since Problem (4.4) is semi-algebraic, we have from [10] that Algorithm 12 (and its subsequent reductions, Algorithm 13-Algorithm 15) converge to a critical point. However, we can exploit the nature of our relaxation to make simple improvements on the convergence rates for our particular problem via selection of proximal-gradient step β, detailed in upcoming sections.

Problem (4.5) is the sum of a convex quadratic and a nonconvex regularizer, and the rate of convergence for this problem class can be quantified using [153, Theorem 2], reproduced below.

Theorem 4.2.1 (Prox-gradient for Regularized Least Squares)**.** *Consider the least squares objective*

$$\underset{\mathbf{s}}{\text{minimize}}\, p(\mathbf{s}) := \frac{1}{2}\|G\mathbf{s} - g\|^2 + \Phi(\mathbf{s}).$$

Algorithm 13 Value-function optimization for (4.4).

Input: $x_0, w_{1,0}, w_{2,0}$

Initialize: $k = 0$

Define: $\mathcal{H} = \frac{1}{\nu_1}\mathcal{C}^T\mathcal{C} + \frac{1}{\nu_2}\mathcal{A}^T\mathcal{A}$

while not converged **do**

$x_{k+1} \leftarrow \mathcal{H}^{-1}\left(\frac{1}{\nu_1}\mathcal{C}^T w_{k,1} + \frac{1}{\nu_2}\mathcal{A}^T(b + w_{k,2})\right)$

$w_{k+1,1} \leftarrow \mathrm{prox}_{\beta\varphi}\left(w_{k,1} - \frac{\beta}{\nu_1}(w_{k,1} - \mathcal{C}(x_{k+1}))\right)$

$w_{k+1,2} \leftarrow \mathrm{proj}_{\Delta\mathbb{B}_\psi}\left(w_{k,2} - \frac{\beta}{\nu_2}(w_{k,2} - (\mathcal{A}(x_{k+1}) - b)\right)$

$k \leftarrow k + 1$

Output: $w_{k,1}, w_{k,2}, x_k$

with p bounded below, and Φ potentially nonsmooth, nonconvex, and non-finite valued. With step $\beta = \|G\|_2^{-2} = \Delta_{\max}(G)^{-2}$, the iterates (4.6) satisfy

$$\min_{k=0,\ldots,N} \|v_{k+1}\|^2 \leq \frac{\|G\|^2}{N}(p(\mathbf{s}_0) - \inf p)$$

where

$$v_k = (\|G\|_2^2 I - G^T G)(\mathbf{s}_k - \mathbf{s}_{k+1}) \in \partial p(\mathbf{s}_{k+1})$$

is a subgradient (generalized gradient) of p at $\mathbf{s}_{k+1}$.

We can specialize Theorem 4.2.1 to our case by computing the norm of the least squares system in (4.5).

Corollary 4.2.2 (Rate for Algorithm 12)**.** *Theorem 4.2.1 applied to Problem (4.4) gives*

$$\min_{k=0,\ldots,N} \|v_{k+1}\|^2 \leq C(\nu_1, \nu_2, \mathcal{C}, \mathcal{A})\frac{1}{N}(p(\mathbf{s}_0) - \inf p)$$

with

$$C(\nu_1, \nu_2, \mathcal{C}, \mathcal{A}) = \frac{1}{\nu_1}(c + \|\mathcal{C}\|_F^2) + \frac{1}{\nu_2}(d + \|\mathcal{A}\|_F^2).$$

Problem (4.4) also admits a different optimization strategy, summarized in Algorithm 13. We can formally minimize the objective in x directly via the gradient, with the minimizer given by

$$x(\mathbf{w}) = \mathcal{H}^{-1}\left(\begin{bmatrix}\nu_1^{-1}\mathcal{C}^T & \nu_2^{-1}\mathcal{A}^T\end{bmatrix}\mathbf{w} + \nu_2^{-1}\mathcal{A}^T b\right)$$
$$\mathcal{H} = \frac{1}{\nu_1}\mathcal{C}^T\mathcal{C} + \frac{1}{\nu_2}\mathcal{A}^T\mathcal{A}$$

with $\mathbf{w} = [w_1, w_2]^T$. From $\mathcal{A}$ and $\mathcal{C}$, we have that $\mathcal{H} \in \mathbb{C}^{mn\times mn}$ (for vectorized x). A direct solution is obtained by taking a Cholesky decomposition $\mathcal{H}$ (as it is SPD) and using back-substitution on the result for $\mathcal{O}\left((mn)^3/3\right)$ FLOPs. However, this requires a Cholesky decomposition every time ν_1 and ν_2 are updated. This cost and potentially huge nature of $\mathcal{H}$ means that conjugate gradient descent can also be used to solve the least squares problem for $x(\mathbf{w})$. In section 4.2.1, we explore *inexact* least square solves, and show that convergence is possible even for minimal CG iterations. Once $x(\mathbf{w})$ is solved for directly, this expression is plugged back in to give a regularized least squares problem in $\mathbf{w}$ alone:

$$\begin{aligned}\min_{w_1,w_2} p(\mathbf{w}) &:= \varphi(w_1) + \left\|\mathcal{F}\mathbf{w} - \tilde{b}\right\|^2 \quad \text{s.t.} \quad \psi(w_2) \le \Delta \\ \mathcal{F} &= \begin{bmatrix}\frac{1}{\sqrt{\nu_1}}\left(\frac{1}{\nu_1}\mathcal{C}\mathcal{H}^{-1}\mathcal{C}^T - I\right) & \frac{1}{\sqrt{\nu_1}\nu_2}\mathcal{C}\mathcal{H}^{-1}\mathcal{A}^T \\ \frac{-1}{\sqrt{\nu_2}\nu_1}\mathcal{A}\mathcal{H}^{-1}\mathcal{C}^T & \frac{1}{\sqrt{\nu_2}}\left(I - \frac{1}{\nu_1}\mathcal{A}\mathcal{H}^{-1}\mathcal{A}^T\right)\end{bmatrix} \\ \widetilde{b} &= \begin{bmatrix}\frac{-1}{\sqrt{\nu_1}\nu_2}\mathcal{C}\mathcal{H}^{-1}\mathcal{A}^T b \\ \frac{1}{\sqrt{\nu_2}}\left(\frac{1}{\nu_1}\mathcal{A}\mathcal{H}^{-1}\mathcal{A}^T - I\right)b\end{bmatrix}.\end{aligned} \tag{4.7}$$

Prox-gradient applied to the *value function* $p(\mathbf{w})$ in (4.7) with step β gives the iteration

$$\mathbf{w}_{k+1} = \operatorname*{prox}_{\beta\Phi}(\mathbf{w}_k - \beta\mathcal{F}^T(\mathcal{F}\mathbf{w}_k - \widetilde{b})) \tag{4.8}$$

This iteration, as formally written, requires forming and applying the system $\mathcal{F}$ in (4.7) at each iteration. In practice we compute the $x(\mathbf{w})$ update on the fly, as detailed in Algorithm 13. The equivalence of Algorithm 13 to iteration (4.8) comes from the following

derivative formula for value functions [19]:

$$\mathcal{F}^T(\mathcal{F}\mathbf{w} - \widetilde{b})) = \frac{1}{\nu_1}\mathcal{C}^T(\mathcal{C}(x(\mathbf{w})) - w_1) + \frac{1}{\nu_2}\mathcal{A}^T(\mathcal{A}(x(\mathbf{w})) - (w_2 + b)).$$

In order to compute β, and apply Theorem 4.2.1, we first prove the following lemma:

Lemma 4.2.3 (Bound on $\|\mathcal{F}^T\mathcal{F}\|_2$)**.** *The operator norm $\|\mathcal{F}^T\mathcal{F}\|_2$ is bounded above by* $\max\left(\frac{1}{\nu_1}, \frac{1}{\nu_2}\right)$.

Proof. Considering the function

$$\|\mathcal{F}\mathbf{w} - \widetilde{b}\|^2 = \min_x \underbrace{\frac{1}{2\nu_1}\|\mathcal{C}(x) - w_1\|^2 + \frac{1}{2\nu_2}\|w_2 - \mathcal{A}(x) + b\|_2^2}_{Q(x,w)},$$

we know that the gradient is given by $\mathcal{F}^T(\mathcal{F}\mathbf{w} - \widetilde{b})$, and any Lipschitz bound L gives

$$\|\mathcal{F}^T\mathcal{F}w_1 - \mathcal{F}^T\mathcal{F}w_2\| \leq L\|w_1 - w_2\|,$$

which means $\|\mathcal{F}^T\mathcal{F}\|_2 \leq L$. On the other hand, we can write the right hand side as

$$Q(\mathbf{w}, x) = q(D\mathbf{w}, x)$$

where

$$q(\mathbf{s}, x) = \frac{1}{2}\left\|\mathbf{s} - \begin{bmatrix} \frac{1}{\sqrt{\nu_1}}\mathcal{C}(x) \\ \frac{1}{\sqrt{\nu_2}}\mathcal{A}(x) \end{bmatrix} - \begin{bmatrix} 0 \\ b \end{bmatrix}\right\|^2$$

and

$$D = \begin{bmatrix} \frac{1}{\sqrt{\nu_1}} & 0 \\ 0 & \frac{1}{\sqrt{\nu_2}} \end{bmatrix}.$$

Using [152, Theorem 1] with $g(\mathbf{z}) = 0$, we have that the value function

$$\widetilde{q}(\mathbf{s}) = \min_x q(\mathbf{s}, x)$$

is differentiable, with $\text{lip}(\nabla\widetilde{q}) \leq 1$. Therefore

$$\widetilde{Q}(\mathbf{w}) = \min_x Q(\mathbf{w}, x)$$

is also differentiable, with

$$\nabla \widetilde{Q}(\mathbf{w}) = D^T \nabla \widetilde{q}(D\mathbf{w}),$$

and hence

$$\operatorname{lip}(\nabla \widetilde{Q}) \leq \|D^T D\|_2 = \max\left(\frac{1}{\nu_1}, \frac{1}{\nu_2}\right).$$

This immediately gives the result. □

Now we can combine iteration (4.8) with Theorem 4.2.1 to get a rate of convergence for Algorithm 13.

Corollary 4.2.4 (Convergence of Algorithm 13)**.** *When β satisfies*

$$\beta \leq \min(\nu_1, \nu_2),$$

the iterates of Algorithm 13 satisfy

$$\min_{k=0,\ldots,N} \|v_{k+1}\|^2 \leq \frac{1}{N} \max\left(\frac{1}{\nu_1}, \frac{1}{\nu_2}\right) (p(\mathbf{w}_0) - \inf p))$$

where v_k is in the subdifferential (generalized gradient) of objective (4.7) at $\mathbf{w}_k$. Moreover, if $\nu_1 = \nu_2$, then Algorithm (13) is equivalent to block-coordinate descent, as detailed in Algorithm 14.

Proof. The convergence statement comes directly from plugging the estimate of iteration (4.8) into Theorem 4.2.1. The equivalence of Algorithm 14 with Algorithm 13 is obtained by plugging in step size $\beta = \nu_1 = \nu_2$ into each line of Algorithm 13. □

An important consequence of Corollary 4.2.4 is that the convergence rate of Algorithm 13 does not depend on $\mathcal{C}$ or $\mathcal{A}$, in contrast to Algorithm 12, whose rate depends on both matrices (Corollary 4.2.2). The rates of both algorithms are affected by (ν_1, ν_2). We use continuation in ν_i, driving (ν_1, ν_2) to $(0, 0)$ at the same rate, and warm-starting each problem at the previous solution. A convergence theory that takes this continuation into account is left to future work.

Algorithm 14 Block-coordinate descent for (4.4).

Input: $x_0, w_{1,0}, w_{2,0}$

Initialize: $k = 0$

Define: $\mathcal{H} = \frac{1}{\nu_1}\mathcal{C}^T\mathcal{C} + \frac{1}{\nu_2}\mathcal{A}^T\mathcal{A}$

while not converged **do**

$x_{k+1} \leftarrow \mathcal{H}^{-1}\left(\frac{1}{\nu_1}\mathcal{C}^T w_{k,1} + \frac{1}{\nu_2}\mathcal{A}^T(b + w_{k,2})\right)$

$w_{k+1,1} \leftarrow \text{prox}_{\nu_1\varphi}\left(\mathcal{C}(x_{k+1})\right)$

$w_{k+1,2} \leftarrow \text{proj}_{\Delta\mathbb{B}_\psi}\left(\mathcal{A}(x_{k+1}) - b\right)$

$k \leftarrow k + 1$

Output: $w_{k,1}, w_{k,2}, x_k$

Algorithm 14 is similar to the Proximal Alternating Minimization (PAM) algorithm [9]. Indeed PAM and other algorithms, such as the linearized version of PAM called PALM [27] can be used to solve the relaxed problem (4.2). However the PAM algorithm is different from Algorithm 14, since it requires additional proximal terms. The analysis using the value function reduces problem (4.2) to a simpler problem, the sum of a quadratic in $\begin{bmatrix} w_1 & w_2 \end{bmatrix}$ and a nonconvex regularizer in $\begin{bmatrix} w_1 & w_2 \end{bmatrix}$, and allows the simple proximal gradient method. The detailed implementation of this approach, with explicit x updates, gives Algorithm 14. Moreover we get a clear rate of convergence for the algorithm and can show that it does not depend on the quantities $\mathcal{C}$ and $\mathcal{A}$.

4.2.1 Inexact Least-Squares Solves.

Algorithm 14 has a provably faster rate of convergence than Algorithm 12. The practical performance of these algorithms is compared in Figure 4.1, which is solving a problem with both a ℓ_1 norm regularizer and ℓ_1 norm BPDN constraint, with $\alpha = \|\mathcal{A}\|_F^{-2}$, $\mathcal{C} = I$, and $\nu_1 = \nu_2 = 10^{-4}$. We see a huge performance difference in practice as well as in theory: the proximal gradient descent from Algorithm 12 yields a slower cost function decay than solving exactly for $x(\mathbf{w})$ as in Algorithm 4. Indeed, Algorithm 14 admits the fastest cost

function decay as shown in Corollary 4.2.4, albeit at the expense of more operations per iteration. This is due to the fact that fully solving the least squares problem in Line 5 is not tractable for large-scale problems. Hence, we implement Algorithm 14 inexactly, using the Conjugate Gradient (CG) algorithm. Figure 4.1 shows the results when we use 1, 5, and 20 CG iterations. Each CG iteration is implemented using matrix-vector products, and at 20 iterations the results are indistinguishable from those of Algorithm 14 with full solves. Even at 5 iterations, the performance is remarkably close to that of of Algorithm 14 with full solves. Algorithm 14 has a natural warm-start strategy, with the x from each previous iteration used in the subsequent least-squares solve using CG. Using a CG method with a bounded number of iterates gives fast convergence and saves computational time. This approach is used in the subsequent experiments.

4.3 Application to Basis Pursuit Denoise Models

The Basis Pursuit Denoise problem can be formulated as

$$\min_x \varphi(x) \quad \text{s.t. } \psi\left(\mathcal{A}(x) - b\right) \leq \Delta \tag{4.9}$$

where $\psi(\cdot)$ is classically taken to be the ℓ_2 norm and the sparsity-inducing cost-functional $\varphi(\cdot)$ is often chosen to be $\|\cdot\|_1$, though more general norms/gauges are also used [143]. In this problem, x represents unknown coefficients that are sparse in a transform domain, while $\mathcal{A}$ is a composition of the observation operator with a transform matrix; popular examples of transform domains include discrete cosine transforms, wavelets, and curvelets. The observed and noisy data b resides in the temporal/spatial domain, and Δ is the misfit tolerance. This problem was famously solved with the SPGL1 [142] algorithm for $\psi(\cdot) = \|\cdot\|_2$ while minimizing $\|x\|_1$.

A nonsmooth variant of (4.9) is very difficult for approaches such as SPGL1, which solves subproblems of the form

$$\min_x \psi\left(\mathcal{A}(x) - b\right) \quad \text{s.t.} \quad \|x\|_1 \leq \tau.$$

and cannot handle nonsmooth functions, let alone the cardinality 'norm' $\psi(\cdot) = \ell_0(\cdot)$. When the observed data is affected by large sparse noise, this smooth constraint used by SPGL1 is ineffective. However, the proposed Algorithm 13 is easily adaptable to different norms/penalties in both cost-functional and constraint. We can solve Problem (4.9) by applying Algorithm 14 with $\varphi(x) = \|x\|_1$, taking $(\nu_1, \nu_2) \to (0, 0)$ so that $(w_1, w_2) \to (x, \mathcal{A}(x) - b)$. This means that the w_1 update in Algorithm 14 is the well-known proximal operator of ℓ_1, given by

$$\underset{\beta\|\cdot\|_1}{\text{prox}}(x) = \text{sign}(x)\max(0, |x| - \beta).$$

In addition, we know from [126] that the ℓ_1 norm can underestimate true signal values. To that end, our first example in section 4.4 also solves the sparsity-inducing cost-functional $\|x\|_0$, where our w_1 update in Algorithm 4 becomes

$$\underset{\beta\|\cdot\|_0}{\text{prox}}(x)_i = \begin{cases} 0 & |x_i| < \sqrt{2\beta}, \\ \{0, x_i\} & |x_i| = \sqrt{2\beta}, \\ x_i & |x_i| > \sqrt{2\beta} \end{cases} \tag{4.10}$$

or the *hard-thresholding* operator. We can take many different $\psi(\cdot)$, including $\ell_2, \ell_1, \ell_\infty$, and ℓ_0.

Algorithm 14 is simple to implement. The least squares update in step 4 can be computed efficiently using either factorization with Woodbury, or an iterative method in cases where $\mathcal{A}$ is too large to store.

For the Woodbury approach, we have

$$\left(\nu_2 + \nu_1 \mathcal{A}^T \mathcal{A}\right)^{-1} = \frac{1}{\nu_2} I - \frac{1}{\nu_2^2} \mathcal{A}^T \left(\frac{1}{\nu_1} I + \frac{1}{\nu_2} \mathcal{A}\mathcal{A}^T\right)^{-1} \mathcal{A}. \tag{4.11}$$

For moderate size systems, we can store Cholesky factor

$$LL^T = \frac{1}{\nu_1} I + \frac{1}{\nu_2} \mathcal{A}\mathcal{A}^T,$$

with $L \in \mathbb{R}^{m \times m}$, and use L with (4.11) to implement step 4. However, in the seismic/curvelet experiment described below, the left-hand side of (4.11) is too large to store in memory, but

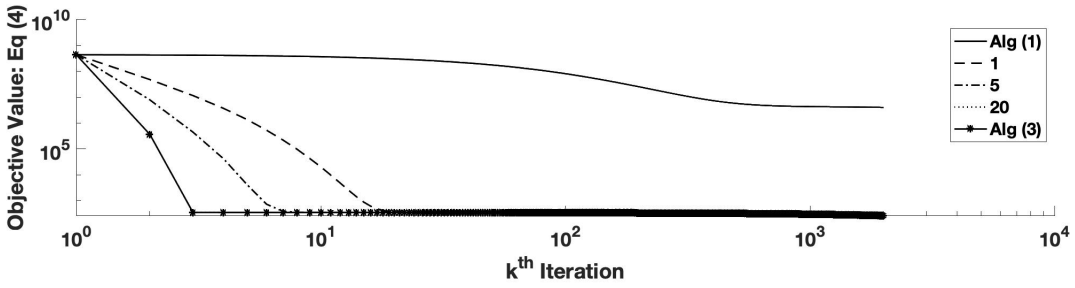


Figure 4.1: Objective function decay for (4.4) with proximal-gradient descent (Algorithm 12), Direct solving (Algorithm 4), and several steps in between where we only partially solve for $\mathcal{H}^{-1}(\ldots)$ with Algorithm 13.

is positive definite. Hence, we solve the resulting linear system in step 4 of Algorithm 14 with CG, using matrix-vector products. The w_1 update is implemented via the ℓ_1 proximal operator (soft thresholding), while the w_2 update requires a projection onto the ℓ_p ball. The projectors used in our experiments are collected in Table 4.1.

The least squares solve for x is when $\mathcal{C}^T$ is an orthogonal matrix or tight frame, so that $\mathcal{C}^T\mathcal{C} = I$; this is the case for Fourier transforms, wavelets, and curvelets. When $\mathcal{A}$ is a restriction operator, as for many data interpolation problems, $\mathcal{A}^T\mathcal{A}$ is a diagonal matrix with zeros and ones, and hence

$$\mathcal{H} = \frac{1}{\nu_1}\mathcal{C}^T\mathcal{C} + \frac{1}{\nu_2}\mathcal{A}^T\mathcal{A}$$

is a diagonal matrix with entries either $\frac{1}{\nu_1}$ or $\frac{1}{\nu_1} + \frac{1}{\nu_2}$; the least squares problem for the x update is then trivial.

Table 4.1: Projectors for ℓ_p balls.

Norm	$\ell(x)$	$\text{proj}_{\tau\mathbb{B}_\ell}(z)$	Solution
ℓ_2	$\sqrt{\sum_i x_i^2}$	$\begin{cases} z, & \|z\| < \tau \\ \tau z/\|z\|_2, & \|z\| > \tau \end{cases}$	Analytic
ℓ_∞	$\max_i \|x_i\|$	$\max(\min(x, 1), -1)$	Analytic
ℓ_1	$\sum_i \|x_i\|$	See e.g. [142]	$O(n \ln n)$
ℓ_0	$\sum_i \mathbf{1}_{x_i \neq 0}$	$\begin{cases} z_i, & i \text{ in } \tau \text{ largest indices} \\ 0 & \text{otherwise.} \end{cases}$	Analytic

4.4 Basis Pursuit De-Noise Experiments

In this application, we consider two examples: the first is a small-scale BPDN to illustrate the proof of concept of our technique, while the second is an application to denoising a common source gather extracted from a seismic line simulated using a 2D BG Compass model.

4.4.1 Spike-Train BPDN

The first example considers the same model as in (4.9) where we want to enforce sparsity on x while constraining the data misfit. The variable x is a vector of length n that has values $\{-1, 1\}$ on a random 4% of its entries and zeros everywhere else; represents a spike train that is acted upon by a linear operator, $A \in \mathbb{R}^{n,m}$. A was generated with independent standard Gaussian entries, and $b \in \mathbb{R}^m$ is observed data with large, sparse noise. We take $m = 120$ and $n = 512$. The noise is generated by placing large values on 10% of the observations and assuming everything else was observed cleanly (ie no noise). For this example, we first test the efficacy of using different ℓ_p norms on the residual constraint only, keeping the $\varphi(\cdot) = \|\cdot\|_1$. With the addition of large, sparse noise to the data, smooth norms on the residual constraint should not be able to effectively deal with such outlier residuals. With

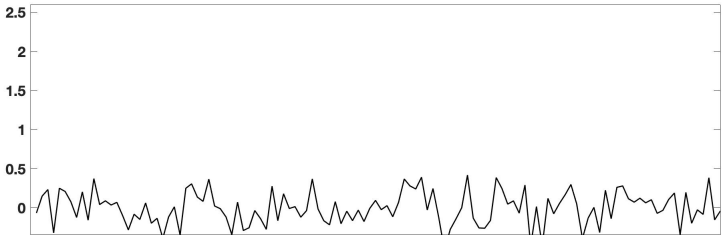
our adaptable formulation, it should be easy to enforce both sparsity in the x domain as well as the residuals. Other formulations, such as SPGL1, do not have this capability. We offer a comparable result to CVX with the linear program $\varphi(\cdot),\ \psi(\cdot) = \ell_1$.

True signal values, the transformed signal, and the observed data is given in Figure 4.2. The results of solving Problem (4.9) with cost-function $\varphi(\cdot) = \|\cdot\|_1$ are shown in Figure 4.3 and in Table 4.2. From these, we can clearly see that the ℓ_2 norm is not effective for sparse noise, even at the correct error budget Δ. Our approach is resilient to different types of noise since we can easily change the residual ball projection. This is seen by the accuracy of the ℓ_1 and ℓ_0 norms as choices for $\psi(\cdot)$, with SNR's of 33 and 45 respectively. This is comparable to solving $\varphi(\cdot), \psi(\cdot) = \ell_1(\cdot)$ with CVX, as implementation with that program yields an SNR of 35, as seen in Table 4.2. CVX, fails for larger examples given in subsequent sections.

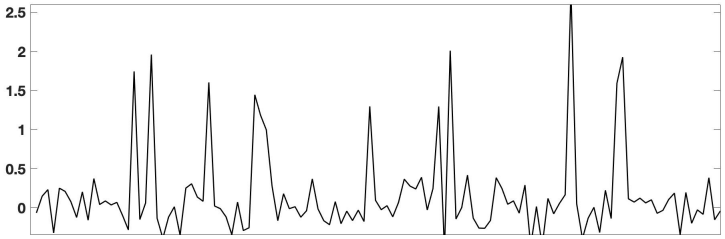
Table 4.2: SNR values against the true x for $\varphi(\cdot) = \ell_1$ and different $\psi(\cdot) = \ell_p$ norms with SPGL1, CVX, and Algorithm 14.

Spike-Train BPDN (4.9)	
$\psi(\cdot)$/Method	SNR
ℓ_2, SPGL1	0.2007
ℓ_2, Algorithm 14	0.2032
ℓ_1, CVX	35.3611
ℓ_1, Algorithm 14	33.7281
ℓ_∞, Algorithm 14	-0.6708
ℓ_0, Algorithm 14	45.0601

(a) True Signal: x

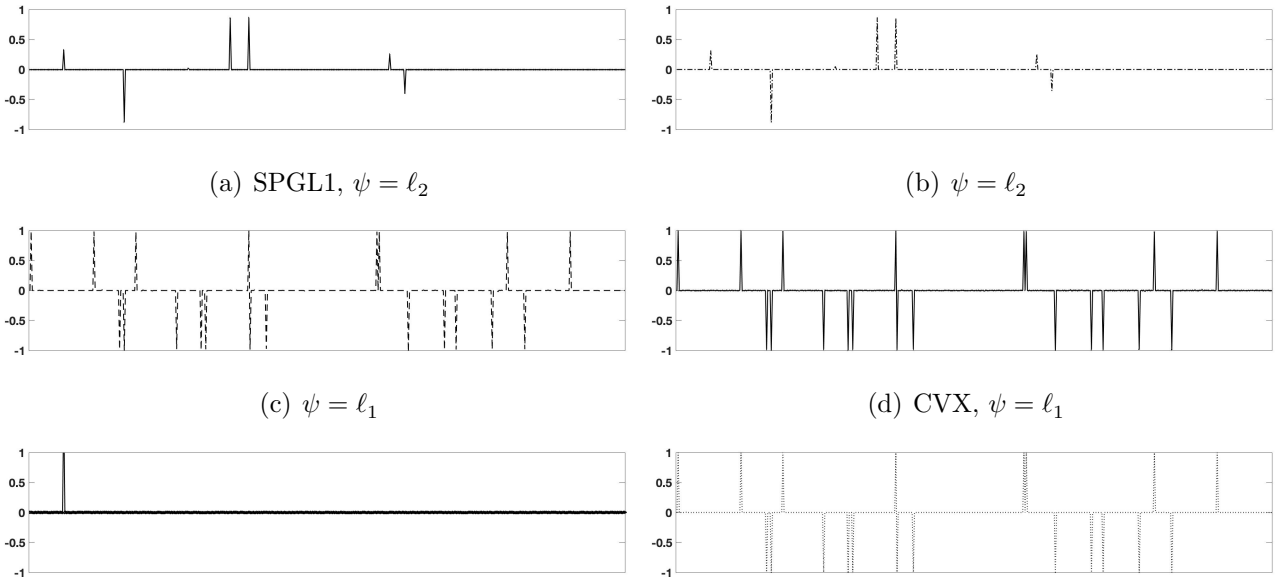


(b) True transformed signal: Ax



(c) Observed/noisy values: b

Figure 4.2: True signal, transformed signal, and noisy signals used in for the first experiment in section 4.4.



(a) SPGL1, $\psi = \ell_2$

(b) $\psi = \ell_2$

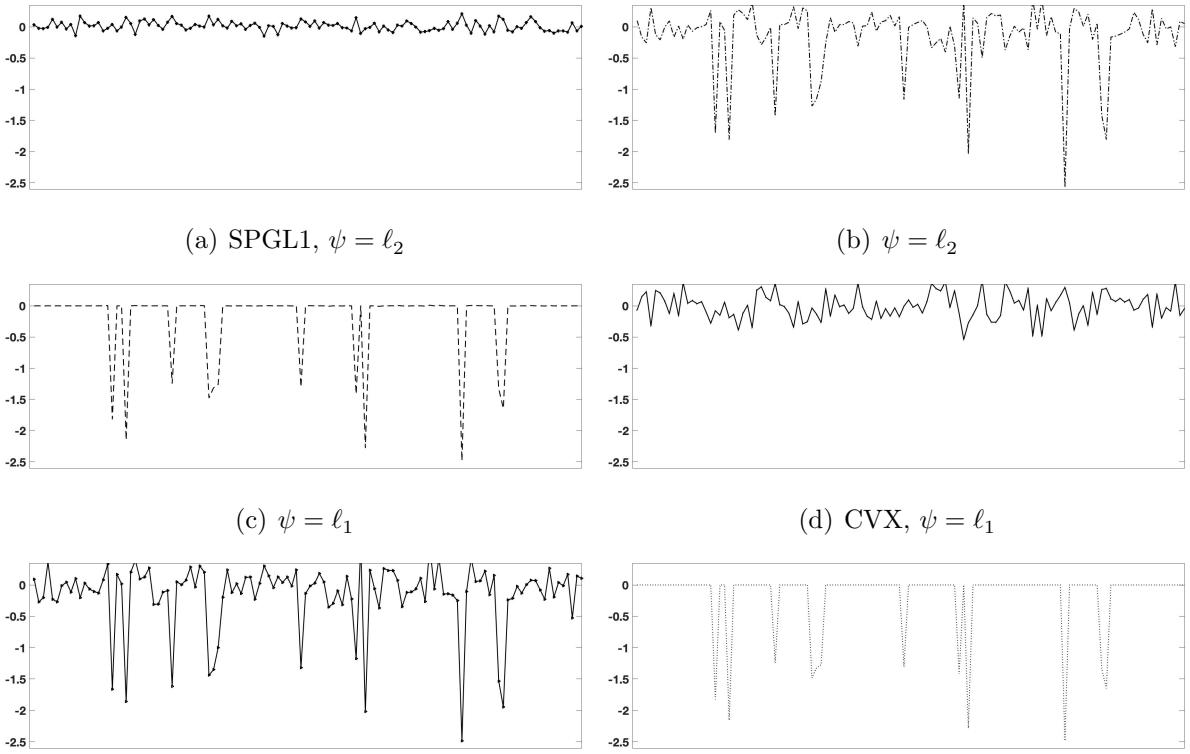
(c) $\psi = \ell_1$

(d) CVX, $\psi = \ell_1$

(e) $\psi = \ell_\infty$

(f) $\psi = \ell_0$

Figure 4.3: Basis Pursuit Denoising results on Problem (4.9) with cost-functional $\varphi(\cdot) = \|\cdot\|_1$ for a randomly generated linear model with large, sparse noise. Here, $\psi(\cdot)$ can be any of the ℓ_p norms in Table 4.1.



(a) SPGL1, $\psi = \ell_2$ (b) $\psi = \ell_2$

(c) $\psi = \ell_1$ (d) CVX, $\psi = \ell_1$

(e) $\psi = \ell_\infty$ (f) $\psi = \ell_0$

Figure 4.4: Residuals after algorithm termination for solving (4.9) where $\varphi(\cdot) = \|\cdot\|_1$, and using SPGL1 and Algorithm 14 with different $\psi(\cdot) = \ell_p$ norms. Note how the ℓ_1 and ℓ_0 norms can capture the outliers only.

Table 4.3: SNR values against the true x for different combinations of sparsity-inducing $\varphi(\cdot) = \ell_1,\ \ell_0$ and $\psi(\cdot) = \ell_2,\ \ell_0$ norms with SPGL1 and Algorithm 14.

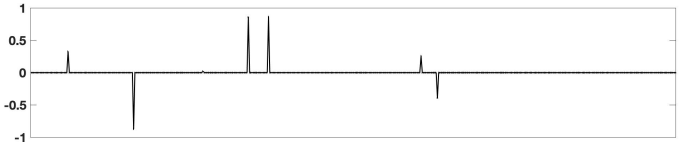
Spike-Train BPDN (4.9)	
$\varphi(\cdot)/\psi(\cdot)$/Method	SNR
ℓ_1 / ℓ_2, SPGL1	0.2007
ℓ_1 / ℓ_2, Algorithm 14	0.2031
ℓ_1 / ℓ_0, Algorithm 14	45.0440
ℓ_0 / ℓ_2, Algorithm 14	-1.2828
ℓ_0 / ℓ_0, Algorithm 14	44.4239

Secondly, we conduct a similar experiment, but test the resilience of our formulation to different sparsity-inducing metrics; this amounts to changing the $\varphi(\cdot)$ in the cost-function of Problem (4.9). Specifically, we change it to the $\ell_0(\cdot)$ norm (since we know we have to promote sparsity), and conduct a similar experiment as above - that is, we solve Problem (4.9), desiring a sparse signal from observed data with large sparse outliers. Hence, we want sparsity in *both* our signals and our observations. Note that this equations to setting w_1 as the proximal operator of the ℓ_0 norm, which is the hard-thresholding operator detailed in (4.10). Our results are shown in Table 4.3. We see that the in Table 4.3, typical Problem (4.9) with ℓ_1 norm solved with both Algorithm 14 and SPGL1 does not perform well. Indeed, even inducing sparsity with the $\ell_0(\cdot)$ cost-functional and $\psi(\cdot) = \ell_2(\cdot)$ in the constraints, we do not get the desired performance. However, inducing sparsity in the cost-functional (with both ℓ_1, and ℓ_0) gives us the most favorable results, with SNRs of 45 and 44 respectively. Recovered signals are shown in Figure 4.5.

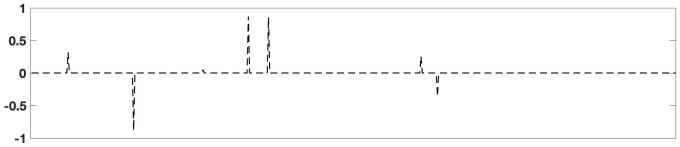
4.4.2 Curvelet Interpolation

The next test of the BPDN formulation is for a common source gather where entries are both omitted and corrupted with synthetic noise. The data set contains time samples with a temporal-interval of 4ms, and the spatial sampling is 10m. Here, the objective function looks for sparsity in the curvelet domain, while the residual constraint seeks to match observed data within a certain tolerance Δ. First, we note that doing interpolation only without added noise yields an SNR of approximately 13 for all formulations and algorithms; that is, all ℓ_p norms for Algorithm 14 and SPGL1. Here, we again want to enforce sparsity both in the curvelet domain ($\mathcal{C}(x)$) and the data residual ($\|\mathcal{A}(x) - b\|$), for which our algorithm is uniquely adapted to solve.

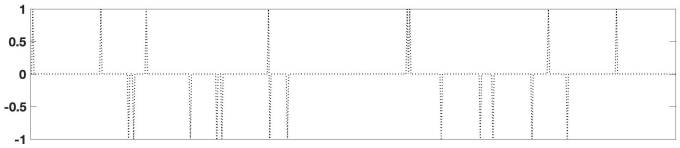
Following the first experiment of the spike-train example in section 4.4.1, we add large sparse noise to a handful of data points; in this case, we added large values to a random 1% of observations (this does not include omitted entries). The noise added is approximately 120, while the observed data can range from 0 to 30. The interpolated and denoising results



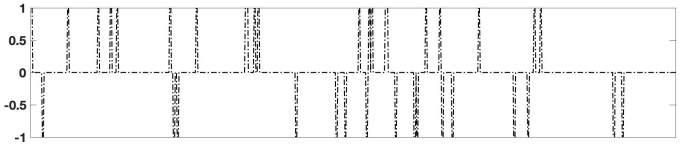
(a) SPGL1: $\varphi(\cdot) = \ell_1,\ \psi(\cdot) = \ell_2$



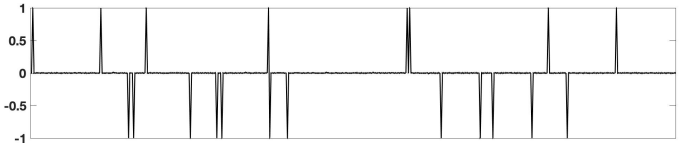
(b) Algorithm 14: $\varphi(\cdot) = \ell_1,\ \psi(\cdot) = \ell_2$



(c) Algorithm 14: $\varphi(\cdot) = \ell_1,\ \psi(\cdot) = \ell_0$



(d) Algorithm 14: $\varphi(\cdot) = \ell_0,\ \psi(\cdot) = \ell_2$



(e) Algorithm 14: $\varphi(\cdot) = \ell_0,\ \psi(\cdot) = \ell_0$

Figure 4.5: Sparse signal after algorithm termination for solving Problem (4.9) where $\varphi(\cdot) = \ell_1$, and using SPGL1 and Algorithm 14 with different $\psi(\cdot) = \ell_p$ norms. Note how the ℓ_1- and ℓ_0 norms can capture the outliers only.

Table 4.4: Curvelet Interpolation and Denoising results for SPGL1 and Algorithm 14 $\varphi(\cdot) = \ell_1$ and $\psi(\cdot) = \ell_p$ norms for BPDN (4.9).

Curvelet Interpolation & Denoising			
$\psi(\cdot)$/Method	SNR	SNR w_1	Time (s)
ℓ_2, SPGL1	1.4857	-	68.4 (early stoppage)
ℓ_2, Algorithm 14	0.9769	0.9693	6199
ℓ_1, Algorithm 14	14.9574	14.9436	5037
ℓ_∞, Algorithm 14	0.0000	0	1527
ℓ_0, Algorithm 14	14.9212	14.9142	6262

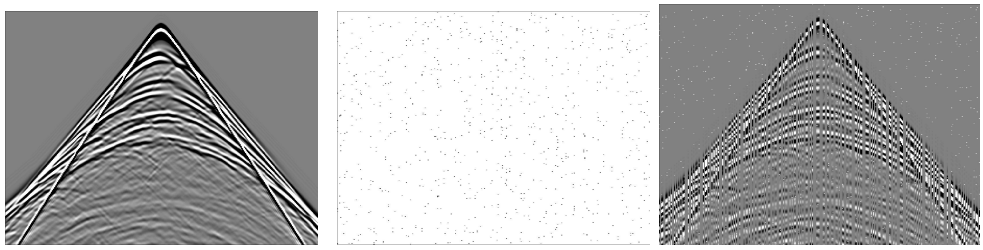
are shown in Figure 4.6 and Table 4.4. Large, sparse noise cannot be filtered effectively by a smooth norm constraint, using either Algorithm 14 or SPGL1. However, ℓ_1 and ℓ_0 norms effectively handle such noise, and can be optimized using our approach. The SNR's for these implementations are approximately 15 respectively, approaching that of the noiseless data mentioned above.

We then repeated the experiment done in section 4.4.1 where $\varphi(\cdot)$ is changed to be the ℓ_0 norm as well as the ℓ_1 norm. Similarly to Table 4.3, we have that sparsity in *both* $\varphi(\cdot)$ and $\psi(\cdot)$ can efficiently recapture the original signal. Table 4.5 shows that setting $\varphi(\cdot) = \ell_0$ together with sparse constraints (either ℓ_1 or ℓ_0) recapture the signal quite well; both have an SNR of approximately 14, over SNR's of close to 1 for every other combination.

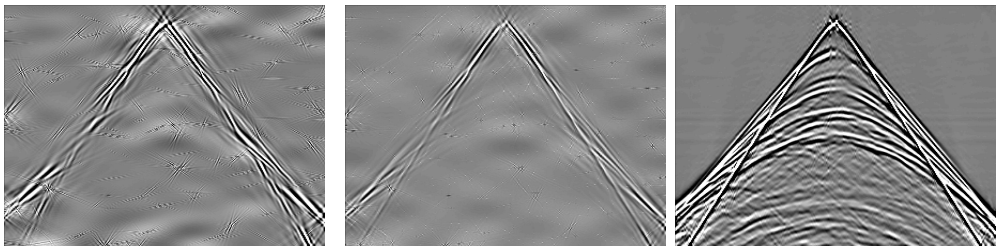
4.5 Extension to Low-Rank Models

Treating the data as having a matrix structure gives additional regularization tools — in particular low-rank structure in particular domains. The BPDN formulation for residual-constrained low-rank interpolation is given by

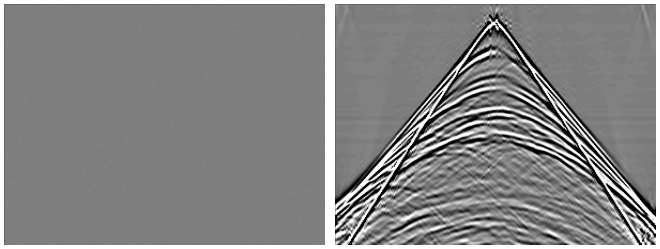
$$\min_X \|X\|_* \quad \text{s.t. } \psi\left(\mathcal{A}(X) - b\right) \leq \Delta \tag{4.12}$$



(a) True Data (b) Added Noise (binary) (c) Noisy Data with Missing Sources



(d) SPGL1 (e) $\psi = l_2$ (f) $\psi = l_1$



(g) $\psi = l_\infty$ (h) $\psi = l_0$

Figure 4.6: Interpolation and denoising results for BPDN in the curvelet domain. Observe the complete inaccuracy of smooth norms with large, sparse noise.

Table 4.5: Curvelet Interpolation and Denoising results for SPGL1 and Algorithm 14 with different combinations of sparsity-inducing $\varphi(\cdot) = \ell_1$, ℓ_0, and $\psi(\cdot) = \ell_2$, ℓ_0 norms for BPDN (4.9).

Curvelet Interpolation & Denoising			
$\varphi(\cdot)/\psi(\cdot)$/Method	SNR	SNR w_1	Time (s)
ℓ_1 / ℓ_2, SPGL1	1.4857	-	51.4 (early stoppage)
ℓ_1 / ℓ_2, Algorithm 14	0.9769	0.9693	4043
ℓ_1 / ℓ_0, Algorithm 14	14.9212	14.9142	4256
ℓ_0 / ℓ_2, Algorithm 14	0.1542	0.1199	4084
ℓ_0 / ℓ_0, Algorithm 14	14.042	13.7999	4086

for $X \in \mathbb{C}^{m\times n}$, $\mathcal{A} : \mathbb{C}^{n\times m} \to \mathbb{C}^p$ is a linear masking operator from full to observed (noisy) data b, and Δ is the misfit tolerance. The nuclear norm $\|X\|_*$ is the ℓ_1 norm of the singular values of X. Solving the problem (4.12) requires using a decision variable that is the size of the data, as well as updates to this variable that require SVDs at each iteration. It is much more efficient to model X is a product of two matrices L and R, given by

$$\min_{L,R} \frac{1}{2}(\|L\|_F^2 + \|R\|_F^2) \quad \text{s.t. } \psi\left(\mathcal{A}(LR^T) - b\right) \leq \Delta \tag{4.13}$$

where $L \in \mathbb{C}^{n\times k}$, $R \in \mathbb{C}^{m\times k}$, and LR^T is the low-rank representation of the data. The solution is guaranteed to be at most rank k, and in addition, the regularizer $\frac{1}{2}(\|L\|_F^2 + \|R\|_F^2)$ is an upper bound for $\|LR^T\|_*$, the sum of singular values of LR^T, further penalizing rank by proxy. The decision variables then have combined dimension $k(m \times n)$, which is much smaller than the nm variables required by convex formulations. When ψ is smooth, the problems are solved using a continuation that interchanges the roles of the objective and constraints, solving a sequence of problems where $\psi\left(\mathcal{A}(LR^T) - b\right)$ is minimized over the ℓ_2 ball [7] using projected gradient; an approach we call SPGLR below, which is a modification of SPGL1 specifically adapted for matrix completion.

When ψ is not smooth, SPGLR does not work and there are no available implementations for (4.13). Nonsmooth ψ arise when we want the residual to be in the ℓ_1 norm ball, so we are robust to outliers in the data, and can exactly fit inliers. We now extend Algorithm 14 to this case. For any ψ (smooth or nonsmooth), we introduce a latent variable W for the data matrix, and solve

$$\min_{L,R,W} \left\| \begin{matrix} L \\ R \end{matrix} \right\|_F^2 + \frac{1}{2\eta}\|W - LR^T\|_2^2, \quad \text{s.t. } \|\mathcal{A}(W) - b\|_p \leq \Delta \tag{4.14}$$

with η a parameter that controls the degree of relaxation; as $\eta \downarrow 0$ we have $W \to LR^T$. The relaxation allows a simple block-coordinate descent, detailed in the simple to implement Algorithm 15. It requires two least squares solves (for L and R), which are inherently parallelizable. It also requires a projection of the updated data matrix estimates LR^T onto the Δ-level set of the misfit penalty ψ.

Algorithm 15 Block-Coordinate Descent for (4.14).

Input: w_0, L_0, R_0

Initialize: $k = 0$

while not converged **do**

$$L_{k+1} \leftarrow \left(I + \eta R_k^T R_k\right)^{-1} (\eta W_k R_k)$$

$$R_{k+1} \leftarrow (\eta W_k^T L_{k+1}) \left(I + \eta L_{k+1}^T L_{k+1}\right)^{-1}$$

$$W_{k+1} \leftarrow \begin{cases} (L_{k+1}R_{k+1}^T)_{ij}, & (i,j) \in X_{obs} \\ \text{proj}_{\Delta \mathbb{B}_\psi}\left(\mathcal{A}(L_{k+1}R_{k+1}^T) - b\right), & \text{o.w.} \end{cases}$$

$$k \leftarrow k + 1$$

Output: w_k, L_k, R_k

For unobserved data $(i,j) \notin X_{obs}$, we have $W_{ij} = (LR^T)_{ij}$. For observed data, let a denote $\mathcal{A}(LR^T)$. Then the W update step is given by solving

$$\min_w \|w - a\|_2^2, \quad \text{subject to } \|w - b\|_p \leq \Delta.$$

Using the simple substitution $z = w - b$, the we get

$$\min_z \|z - (a - b)\|_2^2, \quad \text{subject to} \quad \|z\|_p \leq \Delta$$

which is precisely the projection of $\mathcal{A}(LR^T) - b$ onto $\Delta\mathbb{B}_\psi$, the Δ-level set of ψ. We use the same projectors for $\psi \in \{l_0, l_1, l_2, l_\infty\}$ as in section 4.4, see Table 4.1. The convergence criteria for Algorithm 15 is based on the optimality of the quadratic subproblems in L, R and feasibility measure of $W - LR^T$, though in practice we compare performance of algorithms based on a computational budget. This block-coordinate descent scheme converges to a stationary point of (4.14) by [139, Theorem 4.1].

Implementing block-coordinate descent on these forms until convergence produces the completed low-rank matrix. Setting $a = \|LR^T - w\|_2^2$, we iterate until $a < 10^{-5}$ or a maximum number of iterations is reached. In the next section, we develop an application of this method to seismic interpolation and denoising.

4.6 4D Matrix completion with Denoising

There are two main requirements when using the rank-minimization based framework for seismic data interpolation and denoising: *(i)* underlying seismic data should exhibit low-rank structure (singular values should decay fast) in some transform domain, and, *(ii)* subsampling and noise destroy the low-rank structure (singular values decay slow) in that domain. For exploiting the low-rank structure during interpolation and denoising, we follow the matricization strategy proposed by [43]. The matricization (source-x, source-y), i.e., placing both the source coordinates along the columns, gives slow-decay of singular values, while the matricization (source-x, receiver-x) gives fast decay of the singular values. Subsampling destroys the fast singular value decay in the (source-x, receiver-x) matricization, but not in the (source-x, receiver-y) matricization. Thus the latter is more effective for low-rank interpolation. These concepts are discussed in great detail by [73, 81].

Similar to the BPDN experiments, we want to show that nonsmooth constraints on the data residual can be effective for dealing with large, sparse noise. The smooth ℓ_2 norm that

is most common in BPDN problem will fail in such examples, thereby leading to better data estimation with the implementation of non-smooth norms on the residuals. Thus, the goal of the below experiments is to show that enforcing sparsity in the singular values (ie low-rank) and sparsity in the residual constraint can be more effective with large, sparse noise than smooth residual constraints solved by most contemporary algorithms.

4.6.1 Experiment Description

This example demonstrates the efficacy of the proposed approach using data created by a 5D dataset based on a complex SEG/EAGE overthrust model simulation [2]. The dimension of the model is $5\,\text{km} \times 10\,\text{km} \times 10\,\text{km}$ and is discretized on a $25\,\text{m} \times 25\,\text{m} \times 25\,\text{m}$ grid. The simulated data contains 201×201 receivers sampled at 50 m and 101×101 sources sampled at 100 m. We apply the Fourier transform along the time domain and extract a frequency slice at 10 Hz as shown in Figure 4.7(a), which is a 4D object (source-x, source-y, receiver-x and receiver-y). We eliminate 80% of the sources and add large sparse outliers from the random gaussian distribution $\mathcal{N}(0, a_i \max(X_{s_i}))$ (mean zero and variance on the order of the largest value in that particular source). The 10 generated values with the highest magnitudes are kept, and these are randomly added to observations in the remaining sources (Figure 4.7(f)). The largest value of our dataset is approximately 40, while the smallest is close to zero. Thus, we are essentially increasing/decreasing 1% of the entries by several orders of magnitude, which contaminates the data significantly, especially if the original entry was nearly 0. For all low-rank completion and denoising, we let $a_i = 10^{-1}$ except where we test the efficacy of Algorithm 15 against different noise levels Δ. The objective is to recover missing sources and eliminate noise from observed data. We use a rank of $k = 75$ for the formulation (that is, $L \in \mathbb{C}^{n \times 75}$ and similarly for R), and run all algorithms for 150 iterations, using a fixed computational budget. We perform three experiments on the same dataset: 1) Denoising only (Figure 4.7(c)); 2) Interpolation only (Figure 4.7(d)); and 3) Combined Interpolation and Denoising (Figure 4.7(f)). Since we have ground truth, we pick Δ to be the exact difference between generated noisy data and the true data; Δ for the l_0 norm is a cardinality

measure, so it is set to number of noisy points added.

4.6.2 Results

Table 4.6-Table 4.8 display SNR values for different algorithms and formulations for the three types of experiments, and Figure 4.8-Figure 4.10 display the results for a randomly selected number of sources for the three experiments. Even a small number of outliers can greatly impact the quality of the low-rank denoising and interpolation for the standard, smoothly residual-constrained algorithms. The denoising only results (Figure 4.8, Table 4.6) show that all methods perform well when all sources are available. The interpolation only results (Figure 4.9, Table 4.7) show that all constraints perform well in interpolating the missing data. This makes sense, as all algorithms will simply favor the low-rank nature of the data. However, the combined denoising and interpolation dataset shows that the ℓ_0 norm approach does far better than any smooth norm in comparable time. Table 4.8 shows that when data for similar sources is absent/not observed, the smoothly-constrained formulations fail completely. When noise is added to the low-amplitude section of the observed data, the smoothly-constrained norms fail drastically, while the ℓ_0 norm can effectively remove the errors. This is starkly evident in Figure 4.10(a)-Figure 4.10(e), where all except Figure 4.10(e) are essentially noise; the result is supported by the SNR values in Table 4.8. While Figure 4.10(a)-Figure 4.10(e) can mostly capture the structure of the data where there were nonzero values (ie where the seismic wave is observed in the upper left corner of each source), only the ℓ_0 norm can capture the areas of lower energy data.

Table 4.9-Table 4.10 give performance results across noise levels and degree of 'missingness' when using Algorithm 15. The combined problem, where a lot of data is omitted and outliers are present, is harder than either of the problems separately. We vary the percentage of sources omitted from 50% - 90%, and also vary the noise observed in two ways: (1) adding more noise while keeping number of outliers per source constant (nominally at 10), and (2) adding more noise by adding outliers at similar noise levels. The results are shown in Table 4.9 and Table 4.10. From Table 4.9, adding more noise to the outliers does not

Table 4.6: 4D Denoising results for SPGLR and Algorithm 15 for selected ℓ_p norms.

4D Monochromatic Denoising			
Method/$\psi(\cdot)$	SNR	SNR-W	Time (s)
ℓ_2 with SPGLR	11.7489	-	16530
l_2 with Algorithm 15	11.7463	-2.3338	9430
l_1 with Algorithm 15	11.7638	-2.3063	11546
l_∞ with Algorithm 15	11.7456	-2.3338	12108
l_0 with Algorithm 15	17.9595	48.8607	11569

Table 4.7: 4D Interpolation results for SPGLR and Algorithm 15 for selected ℓ_p norms.

4D Monochromatic Interpolation			
Method/$\psi(\cdot)$	SNR	SNR-W	Time (s)
ℓ_2 with SPGLR	16.3976	-	5817
l_2 with Algorithm 15	16.0629	16.5424	7526
l_1 with Algorithm 15	16.0692	16.5491	7996
l_∞ with Algorithm 15	16.0627	16.5423	8119
l_0 with Algorithm 15	16.0096	16.4728	6848

affect our results; the projection onto the ℓ_0 norm handles any obvious outlier. On the other hand, adding more outliers at varying magnitudes affects the results markedly. Increasing the number of outliers to 125 per source observed (which is roughly 1% of total data) affects performance of the ℓ_0 projection. From Table 4.10, performance with added noise alone degrades slowly, but once we start omitting sources, the outlier/normal observation ratio increases drastically for each 10% of sources withheld, we see decreased performance.

Table 4.8: 4D Combined Denoising and Interpolation results for SPGLR and Algorithm 15 for selected ℓ_p norms.

4D Monochromatic Denoising & Interpolation			
Method/$\psi(\cdot)$	SNR	SNR-W	Time (s)
ℓ_2 with SPGLR	-3.2906	-	8712
l_2 with Algorithm 15	0.9185	-0.3321	6802
l_1 with Algorithm 15	0.9193	-0.3235	8068
l_∞ with Algorithm 15	0.9185	-0.3321	8117
l_0 with Algorithm 15	16.0655	16.5445	6893

Table 4.9: 4D Interpolation (left), Denoising (center), and Combined (right) SNR results for Algorithm 15 with $\varphi(\cdot) = \ell_0$. The number of outliers is constant per source (10).

4D Monochromatic Denoising & Interpolation						
% Obs.	Int	Δ	DN	%	Δ	Both
50	17.7120	4.0e6	17.9597	50	2.1e6	17.7170
40	17.5445	5.2e7	17.9597	40	2.0e7	17.5459
30	17.2183	6.0e8	17.9597	30	1.8e8	17.2136
20	16.0522	6.9e9	17.9596	20	1.3e9	16.0263
10	9.2123	7.7e10	17.9596	10	7.8e9	9.2602

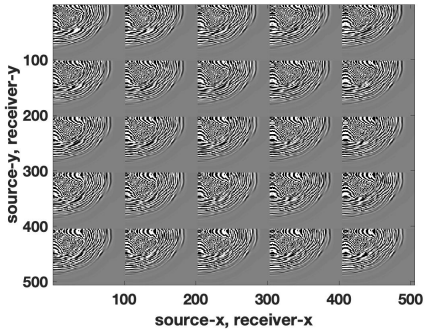
Table 4.10: 4D Interpolation (left), Denoising (center), and Combined (right) SNR results for Algorithm 15 with $\varphi(\cdot) = \ell_0$. The '#Out' column gives the number of outliers is per source.

4D Monochromatic Denoising & Interpolation							
% Obs.	Int	Δ	DN	%	Δ	# Out.	Both
50	17.712	2.2e7	17.955	50	2.2e7	5	17.714
40	17.544	1.6e8	5.055	40	1.6e8	35	1.157
30	17.218	3.4e8	7.629	30	3.4e8	65	-1.019
20	16.052	5.5e8	5.519	20	5.5e8	95	-4.588
10	9.212	7.9e8	3.927	10	7.9e8	125	-9.627

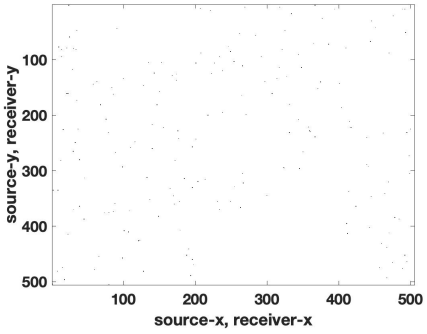
4.7 Conclusions

We proposed a new approach for level-set formulations, including basis pursuit denoise and residual-constrained low-rank formulations. The approach is easily adapted to a variety of nonsmooth and nonconvex data constraints. The resulting problems are solved using Algorithm 13 and Algorithm 15; which require only that the penalty ψ has an efficient projector. The algorithms are simple, scalable, and efficient. Sparse curvelet denoising and low-rank interpolation of a monochromatic slice from the 4D seismic data volumes demonstrate the potential of the approach.

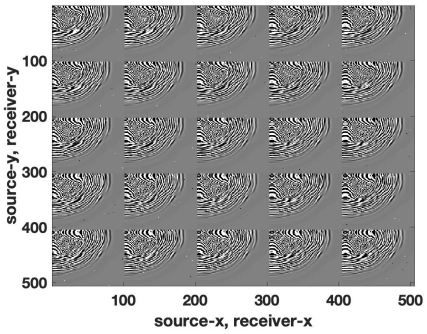
A particular quality of the seismic denoising and interpolation problem is that the amplitudes of the signal have significant spatial variation. The error in the data is a much larger problem for low-amplitude data. This quality makes it very difficult to obtain reasonable results using Gaussian misfits and constraints. Nonsmooth exact formulations (including ℓ_1 and particularly ℓ_0) appear to be extremely well-suited for this magnified heteroscedastic issue.



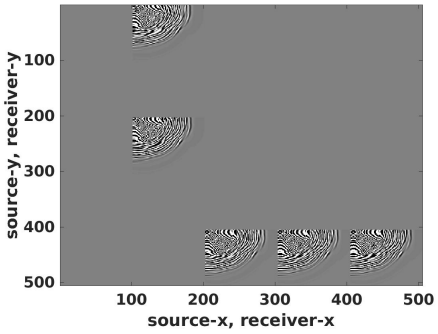
(a) Fully sampled monochromatic slize at 10 Hz.



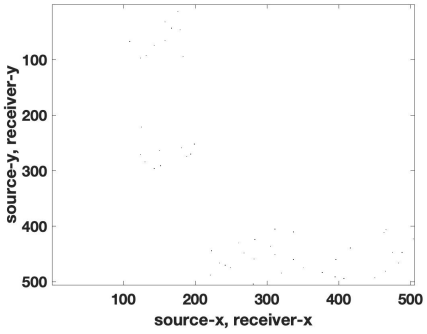
(b) Noisy data alone (binary). Selected top 10 entries from $\mathcal{N}(0, 0.1 \max(X_{s_i}))$



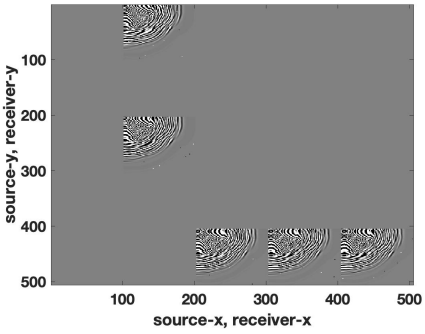
(c) Observed noisy data.



(d) Subsampled noiseless data. 80% of sources omitted.

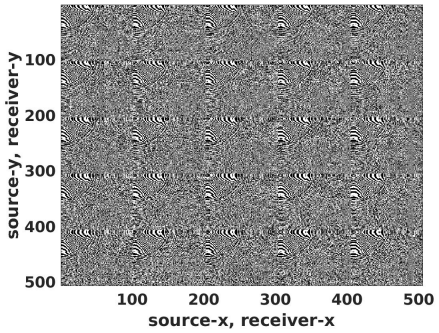


(e) Subsampled and noise, with noise only present (binary).

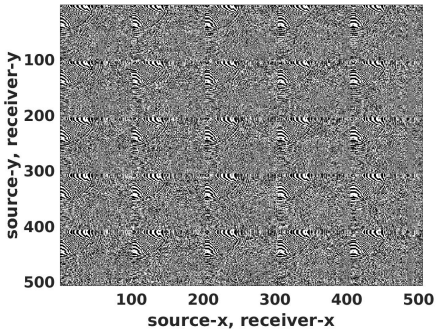


(f) Subsampled and noisy data. 80% of sources omitted and added noise.

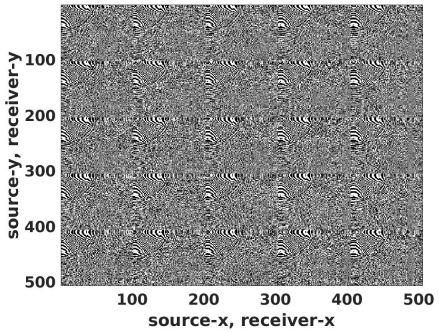
Figure 4.7: True data and three different experiments for testing our completeness algorithm.



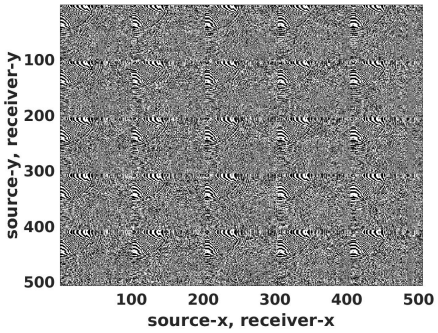
(a) SPGLR



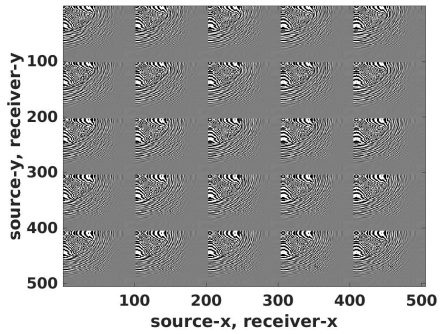
(b) $\psi = l_2$



(c) $\psi = l_1$

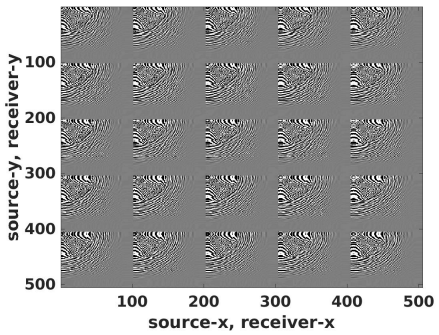


(d) $\psi = l_\infty$

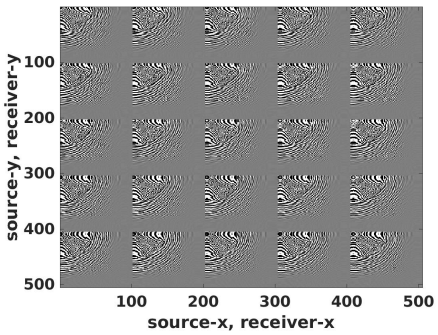


(e) $\psi = l_0$

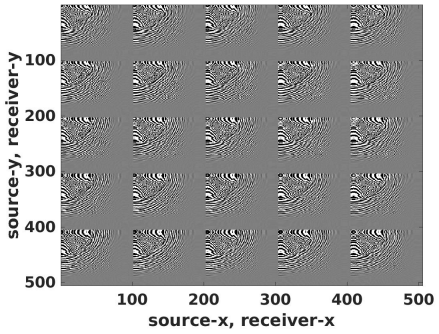
Figure 4.8: Denoising-only results.



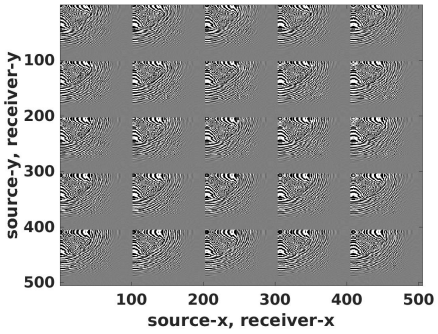
(a) SPGLR



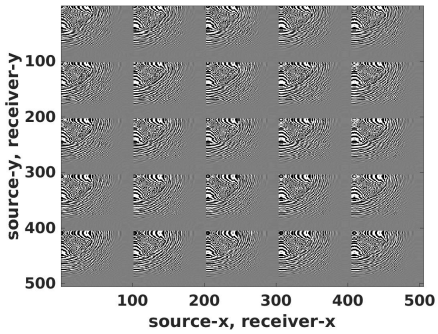
(b) $\psi = l_2$



(c) $\psi = l_1$

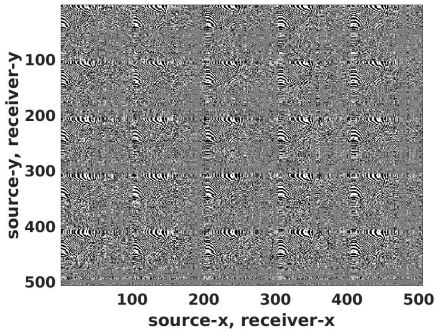


(d) $\psi = l_\infty$

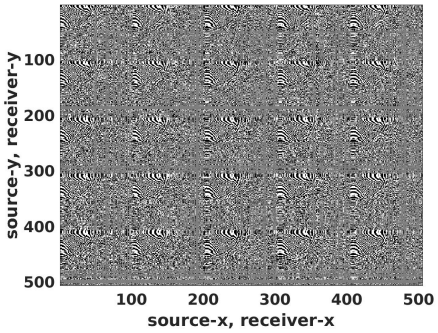


(e) $\psi = l_0$

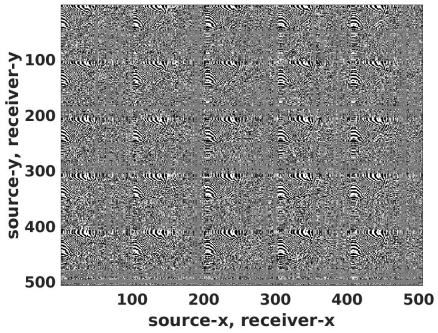
Figure 4.9: Interpolation-only results.



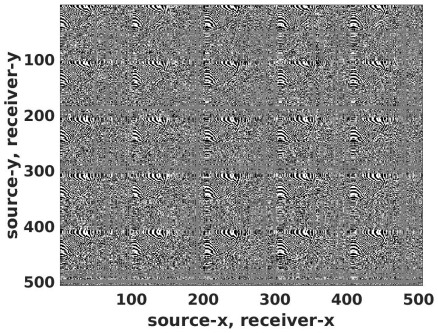
(a) SPGLR



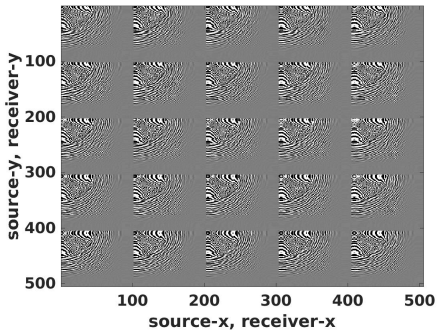
(b) $\psi = l_2$



(c) $\psi = l_1$



(d) $\psi = l_\infty$



(e) $\psi = l_0$

Figure 4.10: Interpolation and Denoising results.

Chapter 5

PROXIMAL QUASI-NEWTON TRUST-REGION METHOD FOR NONSMOOTH REGULARIZED OPTIMIZATION

5.1 Introduction

We consider the problem class

$$\underset{x}{\text{minimize}} \;\; f(x) + h(x), \tag{5.1}$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is continuously differentiable, $h : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ is proper and lower semi-continuous, and both may be nonconvex. Smooth and nonsmooth optimization problems are special cases corresponding to $h := 0$ and $f := 0$, respectively. Certain authors [36, 87] refer to (5.1) as a *composite* problem. We use instead the term *nonsmooth regularized* to differentiate with problems where $f = 0$ and $h(x) = g(c(x))$, where g is nonsmooth and c is smooth, which is indeed the composition of two functions. In practice, h is often a regularizer designed to promote desirable properties in solutions, such as sparsity. The class (5.1) captures the natural structure of a wide range of problems; problems with simple constraints, exact penalty formulations, basis selection problems with both convex [133, 142] and nonconvex [12, 24, 153] regularization, and more general inverse and learning problems [5, 26, 38].

We describe a trust-region method for (5.1) in which steps are computed by approximately minimizing simpler nonsmooth iteration-dependent models inside a trust region defined by an arbitrary norm. In practice, the norm is chosen based on the nonsmooth term in the model and the tractability of the step-finding subproblem, which is not required to be convex. Our analysis hinges on the observation that in the nonsmooth context, the first step of the proximal gradient method is the right generalization of the gradient step in smooth optimization. We establish global convergence in terms of an optimality measure describing the decrease achievable in the model by a single step of the proximal gradient method inside the trust-region. We also establish a worst-case complexity bound of $O(1/\epsilon^2)$ iterations to bring this optimality measure below a tolerance $0 < \epsilon < 1$. Others [36, 59] have observed that it is possible to devise trust-region methods for regularized optimization with complexity equivalent to that for smooth optimization. However, past research typically assumes that h is either globally Lipschitz continuous and/or convex.

We also revisit a quadratic regularization method, and establish similar convergence properties and same worst-case compexity under the same assumptions. Our description highlights the connection between the quadratic regularization method and the standard proximal gradient method. The former may be seen as an implementation of the latter with adaptive step size.

We provide implementation details and illustrate the performance of an instance where the trust-region model is the sum of a limited-memory quasi-Newton, possibly nonconvex, approximation of f with a nonsmooth model of h and various choices of the trust-region norm. Our trust-region algorithm exhibits promising performance and compares favorably with linesearch proximal quasi-Newton methods based on convex models [129, 132]. Our open source implementations are available from `github.com/UW-AMO/TRNC` as packages in the emerging Julia programing language [22].

As far as we can tell from the literature, the method described in the present paper is the first trust-region method for the fully nonconvex nonsmooth regularized problem. Our approach offers flexibility in the choice of the norm used to define the trust-region, provided an efficient procedure is known to solve the subproblem. We show that such procedures are easily obtained in a number of applied scenarios.

Related research

We focus on (5.1) and do not provide an extensive review of approaches for smooth optimization. Conn et al. [40] cover trust-region methods for smooth optimization thoroughly, as well as a number of select generalizations, and we refer the reader to their comprehensive treatment for background.

Yuan [149] formulates conditions for convergence of trust-region methods for convex-composite objectives, i.e., $g(c(x))$ where c is continuously differentiable and g is convex. In particular, he considers models of the form $s \mapsto g(c(x) + \nabla c(x)s)$, that are relevant to exact penalty methods for constrained optimization, and that are a special case of the models we consider.

Dennis et al. [47] develop convergence properties of trust-region methods for the case where $f = 0$ and h is Lipschitz continuous. Their analysis is based on a generalization of the concept of Cauchy point in terms of Clarke directional derivatives, but they do not provide an approach to solve the typically nonsmooth subproblem. Kim et al. [75] analyze a trust-region method for (5.1) when f is convex and h is continuous and convex with assumptions based on those of Dennis et al. [47]. Their model around a current x has the form $f(x) + \nabla f(x)^T s + \frac{1}{2}\alpha\|s\|^2 + h(x+s)$, where α is a Barzilai-Borwein step length safeguarded to stay sufficiently positive and bounded. By contrast, our approach allows general quadratic models, possibly indefinite, and explicitly accounts for the trust-region constraint in the subproblem by devising specialized proximal operators.

Qi and Sun [114] propose a trust-region method inspired by that of Dennis et al. [47] for the case where $f = 0$ and h is locally Lipschitz continuous with bounded level sets. They establish convergence under the further assumption that the models are $[0, 1]$-subhomogeneous. Martínez and Moretti [100] employ similar assumptions to generalize the approach to problems with linear constraints.

Cartis et al. [36] consider (5.1) where h is convex and globally Lipschitz continuous. They analyze both a trust-region algorithm and a quadratic regularization variant, develop convergence and iteration complexity results, but do not provide guidance on how to compute steps in practice. Their analysis revolves around properties of a stationarity measure that are strongly anchored to the convexity assumption. The algorithms that we develop below are most similar to theirs but rest upon significantly weaker assumptions and concrete subproblem solvers. Grapiglia et al. [59] detail a unified convergence theory for smooth optimization that has trust-region methods as a special case. They also generalize the results of [36] but focus on objectives of the form $f(x) + g(c(x))$ where f and c are smooth and g is convex and globally Lipschitz.

Lee et al. [87] fully explore the global and fast local convergence properties of exact and inexact proximal Newton and quasi-Newton methods for the case where both f and h are convex. They show that those methods inherit all the desired properties of their counterparts

in smooth optimization.

Bolte et al. [26] present a proximal alternating method for objectives of the form $g(x) + Q(x, y) + h(y)$ where g and h are proper and lower semi-continuous and the coupling function Q is continuously differentiable. Their setting has (5.1) as a special case. They establish convergence under the Kurdyka-Łojasiewicz assumption and provide a general recipe for algorithmic convergence under such an assumption.

Li and Lin [89] consider monotone and non-monotone accelerations of the proximal gradient method for possibly nonconvex f and h. They establish global convergence under the assumptions that f has a Lipschitz continuous gradient, h is proper and lower semi-continuous, and that $f + h$ is coercive. This leads to a sublinear iteration complexity bound when a Kurdyka-Łojasiewicz condition holds. Boţ et al. [30] employ an inertial acceleration strategy which converges under the assumptions that h is bounded below and possesses a Kurdyka-Łojasiewicz condition.

Stella et al. [129] initially devised PANOC, a linesearch quasi-Newton method for (5.1) with limited-memory BFGS Hessian approximations, for model predictive control. PANOC assumes that the objective has the form $f(x) + h_1(x) + h_2(c(x))$, where f and c are smooth, h_1 is nonsmooth and may be nonconvex, and h_2 is nonsmooth and convex. Themelis et al. [132] develop ZeroFPR, a nonmonotone linesearch proximal quasi-Newton method for (5.1) based on the concept of forward-backward envelope. ZeroFPR converges under a Kurdyka-Łojasiewicz assumption and enjoys the fast local convergence properties of quasi-Newton methods for smooth optimization when a Dennis-Moré condition holds.

Notation

Sets are represented by calligraphic letters. The cardinality of set $\mathcal{S}$ is represented by $|\mathcal{S}|$. We use $\|\cdot\|$ to denote a generic norm on $\mathbb{R}^n$. The symbols ν, λ, σ and Δ are scalars. $\mathbb{B}(0, \Delta)$ is the ball centered at 0 with radius $\Delta > 0$ defined by a norm that should be clear from the context. We use the shorthands $\mathbb{B} = \mathbb{B}(0, 1)$ and $\Delta\mathbb{B} = \mathbb{B}(0, \Delta)$. When necessary, we write $\mathbb{B}_p$ to indicate that the ℓ_p-norm is used. Functional symbols f, g, h, as well as ϕ, φ and ψ

are used for functions. $\chi(\cdot; A)$ represents the indicator function of $A \subseteq \mathbb{R}^n$. In particular, the indicator of $\mathbb{B}(0, \Delta)$ is denoted $\chi(\cdot; \Delta\mathbb{B})$ or just $\chi(\cdot; \Delta)$ when the norm is clear from the context. We use the alternative notation $\chi(\cdot; \Delta\mathbb{B}_p)$ to emphasize that the ℓ_p-norm is used to define the ball. If $A \subseteq \mathbb{R}^n$ is closed and convex, $\text{proj}_A(x)$ denotes the projection of x into A. Finally, j and k are iteration counters.

Roadmap

The paper proceeds as follows. section 5.2 develops the general trust-region method for (5.1), including the new Algorithm 16, and introduces several innovations that yield the main results. In section 5.3, we explain how to compute a trust-region step based on a proximal quasi-Newton model. New relevant proximal operators needed to implement the trust-region method are studied in section 5.4. A quadratic regularization variant of the trust-region algorithm together with its convergence analysis are presented in section 5.5. Numerical results and experiments are in section 5.6. We end with a brief discussion in section 5.7.

5.2 Trust-region methods for nonsmooth regularized optimization

In this section, we develop and analyze a general trust-region method for (5.1). Section 5.2.1 examines properties of trust-region subproblems. Section 5.2.2 discusses optimality measures, and highlights the role of the prox-gradient step in quantifying descent in the general context of (5.1). In Section 5.2.3, we present the trust-region approach, and highlight key innovations that make it possible to obtain the convergence results and complexity analysis presented in Section 5.2.4.

5.2.1 Properties of trust-region subproblems

For fixed $x \in \mathbb{R}^n$, consider the parametric problem and its optimal set

$$p(\Delta; x) := \underset{s}{\text{minimize}} \;\; \varphi(s; x) + \psi(s; x) + \chi(s; \Delta), \tag{5.2a}$$

$$P(\Delta; x) := \arg\min_{s} \;\; \varphi(s; x) + \psi(s; x) + \chi(s; \Delta), \tag{5.2b}$$

where $\varphi(s; x) \approx f(x + s)$, $\psi(s; x) \approx h(x + s)$, $\chi(s; \Delta)$ is the indicator function of the trust region $\Delta\mathbb{B}$ and $\Delta > 0$. The form of (5.2) is representative of a trust-region subproblem for (5.1) in which f and h are modeled separately and the trust-region constraint appears implicitly via an indicator function.

We make the following additional assumption.

Model Assumption 5.2.1. *For any $x \in \mathbb{R}^n$, $\varphi(\cdot; x)$ is continuously differentiable, $\psi(\cdot; x)$ is proper and lsc.*

By Proposition 2.2.3,

$$s \in P(\Delta; x) \quad \Longrightarrow \quad 0 \in \nabla\varphi(s; x) + \partial(\psi(\cdot; x) + \chi(\cdot; \Delta))(s).$$

The following result summarizes properties of (5.2).

Proposition 5.2.1. *Let Model Assumption 5.2.1 be satisfied. If we define $p(0; x) := \varphi(0; x) + \psi(0; x)$ and $P(0; x) = \{0\}$, the domain of $p(\cdot; x)$ and $P(\cdot; x)$ is $\{\Delta \mid \Delta \geq 0\}$. In addition,*

1. *$p(\cdot; x)$ is proper lsc and for each $\Delta \geq 0$, $P(\Delta; x)$ is nonempty and compact;*

2. *if $\{\Delta_k\} \to \bar{\Delta} \geq 0$ in such a way that $\{p(\Delta_k; x)\} \to p(\bar{\Delta}; x)$, and for each k, $s_k \in P(\Delta_k; x)$, then $\{s_k\}$ is bounded and all its limit points are in $P(\bar{\Delta}; x)$;*

3. *if $\varphi(\cdot; x) + \psi(\cdot; x)$ is strictly convex, $P(\Delta; x)$ is single-valued;*

4. *if $\bar{\Delta} > 0$ and there exists $\bar{s} \in P(\bar{\Delta}; x)$ such that $\|\bar{s}\| < \bar{\Delta}$, then $p(\cdot; x)$ is continuous at $\bar{\Delta}$ and $\{p(\Delta_k; x)\} \to p(\bar{\Delta}; x)$ holds in part 2.*

Proof. Model Assumption 5.2.1 and compactness of the trust region ensure that the objective of (5.2a) is always level-bounded in s locally uniformly in Δ [120, Definition 1.16] because for any $\bar{\Delta} > 0$ and $\epsilon > 0$, and for any $\Delta \in (\bar{\Delta} - \epsilon, \bar{\Delta} + \epsilon)$ with $\Delta \geq 0$, the level sets of $\varphi(\cdot; x) + \psi(\cdot; x) + \chi(\cdot; \Delta)$ are contained in $\Delta\mathbb{B} \subseteq (\bar{\Delta} + \epsilon)\mathbb{B}$. Parts 1–2 follow by Rockafellar and Wets [120], Theorems 1.17 and 7.41. Part 3 follows from Rockafellar and Wets [120, Exercice 7.45]. Part 4 follows by noting that if $\|\bar{s}\| < \bar{\Delta}$, then $\varphi(\bar{s}; x) + \psi(\bar{s}; x) + \chi(\bar{s}; \Delta)$ is continuous in Δ in a neighborhood of $\bar{\Delta}$; the rest follows from Rockafellar and Wets [120, Theorem 1.17*c*]. □

It is not necessary to assume that $\psi(\cdot; x)$ is prox-bounded in Model Assumption 5.2.1 because under the assumptions stated and compactness of the trust region, the objective of (5.2a) is necessarily bounded below, and therefore prox-bounded. Proposition 5.2.1 allows us to think of how approximate solutions "truncated" by a trust-region constraint approach $\bar{s}$ as the trust-region radius increases. Indeed, we may choose any $\bar{\Delta} > \|\bar{s}\|$ in parts 2 and 4. When $\psi(\cdot; x) = 0$ and $\varphi(\cdot; x)$ is quadratic and strictly convex, the graph of $P(\cdot; x)$ is known to be a smooth curve such that $P(0; x) = \{x_k\}$, that is tangential to $-\nabla f(x_k)$ at $\Delta = 0$ and such that $\lim_{\Delta \to \infty} P(\Delta; x)$ contains the Newton step as its only element. This observation gives rise to several numerical methods to approximate the solution of (5.2), including the dogleg [113] and double dogleg methods [48].

5.2.2 *Optimality measures*

In this section, we seek a convenient way of assessing whether a given x is first-order critical for (5.1) based on the trust-region subproblem (5.2). We begin with the following result.

Proposition 5.2.2. *Let Model Assumption 5.2.1 be satisfied. Assume in addition that* $\nabla_s\varphi(0;x) = \nabla f(x)$, $\partial\psi(0;x) = \partial h(x)$, *and let* $\Delta > 0$. *Then* $0 \in P(\Delta;x) \iff s = 0$ *is first-order stationary for* (5.2) $\iff x$ *is first-order stationary for* (5.1).

Proof. By definition, x is first-order stationary if and only if $0 \in \nabla f(x)+\partial h(x) = \nabla_s\varphi(0;x)+\partial\psi(0;x)$. But $\psi(0;x) = \psi(0;x) + \chi(0;\Delta)$ and $\partial(\psi(\cdot;x) + \chi(\cdot;\Delta))(0) = \partial\psi(0;x) + \partial\chi(0;\Delta)$ because $\partial\chi(0;\Delta) = \{0\}$. Thus we obtain $0 \in \nabla_s\varphi(0;x) + \partial(\psi(\cdot;x) + \chi(\cdot;\Delta))(0)$, i.e., $s = 0$ is first-order stationary for (5.2). □

Proposition 5.2.2 suggests we may use an element of $P(\Delta;x)$ as first-order optimality measure for any $\Delta > 0$, such as for example $\|g(\Delta;x)\|$, where $g(\Delta;x)$ is the least-norm element of $P(\Delta;x)$. However, the dependency on Δ is inconvenient. In order to circumvent this difficulty, we focus our attention temporarily on the choice

$$\begin{aligned}\varphi(s;x) &= f(x) + \nabla f(x)^T s + \tfrac{1}{2}\nu^{-1}\|s\|^2 \\ &= \tfrac{1}{2}\nu^{-1}\|s + \nu\nabla f(x)\|^2 + f(x) - \tfrac{1}{2}\nu\|\nabla f(x)\|^2,\end{aligned} \tag{5.3}$$

where $\nu > 0$ is fixed, so that for any $x \in \mathbb{R}^n$,

$$p(\Delta;x,\nu) = e_{\nu\psi(\cdot;x)+\chi(\cdot;\Delta)}(-\nu\nabla f(x)) + f(x) - \tfrac{1}{2}\nu\|\nabla f(x)\|^2, \tag{5.4a}$$

$$P(\Delta;x,\nu) = \operatorname*{prox}_{\nu\psi(\cdot;x)+\chi(\cdot;\Delta)}(-\nu\nabla f(x)), \tag{5.4b}$$

and p only differs from a Moreau envelope by a constant. The above choice of $\varphi(\cdot;x)$ allows us to derive a convenient, computable optimality measure, and to generalize the concept of decrease along the steepest descent direction, also known as Cauchy decrease, which is so fundamental to the convergence analysis of computational methods for smooth optimization.

In the special case where $\psi(\cdot;x) = 0$, Proposition 5.2.1 part 3 indicates that $P(\Delta;x,\nu)$ is single valued, and its only element is the projection of $-\nu\nabla f(x)$ into the trust region. On

the other hand, $p(\Delta; x, \nu)$ measures the decrease of (5.3) in the direction of the projected gradient. Cartis et al. [36] study the special case where $h(x) = g(c(x))$ with g convex and globally Lipschitz continuous, and c smooth. In lieu of (5.4a), they minimize $f(x) + \nabla f(x)^T s + g(c(x) + \nabla c(x)^T s)$ in the trust region, which is analogous.

Crucially, (5.4) describes the first step of the proximal gradient method with step size ν applied to (5.2a) where $\varphi(\cdot; x)$ is as in (5.3) from $s = 0$ with a trust region of radius Δ. In the notation of section 2.3.1, φ is $\varphi(\cdot; x)$ and ψ is $\psi(\cdot; x) + \chi(\cdot; \Delta)$. If $\psi(\cdot; x)$ is finite at $s_0 = 0$, the first step of the proximal gradient method is

$$\begin{aligned} s_1 &\in \arg\min_s \ \tfrac{1}{2}\nu^{-1}\|s + \nu\nabla f(x)\|^2 + \psi(s; x) + \chi(s; \Delta) \\ &= \arg\min_s \ f(x) + \nabla f(x)^T s + \tfrac{1}{2}\nu^{-1}\|s\|^2 + \psi(s; x) + \chi(s; \Delta), \end{aligned} \tag{5.5}$$

and yields the decrease

$$(\varphi + \psi)(s_1; x) \leq (f + h)(x) - \tfrac{1}{2}(\nu^{-1} - L)\|s_1\|^2 \tag{5.6}$$

Moreover, s_1 is also the first step of the proximal-gradient method applied to (5.2a) where $\varphi(\cdot; x)$ is *any* model of f about x that is differentiable at $s = 0$ with $\nabla_s \varphi(0; x) = \nabla f(x)$, and, in particular, any quadratic expansion of f about x. In the sequel, we use s_1 as the appropriate generalization to the nonsmooth context of the projected gradient step, which allows us to derive an adequate optimality measure.

Let

$$\xi(\Delta; x, \nu) := f(x) + h(x) - p(\Delta; x, \nu), \tag{5.7}$$

where $p(\Delta; x, \nu)$ is defined in (5.4a). In view of the above, $\xi(\Delta; x, \nu)$ measures the decrease predicted by the first step of the proximal gradient method applied to (5.2a) from $s = 0$ with trust-region radius Δ and step length $\nu > 0$, where $\varphi(\cdot; x)$ is any model of f about x that is differentiable at $s = 0$ with $\nabla_s \varphi(0; x) = \nabla f(x)$.

Assume from now on that $\varphi(0; x) = f(x)$ and $\psi(0; x) = h(x)$. Because $p(\Delta; x, \nu) \leq \varphi(0; x) + \psi(0; x) + \chi(0; \Delta) = f(x) + h(x)$, we necessarily have $\xi(\Delta; x, \nu) \geq 0$.

Examples of models of f satisfying the above assumptions include Taylor expansions of f about x, and in particular quadratic models $f(x)+\nabla f(x)^T s+\frac{1}{2}s^T Bs$ where $B=B^T$. The most straightforward example of a model of h satisfying the above is $\psi(s;x)=h(x+s)$. If $h(x)=g(c(x))$, where $g:\mathbb{R}^m\to\overline{R}$ is proper, lsc and level-bounded, and $c:\mathbb{R}^n\to\mathbb{R}^m$ is continuously differentiable, other possible models include $\psi(s;x)=g(c(x)+\nabla c(x)^T s)$ and $\psi(s;x)=g(c(x)+\nabla c(x)^T s+\sum_{i=1}^m s^T B_i s)$, where each $B_i=B_i^T$.

The following result allows us to rely on the computable values $p(\Delta;x,\nu)$ and $\xi(\Delta;x,\nu)$ to assess stationarity.

Proposition 5.2.3. *Let Model Assumption 5.2.1 be satisfied where $\varphi(0;x)=f(x)$ and $\nabla_s\varphi(0;x)=\nabla f(x)$. Assume furthermore that $\psi(0;x)=h(x)$ and $\partial\psi(0;x)=\partial h(x)$, and let $\Delta>0$. Then, $\xi(\Delta;x,\nu)=0\iff 0\in P(\Delta;x,\nu)\implies x$ is first-order stationary for (5.1).*

Proof. $\xi(\Delta;x,\nu)=0$ if and only if $p(\Delta;x,\nu)=f(x)+h(x)=\varphi(0;x)+\psi(0;x)+\chi(0;\Delta)$, which occurs if and only if $0\in P(\Delta;x,\nu)$. Proposition 5.2.2 then implies that x is first-order stationary for (5.1). □

5.2.3 A trust-region algorithm

We focus on the solution of (5.1) under Problem Assumption 5.2.1.

Problem Assumption 5.2.1. *In (5.1), $f\in\mathcal{C}^1(\mathbb{R}^n)$, and h is proper and lsc.*

At iteration k, we construct a model $m_k(s;x_k):=\varphi(s;x_k)+\psi(s;x_k)\approx f(x_k+s)+h(x_k+s)$ and we approximately solve

$$\underset{s}{\text{minimize}}\ m_k(s;x_k)\quad\text{subject to}\ \|s\|\le\Delta_k \tag{5.8}$$

by computing a step s_k required to result in at least a fraction of the decrease achieved with one step of the proximal gradient method. Step Assumption 5.2.1 formalizes our requirement.

Step Assumption 5.2.1. *There exists* $\kappa_{\text{m}} > 0$ *and* $\kappa_{\text{mdc}} \in (0, 1)$ *such that for all* k, $\|s_k\| \leq \Delta_k$ *and*

$$|f(x_k + s_k) + h(x_k + s_k) - m_k(s_k; x_k)| \leq \kappa_{\text{m}} \|s_k\|^2, \tag{5.9a}$$

$$m_k(0; x_k) - m_k(s_k; x_k) \geq \kappa_{\text{mdc}} \xi(\Delta_k; x_k, \nu_k), \tag{5.9b}$$

where m_k *is defined above and* $\xi(\Delta_k; x_k, \nu_k)$ *is defined in* (5.7).

Condition (5.9a) is certainly satisfied if both f and φ are twice continuously differentiable with bounded second derivatives, and $\psi(s; x_k) := h(x_k+s)$. It also holds when $h(x) = g(c(x))$ where $c : \mathbb{R}^n \to \mathbb{R}^m$ has Lipschitz-continuous Jacobian and $g : \mathbb{R}^m \to \mathbb{R}^n$ is Lipschitz continuous. Such a situation arises when (5.1) results from penalizing infeasibility in the process of solving a smooth constrained problem. A useful model is then $\psi(s; x_k) := g(c(x_k)+ \nabla c(x_k)^T s)$. If $L > 0$ is the Lipschitz constant of g and $M > 0$ that of the Jacobian of c, we have

$$|h(x_k + s) - \psi(s; x_k)| \leq L\|c(x_k + s) - c(x_k) - \nabla c(x_k)^T s\| \leq \tfrac{1}{2} LM\|s\|^2,$$

for all s, and (5.9a) is satisfied.

In order to develop a convergence analysis, we further assume that the gradient of $\varphi(\cdot; x_k)$ is Lipschitz continuous, which is satisfied, for instance, in the case of a quadratic model. It is not necessary to assume at this point that those Lipschitz constants are uniformly bounded; we will make such an assumption when needed. We gather the assumptions on the model from sections 5.2.1 and 5.2.2 in Model Assumption 5.2.2.

Model Assumption 5.2.2. *For any* $x \in \mathbb{R}^n$, $\varphi(\cdot; x)$ *is continuously differentiable with* $\varphi(0; x) = f(x)$ *and* $\nabla_s \varphi(0; x) = \nabla f(x)$. *In addition,* $\nabla_s \varphi(\cdot; x)$ *is Lipschitz continuous with constant* $L(x)$ *for all* $x \in \mathbb{R}^n$. *Finally,* $\psi(\cdot; x)$ *is proper, lsc, and satisfies* $\psi(0; x) = h(x)$ *and* $\partial\psi(0; x) = \partial h(x)$.

The complete process is formalized in Algorithm 16, which differs from a traditional trust-region algorithm in a few respects. First, each iteration begins with the choice of a

steplength $\nu_k > 0$ for the proximal-gradient method. Steplength ν_k must be below $1/L(x_k)$ to ensure descent; in addition, we connect ν_k explicitly to Δ_k for a reason that becomes apparent in Theorem 5.2.4. Second, a step computation occurs in two phases. In the first phase, we compute the first step $s_{k,1}$ of the proximal-gradient method applied to our model with trust-region radius Δ_k. Step $s_{k,1}$ is an analog of the scaled projected gradient for nonsmooth regularized problems. In the second phase, we continue the proximal-gradient iterations from $s_{k,1}$ but possibly modify the trust-region radius so it does not exceed $\beta\|s_{k,1}\|$ for a prescribed $\beta \geq 1$. This choice is similar in spirit to the analysis of Curtis et al. [42] for smooth problems, who set the radius to be proportional to the gradient norm. More precisely, if $\|s_{k,1}\| < \Delta_k$, we explore a trust region of radius $\beta\|s_{k,1}\| \geq \|s_{k,1}\|$. Because the constraint $\|s\| \leq \Delta_k$ is inactive at $s_{k,1}$, the first step of the proximal gradient method computed in the updated trust region remains $s_{k,1}$, so that subsequent proximal gradient iterations will result in further decrease and the ultimate step s_k will satisfy (5.9b). If, on the other hand, $\|s_{k,1}\| = \Delta_k$, the first step of the proximal gradient method computed in a larger trust region might differ from $s_{k,1}$, which would jeopardize satisfaction of (5.9b). In order to preserve (5.9b), we leave Δ_k unchanged.

A final difference with traditional trust-region methods is that h and/or $\psi(\cdot; x)$ can take the value $+\infty$. In accordance with [120], we employ extended arithmetic rules in which $+\infty \cdot 0 = 0 \cdot (+\infty) = +\infty/(+\infty) := 0$. Thus if $h(x_k + s_k) = \psi(s_k; x_k) = +\infty$, we set $\rho_k := 0$ at step 9 of Algorithm 16.

5.2.4 Convergence analysis and iteration complexity

Our first result states that a successful step is guaranteed provided the trust-region radius is small enough.

Theorem 5.2.4. *Let Model Assumption 5.2.2 and Step Assumption 5.2.1 be satisfied and let*

$$\Delta_{\text{succ}} := \frac{\kappa_{\text{mdc}}(1-\eta_2)}{2\kappa_{\text{m}}\alpha\beta^2} > 0. \tag{5.10}$$

Algorithm 16 Nonsmooth Regularized Trust-Region Algorithm.

Choose constants

$$0 < \eta_1 \leq \eta_2 < 1, \quad 0 < \gamma_1 \leq \gamma_2 < 1 < \gamma_3 \leq \gamma_4 \quad \text{and} \quad \alpha > 0,\ \beta \geq 1.$$

Choose $x_0 \in \mathbb{R}^n$ where h is finite, $\Delta_0 > 0$, compute $f(x_0) + h(x_0)$.

for $k = 0, 1, \ldots$ **do**

  Choose $0 < \nu_k \leq 1/(L(x_k) + \alpha^{-1}\Delta_k^{-1})$.

  Define $m_k(s; x_k) := \varphi(s; x_k) + \psi(s; x_k)$ satisfying Model Assumption 5.2.2.

  Define $m_k^\nu(s; x_k) := \varphi^\nu(s; x_k) + \psi(s; x_k)$ where $\varphi^\nu(\cdot; x_k)$ is as in (5.3).

  Compute $s_{k,1}$ as the solution of (5.8) with model $m_k^\nu(s; x_k)$.

  Compute an approximate solution s_k of (5.8) with model $m_k(s; x_k)$ satisfying Step Assumption 5.2.1 and such that $\|s_k\| \leq \min(\Delta_k, \beta\|s_{k,1}\|)$.

  Compute the ratio

$$\rho_k := \frac{f(x_k) + h(x_k) - (f(x_k + s_k) + h(x_k + s_k))}{m_k(0; x_k) - m_k(s_k; x_k)}.$$

  If $\rho_k \geq \eta_1$, set $x_{k+1} = x_k + s_k$. Otherwise, set $x_{k+1} = x_k$.

  Update the trust-region radius according to

$$\Delta_{k+1} \in \begin{cases} [\gamma_3\Delta_k,\ \gamma_4\Delta_k] & \text{if } \rho_k \geq \eta_2, \quad \text{(very successful iteration)} \\ [\gamma_2\Delta_k,\ \Delta_k] & \text{if } \eta_1 \leq \rho_k < \eta_2, \quad \text{(successful iteration)} \\ [\gamma_1\Delta_k,\ \gamma_2\Delta_k] & \text{if } \rho_k < \eta_1 \quad \text{(unsuccessful iteration).} \end{cases}$$

If x_k is not first-order stationary and $\Delta_k \leq \Delta_{\text{succ}}$, then iteration k is very successful and $\Delta_{k+1} \geq \Delta_k$.

Proof. Because x_k is not first-order stationary, $s_{k,1} \neq 0$ and $s_k \neq 0$. Note first that (5.6), (5.7) and Model Assumption 5.2.2 give

$$\xi(\Delta_k; x_k, \nu_k) \geq (f+h)(x_k) - (\varphi+\psi)(s_1; x_k) \geq \tfrac{1}{2}(\nu_k^{-1} - L(x_k))\|s_{k,1}\|^2.$$

Line 4 of Algorithm 16 implies in turn that $\nu_k^{-1} - L(x_k) \geq \alpha^{-1}\Delta_k^{-1}$, so that

$$\xi(\Delta_k; x_k, \nu_k) \geq \tfrac{1}{2}\alpha^{-1}\Delta_k^{-1}\|s_{k,1}\|^2.$$

Model Assumption 5.2.2 and Step Assumption 5.2.1 together with the bound $\|s_k\| \leq \beta\|s_{k,1}\|$ yield

$$\begin{aligned}
|\rho_k - 1| &= \left| \frac{f(x_k+s_k) + h(x_k+s_k) - m_k(s_k; x_k)}{m_k(0; x_k) - m_k(s_k; x_k)} \right| \\
&\leq \frac{\kappa_{\text{m}}\|s_k\|^2}{\kappa_{\text{mdc}}\xi(\Delta_k; x_k, \nu_k)} \\
&\leq \frac{\kappa_{\text{m}}\beta^2\|s_{k,1}\|^2}{\frac{1}{2}\alpha^{-1}\Delta_k^{-1}\|s_{k,1}\|^2} \\
&= \frac{2\kappa_{\text{m}}\alpha\beta^2}{\kappa_{\text{mdc}}}\Delta_k.
\end{aligned}$$

Therefore, $\Delta_k \leq \Delta_{\text{succ}}$ implies $\rho_k \geq \eta_2$ and iteration k is very successful. The trust-region update of Algorithm 16 ensures that $\Delta_{k+1} \geq \Delta_k$. □

A careful examination of the proof of Theorem 5.2.4 reveals that the model adequacy condition (5.9a) could be replaced with the weaker condition

$$|f(x_k+s_k) + h(x_k+s_k) - m_k(s_k; x_k)| \leq \kappa_{\text{m}}\beta^2\|s_{k,1}\|^2, \tag{5.11}$$

which encapsulates the step size and the trust-region radius simultaneously, and suggests that $s_{k,1}$ is the appropriate generalization of the projected gradient for nonsmooth regularized optimization.

We are now in position to show that Algorithm 16 identifies a first-order critical point. We first consider the case where there are finitely many successful iterations.

Theorem 5.2.5. *Let Model Assumption 5.2.2 and Step Assumption 5.2.1 be satisfied. If Algorithm 16 only generates finitely many successful iterations, then $x_k = x^*$ for all sufficiently large k and x^* is first-order critical.*

Proof. The proof mirrors that of Conn et al. [40, Theorem 6.4.4]. Under the assumptions given, there exists $k_0 \in \mathbb{N}$ such that all iterations $k \geq k_0$ are unsuccessful and $x_k = x_{k_0} = x^*$. Assume by contradiction that x^* is not first-order critical. The mechanism of Algorithm 16 ensures that Δ_k decreases on unsuccessful iterations. Thus, there must be $k_1 \geq k_0$ such that $\Delta_k \leq \Delta_{\text{succ}}$, where Δ_{succ} is defined in Theorem 5.2.4, which ensures that iteration k_1 is successful and contradicts our assumption. □

We now turn to the case where there are infinitely many successful iterations and show that the objective is either unbounded below or a measure of criticality converges to zero. The mechanism of Algorithm 16 and Theorem 5.2.4 together ensure that

$$\Delta_k \geq \Delta_{\min} \quad \text{for all } k \in \mathbb{N} \text{ where } \quad \Delta_{\min} := \min(\Delta_0, \gamma_1 \Delta_{\text{succ}}) > 0. \tag{5.12}$$

Thus, by definition of $\xi(\cdot; x_k, \nu_k)$ and (5.12), we have

$$\xi(\Delta_k; x_k, \nu_k) \geq \xi(\Delta_{\min}; x_k, \nu_k) \quad \text{for all } k \in \mathbb{N}. \tag{5.13}$$

Following this last observation and in view of Proposition 5.2.3 and (2.8), we define $\nu_k^{-1}\xi(\Delta_{\min}; x_k, \nu_k)^{\frac{1}{2}}$ as our measure of criticality. Observe the similarity between this measure and $\|G_{\nu_k}(0)\|$ defined in (2.7).

Our objective is to establish that $\liminf \nu_k^{-1}\xi(\Delta_{\min}; x_k, \nu_k) = 0$ provided $f + h$ is bounded below. While doing so, we also establish a complexity result.

Let $\epsilon > 0$ be a stopping tolerance set by the user. We are interested in determining the smallest iteration number $k(\epsilon)$ at which we achieve the first-order optimality condition

$$\nu_k^{-1}\xi(\Delta_{\min}; x_k, \nu_k)^{\frac{1}{2}} \leq \epsilon \quad (0 < \epsilon < 1). \tag{5.14}$$

We denote

$$\mathcal{S} := \{k \in \mathbb{N} \mid \rho_k \geq \eta_1\}, \tag{5.15a}$$

$$\mathcal{S}(\epsilon) := \{k \in \mathcal{S} \mid k < k(\epsilon)\}, \tag{5.15b}$$

$$\mathcal{U}(\epsilon) := \{k \in \mathbb{N} \mid k \notin \mathcal{S} \text{ and } k < k(\epsilon)\}, \tag{5.15c}$$

respectively the set of all successful iterations, the set of successful iterations for which (5.14) has not yet been attained, and the set of unsuccessful iterations before (5.14) is first attained.

We make the following additional assumption on the model.

Model Assumption 5.2.3. *In Model Assumption 5.2.2, there exists $L > 0$ such that $0 < L(x_k) \leq L$ for all $k \in \mathbb{N}$. In addition, we select ν_k at line 4 of Algorithm 16 in a way that there exists $\nu_{\min} > 0$ such that $\nu_k \geq \nu_{\min}$ for all $k \in \mathbb{N}$.*

We stress that it is not necessary to know the value of or estimate L; only to ensure that such a constant exists, which may be achieved either by controling the norm of quasi-Newton approximations [92] or employing exact Hessians and substituting one for a bounded approximation when its norm is too large. Finally, in view of (5.12), there exists $\nu_{\min} > 0$ satisfying the assumption. For instance, choosing $\nu_k := 1/(L(x_k)+\alpha^{-1}\Delta_k^{-1})$ at each iteration ensures that $\nu_k \geq \nu_{\min} := 1/(L + \alpha^{-1}\Delta_{\min}^{-1}) > 0$.

The following two results parallel the now-classic complexity analysis of Cartis et al. [36] and references therein.

Lemma 5.2.6. *Let Model Assumptions 5.2.2 and 5.2.3 and Step Assumption 5.2.1 be satisfied. Assume there are infinitely many successful iterations and that $f(x_k) + h(x_k) \geq (f + h)_{\text{low}}$ for all $k \in \mathbb{N}$. Then, for all $\epsilon \in (0, 1)$,*

$$|\mathcal{S}(\epsilon)| \leq \frac{(f + h)(x_0) - (f + h)_{\text{low}}}{\eta_1 \kappa_{\text{mdc}} \nu_{\min}^2 \epsilon^2} = O(\epsilon^{-2}). \tag{5.16}$$

Proof. If $k \in \mathcal{S}(\epsilon)$, Model Assumption 5.2.3 and Step Assumption 5.2.1 and (5.13) imply

$$\begin{aligned} f(x_k) + h(x_k) - f(x_k + s_k) - h(x_k + s_k) &\geq \eta_1(m_k(0; x_k) - m_k(s_k; x_k)) \\ &\geq \eta_1 \kappa_{\text{mdc}} \xi(\Delta_k; x_k, \nu_k) \\ &\geq \eta_1 \kappa_{\text{mdc}} \xi(\Delta_{\min}; x_k, \nu_k) \\ &\geq \eta_1 \kappa_{\text{mdc}} \nu_k^2 \epsilon^2 \\ &\geq \eta_1 \kappa_{\text{mdc}} \nu_{\min}^2 \epsilon^2. \end{aligned}$$

Because $f + h$ is bounded below by $(f + h)_{\text{low}}$, summing the above inequalities over all $k \in \mathcal{S}(\epsilon)$ yields

$$(f + h)(x_0) - (f + h)_{\text{low}} \geq \sum_{k \in \mathcal{S}(\epsilon)} (f + h)(x_k) - (f + h)(x_{k+1}) \geq |\mathcal{S}(\epsilon)| \eta_1 \kappa_{\text{mdc}} \nu_{\min}^2 \epsilon^2,$$

which establishes (5.16). □

In order to derive a similar bound on the total number of iterations before (5.14) is first attained, we need to bound the number of unsuccessful iterations.

Lemma 5.2.7. *Under the assumptions of Lemma 5.2.6,*

$$|\mathcal{U}(\epsilon)| \leq \log_{\gamma_2}(\Delta_{\min}/\Delta_0) + |\mathcal{S}(\epsilon)| |\log_{\gamma_2}(\gamma_4)| = O(\epsilon^{-2}). \tag{5.17}$$

Proof. Each unsuccessful iteration reduces the trust-region radius by a factor at least γ_2, while at each successful iteration, $\Delta_{k+1} \leq \gamma_4 \Delta_k$. Thus if $k(\epsilon) - 1$ is the iteration index just before (5.14) occurs for the first time,

$$\Delta_{\min} \leq \Delta_{k(\epsilon)-1} \leq \Delta_0 \gamma_2^{|\mathcal{U}(\epsilon)|} \gamma_4^{|\mathcal{S}(\epsilon)|}.$$

Taking logarithms on both sides and remembering that $0 < \gamma_2 < 1$ gives

$$|\mathcal{U}(\epsilon)| \log(\gamma_2) + |\mathcal{S}(\epsilon)| \log(\gamma_4) \geq \log(\Delta_{\min}/\Delta_0),$$

and establishes (5.17). □

Finally, the total number of iteration until (5.14) is attained is given in the next result, which simply combines Lemma 5.2.6 and Lemma 5.2.7.

Theorem 5.2.8. *Under the assumptions of Lemma 5.2.6,*

$$|\mathcal{S}(\epsilon)| + |\mathcal{U}(\epsilon)| = O(\epsilon^{-2}). \tag{5.18}$$

We use the update $\Delta_{k+1} \in [\gamma_3 \Delta_k, \gamma_4 \Delta_k]$ on very successful iterations but other possibilities exist. For instance, it is common to set $\Delta_{k+1} = \max(\gamma_3 \|s_k\|, \Delta_k)$ instead. Lemma 5.2.7 continues to hold because on successful iterations, $\Delta_{k+1} \leq \max(\gamma_3 \Delta_k, \Delta_k) = \gamma_3 \Delta_k$.

Curtis et al. [42] establish a complexity bound of $O(\epsilon^{-2})$ by making Δ_k proportional to an optimality measure—in their context of smooth optimization, they choose the gradient norm. Grapiglia et al. [59] study the convergence and complexity of a generic algorithm that has trust-region methods as a special case and obtain the $O(\epsilon^{-2})$ complexity bound under stronger smoothness assumptions than ours. Among others, they establish a bound for regularized optimization but also require h to be convex and globally Lipschitz continuous. Curtis et al. [41] describe a nonstandard trust-region algorithm with a stronger $O(\epsilon^{-3/2})$ complexity bound.

A straightforward consequence of Theorem 5.2.8 is that if $f + h$ is bounded below, a subsequence of the criticality measure converges to zero.

Corollary 5.2.9. *Let Model Assumptions 5.2.2 and 5.2.3, and Step Assumption 5.2.1 be satisfied. If there are infinitely many successful iterations, then, either*

$$\lim_{k \to \infty} f(x_k) + h(x_k) \to -\infty \quad \textit{or} \quad \liminf_{k \to \infty} \nu_k^{-1} \xi(\Delta_{\min}; x_k, \nu_k)^{\frac{1}{2}} = 0.$$

Proof. Follows directly from Theorem 5.2.8. □

In order to give an interpretation of Corollary 5.2.9, consider (5.2) with $\Delta = \Delta_{\min} > 0$ along with its value function $p(\Delta_{\min}; x, \nu)$, optimal set $P(\Delta_{\min}; x, \nu)$ and the optimality measure $\xi(\Delta_{\min}; x, \nu)$, where (x, ν) now plays the role of the parameter. Similar to Proposition 5.2.1, though with slightly stronger assumptions than Model Assumption 5.2.1, we have the following result.

Proposition 5.2.10. *Let Problem Assumption 5.2.1 be satisfied and consider (5.2) with φ as in (5.3). Assume ψ is proper and lsc in the joint variables (s, x) and $\psi(s; x) + \chi(s; \Delta_{\min})$ is level-bounded in s locally uniformly in x. Then, the domain of $p(\Delta_{\min}; \cdot, \cdot)$ and $P(\Delta_{\min}; \cdot, \cdot)$ is $\mathbb{R}^n \times \{\nu \mid \nu > 0\}$. In addition,*

1. *$p(\Delta_{\min}; \cdot, \cdot)$ is proper continuous and for all $x \in \mathbb{R}^n$ and $\nu > 0$, $P(\Delta_{\min}; x, \nu)$ is nonempty and compact. In addition, $\xi(\Delta_{\min}; \cdot, \cdot)$ is proper lsc;*

2. *if $\{x_k\} \to \bar{x}$ and $\{\nu_k\} \to \bar{\nu} > 0$, and for each k, $s_k \in P(\Delta_{\min}; x_k, \nu_k)$, then $\{s_k\}$ is bounded and all its limit points are in $P(\Delta_{\min}; \bar{x}, \bar{\nu})$.*

Proof. Because h is proper lsc, (5.7) implies that $\xi(\Delta_{\min}; \cdot, \cdot)$ is proper whenever $p(\Delta_{\min}; \cdot, \cdot)$ is proper and is lsc whenever $p(\Delta_{\min}; \cdot, \cdot)$ is continuous. The latter holds because $p(\Delta_{\min}; \cdot, \cdot)$ is the composition of ∇f, which is continuous, with the Moreau envelope of $\psi(\cdot; x) + \chi(\cdot; \Delta)$, and such Moreau envelope is continuous in (x, ν)—see, [120, Theorem 1.25]. The rest follows by [120, Theorems 1.17 and 7.41]. □

By Corollary 5.2.9, if $f + h$ is bounded below, there is an index set $\mathcal{K}$ such that $\{\nu_k^{-1} \xi(\Delta_{\min}; x_k, \nu_k)^{\frac{1}{2}}\}_{\mathcal{K}} \to 0$. Assume that $\{(x_k, \nu_k)\}_{\mathcal{K}}$ possesses a limit point and, without loss of generality, that $\{(x_k, \nu_k)\}_{\mathcal{K}} \to (\bar{x}, \bar{\nu})$ with $\bar{\nu} > 0$. That implies that $\{\xi(\Delta_{\min}; x_k, \nu_k)\}_{\mathcal{K}} \to 0$ because for all sufficiently large k,

$$\nu_k^{-1} \xi(\Delta_{\min}; x_k, \nu_k)^{\frac{1}{2}} \geq \tfrac{1}{2} \bar{\nu}^{-1} \xi(\Delta_{\min}; x_k, \nu_k)^{\frac{1}{2}} \geq 0.$$

Under the assumptions of Proposition 5.2.10, $\xi(\Delta_{\min}; \cdot, \cdot)$ is lsc, which means exactly that

$$0 = \liminf_{k \in \mathcal{K}} \xi(\Delta_{\min}; x_k, \nu_k) = \xi(\Delta_{\min}; \bar{x}, \bar{\nu}),$$

so that $\bar{x}$ is first-order critical.

It turns out that a stronger conclusion holds without further assumptions; the following result implies that *every* limit point of $\{(x_k, \nu_k)\}$ determines a first-order critical point. The proof follows the logic of [40, Theorem 6.4.6] but is significantly simpler due to the form of Step Assumption 5.2.1 and (5.13).

Theorem 5.2.11. *Let Model Assumptions 5.2.2 and 5.2.3 and Step Assumption 5.2.1 be satisfied. If there are infinitely many successful iterations,*

$$\lim_{k\to\infty} f(x_k) + h(x_k) \to -\infty \quad \text{or} \quad \lim_{k\to\infty} \nu_k^{-1}\xi(\Delta_{\min}; x_k, \nu_k)^{\frac{1}{2}} = 0.$$

Proof. If $\{\nu_k^{-1}\xi(\Delta_{\min}; x_k, \nu_k)^{\frac{1}{2}}\} \not\to 0$, there exist $\epsilon > 0$ and an infinite set $\mathcal{K} \subset \mathcal{S}$ such that $\nu_k^{-1}\xi(\Delta_{\min}; x_k, \nu_k)^{\frac{1}{2}} \geq \epsilon$ for all $k \in \mathcal{K}$. Because each $k \in \mathcal{K}$ is a successful iteration, Step Assumption 5.2.1 and (5.13) yield

$$\begin{aligned}(f+h)(x_k) - (f+h)(x_{k+1}) &\geq \eta_1\kappa_{\text{mdc}}\xi(\Delta_k; x_k, \nu_k) \\ &\geq \eta_1\kappa_{\text{mdc}}\xi(\Delta_{\min}; x_k, \nu_k) \\ &\geq \eta_1\kappa_{\text{mdc}}\nu_{\min}^2\epsilon^2\end{aligned}$$

for all $k \in \mathcal{K}$, which is a contradiction if $\{f(x_k) + h(x_k)\}$ is not bounded below. □

5.3 *Proximal-quasi-Newton trust-region method*

In this section, we consider the computation of a trust-region step and develop a special case of Proposition 2.3.1 in which

$$\varphi(s;x) := f(x) + \nabla f(x)^T s + \tfrac{1}{2} s^T B s, \tag{5.19}$$

where $B = B^T$. We assume that $\Delta > 0$ is fixed. For conciseness, we use the notation $\varphi(s) := \varphi(s;x)$ and $\psi(s) := \psi(s;x) + \chi(s;\Delta)$. We work under Model Assumption 5.2.2, i.e., we assume that ψ is proper and lsc with prox-boundedness coming from $\chi(\cdot;\Delta)$.

5.3.1 Computing a trust-region step

The following result states a fundamental relationship between G_ν and $\partial\psi$.

Lemma 5.3.1.

Let s_{j+1} be given by (2.5) and $G_\nu(s_j)$ be defined by (2.7). Then,

$$G_\nu(s_j) - \nabla\varphi(s_j) \in \partial\psi(s_{j+1}). \tag{5.20a}$$

$$(B - \nu^{-1} I)(s_{j+1} - s_j) \in \nabla\varphi(s_{j+1}) + \partial\psi(s_{j+1}). \tag{5.20b}$$

Proof. (5.20a) is a simple restatement of (2.6) and (5.20b) results from adding $\nabla\varphi(s_{j+1})$to both sides of (2.6) and substituting the gradient of φ using (5.19). □

The next result shows that (2.5) is a descent method when φ is a quadratic.

Lemma 5.3.2. *Let $\{s_j\}$ be generated according to (2.5). For all $j \geq 0$,*

$$\psi(s_{j+1}) + \nabla\varphi(s_j)^T(s_{j+1} - s_j) \leq \psi(s_j) - \tfrac{1}{2}\nu^{-1}\|s_{j+1} - s_j\|^2, \tag{5.21a}$$

$$(\varphi + \psi)(s_{j+1}) \leq (\varphi + \psi)(s_j) + \tfrac{1}{2}(s_{j+1} - s_j)^T(B - \nu^{-1} I)(s_{j+1} - s_j). \tag{5.21b}$$

Proof. Because s_{j+1} solves (2.5),

$$\tfrac{1}{2}\nu^{-1}\|s_{j+1} - (s_j - \nu\nabla\varphi(s_j))\|^2 + \psi(s_{j+1}) \leq \tfrac{1}{2}\nu^{-1}\|\nu\nabla\varphi(s_j)\|^2 + \psi(s_j).$$

By expanding the squared norm in the left-hand-side of the above and cancelling the common term $\|\nu\nabla\varphi(s_j)\|^2$, we obtain (5.21a). Because φ is quadratic,

$$\varphi(s_{j+1}) = \varphi(s_j) + \nabla\varphi(s_j)^T(s_{j+1} - s_j) + \tfrac{1}{2}(s_{j+1} - s_j)^T B(s_{j+1} - s_j).$$

We now add $\psi(s_{j+1})$ to both sides and use (5.21a) and obtain

$$\begin{aligned}(\varphi + \psi)(s_{j+1}) &\leq \varphi(s_j) + \psi(s_j) - \tfrac{1}{2}\nu^{-1}\|s_{j+1} - s_j\|^2 + \tfrac{1}{2}(s_{j+1} - s_j)^T B(s_{j+1} - s_j) \\ &= (\varphi + \psi)(s_j) + \tfrac{1}{2}(s_{j+1} - s_j)^T(B - \nu^{-1}I)(s_{j+1} - s_j). \qquad \square\end{aligned}$$

$\square$

We now examine two choices of $\nu > 0$ that result in two decrease behaviors.

Corollary 5.3.3. *Under the assumptions of Lemma 5.3.2, assume $0 < \nu \leq (1-\theta)/\|B\|$ for some $\theta \in (0, 1)$, or simply that $\nu > 0$ if $B = 0$, in which case $\theta = 1$. Then,*

$$(\varphi + \psi)(s_{j+1}) \leq (\varphi + \psi)(s_j) - \tfrac{1}{2}\theta\nu^{-1}\|s_j - s_{j+1}\|^2, \quad (j \geq 0). \tag{5.22}$$

Proof. If $B = 0$, (5.22) with $\theta = 1$ follows directly from (5.21a). If $B \neq 0$, we have by assumption $(1-\theta)\nu^{-1} \geq \|B\|$, so that $\lambda_{\max}(B - \nu^{-1}I) \leq -\theta\nu^{-1} < 0$, and therefore,

$$(s_{j+1} - s_j)^T(B - \nu^{-1}I)(s_{j+1} - s_j) \leq -\theta\nu^{-1}\|s_{j+1} - s_j\|^2,$$

which combines with (5.21b) to complete the proof. $\square$

Corollary 5.3.4. *Under the assumptions of Lemma 5.3.2, assume $B \neq 0$, let $0 < \theta < 1/(4\|B\|)$ and $\nu^{\min} \leq \nu \leq \nu^{\max}$, where*

$$\nu^{\min} := \frac{1 - \sqrt{1 - 4\theta\|B\|}}{2\|B\|}, \qquad \nu^{\max} := \frac{1 + \sqrt{1 - 4\theta\|B\|}}{2\|B\|}.$$

Then, for all $j \geq 0$,

$$(\varphi + \psi)(s_{j+1}) \leq (\varphi + \psi)(s_j) - \tfrac{1}{2}\theta\nu^{-2}\|s_j - s_{j+1}\|^2 = (\varphi + \psi)(s_j) - \tfrac{1}{2}\theta\|G_\nu(s_j)\|^2. \tag{5.23}$$

Proof. Under our assumptions, the quadratic $p(\nu) := \|B\|\nu^2 - \nu + \theta$ has the two positive real roots $\nu^{\min}$ and $\nu^{\max}$. Moreover, for all $\nu \in [\nu^{\min}, \nu^{\max}]$, $p(\nu) \leq 0$, which can also be written $\|B\| - \nu^{-1} \leq -\theta\nu^{-2}$. Therefore, if $\nu \in [\nu^{\min}, \nu^{\max}]$, then for all j,

$$(s_{j+1} - s_j)^T (B - \nu^{-1} I)(s_{j+1} - s_j) \leq -\theta\nu^{-2} \|s_{j+1} - s_j\|^2 = -\theta \|G_\nu(s_j)\|^2,$$

which combines with (5.21b) to complete the proof. □

Because $s_0 = 0$ and $(\varphi + \psi)(s_0) = f(x) + h(x) < +\infty$, if ν is chosen as in Corollary 5.3.3 or Corollary 5.3.4, (2.5) generates iterates $\{s_j\}$ such that $\{(\varphi + \psi)(s_j)\}$ is monotonically decreasing and all its terms are finite. Finiteness implies that $\|s_j\| \leq \Delta$ for all $j \geq 0$, i.e., all iterates lie in the trust region. In particular, for any $j \geq 1$,

$$m_k(s_{j+1}; x_k) \leq m_k(s_j; x_k) \leq m_k(s_1; x_k) = m_k^\nu(s_1; x_k), \tag{5.24}$$

where $m_k^\nu(s_1; x_k) = \frac{1}{2}\nu^{-1}\|s_1 + \nu\nabla f(x_k)\|_2^2 + (\psi + \chi)(s_1)$ and hence s_j satisfies the sufficient decrease condition (5.9b), and the final equality results from the fact that s_1 is the same for any model of the form (5.19).

With regards to proximal gradient convergence, two situations may occur. In the first, (2.5) results in $s_{j_0+1} = s_{j_0}$ for a smallest index $j_0 > 0$. In that case, (2.6) yields

$$0 \in \partial(\varphi + \psi)(s_{j_0}),$$

i.e., we have identified a stationary point of (5.8) in a finite number of iterations, while decreasing the value of m_k at each iteration. Otherwise, $s_{j+1} \neq s_j$ for all $j \geq 0$, and the next result establishes sub-linear convergence of the proximal gradient method (2.5).

Theorem 5.3.5. *Let $\{s_j\}$ be generated according to (2.5) with ν as in Corollary 5.3.3. Denote $(\varphi + \psi)_{\text{low}} := \inf(\varphi + \psi) > -\infty$. Let v_{j+1} denote the left-hand side of (5.20b). For any $N \geq 1$,*

$$\min_{j=0,\ldots,N-1} \|v_{j+1}\| \leq \sqrt{\frac{2}{N\theta}(\nu^{-1} - \lambda_{\min}(B))\,((\varphi + \psi)(s_0) - (\varphi + \psi)_{\text{low}})}.$$

Proof. We rearrange (5.22) and sum from iteration $j = 0$ to iteration $j = N - 1$:

$$\sum_{j=0}^{N-1} \|s_j - s_{j+1}\|^2 \leq \frac{2\nu}{\theta}((\varphi + \psi)(s_0) - (\varphi + \psi)(s_N)) \leq \frac{2\nu}{\theta}((\varphi + \psi)(s_0) - (\varphi + \psi)_{\text{low}}).$$

For any positive sequence $\{c_j\}$,

$$\min_{0\leq j\leq N-1} c_j = \sqrt{\min_{0\leq j\leq N-1} c_j^2} \leq \sqrt{\frac{1}{N}\sum_{j=0}^{N-1} c_j^2}.$$

Therefore,

$$\min_{0\leq j\leq N-1} \|s_j - s_{j+1}\| \leq \sqrt{\frac{2\nu}{N\theta}((\varphi + \psi)(s_0) - (\varphi + \psi)_{\text{low}})}.$$

Because $\|v_{j+1}\| \leq \|B - \nu^{-1}I\| \, \|s_j - s_{j+1}\| = (\nu^{-1} - \lambda_{\min}(B)) \, \|s_j - s_{j+1}\| \leq \nu^{-1}\|s_j - s_{j+1}\|$, we obtain the desired result. □

When solving (5.8), a reasonable stopping condition would be $\|v_{j+1}\| \leq \epsilon$ for a user-chosen tolerance $\epsilon > 0$. Theorem 5.3.5 indicates that such stopping condition is attained after $N(\epsilon)$ iterations, where

$$N(\epsilon) = \left\lceil \frac{2}{\epsilon^2\theta}(\nu^{-1} - \lambda_{\min}(B)) \, ((\varphi + \psi)(s_0) - (\varphi + \psi)_{\text{low}}) \right\rceil.$$

A result similar to Theorem 5.3.5 can be established under the step size rule of Corollary 5.3.4, with nearly identical proof.

Theorem 5.3.6. *Let $\{s_j\}$ be generated according to (2.5) with ν as in Corollary 5.3.4 with $0 < \theta < 1/(4\|B\|)$. Assume ψ, and therefore $\varphi+\psi$, is bounded below and denote $(\varphi + \psi)_{\text{low}} := \inf(\varphi + \psi) > -\infty$. For any $N \geq 1$,*

$$\min_{j=0,\ldots,N-1} \|G_\nu(s_{j+1})\| \leq \sqrt{\frac{2}{N\theta}((\varphi + \psi)(s_0) - (\varphi + \psi)_{\text{low}})}.$$

5.4 *Proximal Operators for Trust-Region Subproblems*

In this section, we develop techniques for computing (2.5) for use in Steps 7 and 8 of Algorithm 16. Many standard proximal operators for both convex and nonconvex prox-bounded functions ψ have been worked out [17, 38], and new examples for nonconvex problems continuously appear. Well-known examples include the firm-thresholding penalty [57], the SCAD penalty [53], MCP penalty [151], lower C^2 functions [64], any ℓ_p^p-seminorm for $0 < p < 1$ [153, Appendix A], and other exotic operators, see e.g. [152, Table 1]. We refer to such functions ψ as *prox-friendly*. However, Algorithm 16 requires evaluating proximal operators for modified functions that combine a shift and a summation with an indicator function. By Model Assumption 5.2.2, our model $\psi(s;x) \approx h(x+s)$ must coincide with h in value and subdifferential at $s = 0$. In particular, the choice $\psi(s;x) = h(x+s)$ seems natural when h itself is prox-friendly. Here we consider

$$\psi(s;x) := h(x+s) + \chi(s;\Delta\mathbb{B}_p), \tag{5.25}$$

where h is prox-friendly, x is a shift, and $p \in \{1, 2, \infty\}$. Below, we provide closed form solutions and/or efficient routines for (5.25) with focus on the following cases:

1. for an arbitrary separable prox-friendly h, we evaluate $\operatorname{prox}_{\nu\psi(\cdot;x)}$ by leveraging $\operatorname{prox}_{\nu h}$, but we restrict our attention to $p = \infty$. This allows us to consider (5.25) with $h(x) = \lambda\|x\|_1$ and $h(x) = \lambda\|x\|_0$;

2. we consider $h(x) = \lambda\|x\|_1$ in (5.25) for $p = 2$.

5.4.1 $p = \infty$, h separable

For the special case of $\mathbb{B}_\infty$, (2.1b) and (5.25) yield

$$\operatorname*{prox}_{\nu\psi}(q) := \arg\min_s \tfrac{1}{2}\nu^{-1}\|s-q\|^2 + h(x+s) + \chi(s;\Delta\mathbb{B}_\infty). \tag{5.26}$$

If h is separable, i.e., $h(x) = \sum_i h_i(x_i)$, (5.26) decouples in each coordinate:

$$\operatorname*{prox}_{\nu\psi}(q)_i = \arg\min_{s_i} \tfrac{1}{2}\nu^{-1}(s_i - q_i)^2 + h_i(x_i + s_i) + \chi(s_i;[-\Delta,\Delta]).$$

Using the change of variable $v_i = x_i + s_i$, we may rewrite

$$\operatorname*{prox}_{\nu\psi}(q)_i = \arg\min_{v_i}\{\tfrac{1}{2}\nu^{-1}(v_i - x_i - q_i)^2 + h_i(v_i) + \chi(v_i; [x_i - \Delta, x_i + \Delta])\} - x_i.$$

If h is convex, we may work backwards from the form of the solution. For any $p_i \in \operatorname{prox}_{\nu\psi}(q)_i$, either

1. $|p_i| < \Delta$, in which case $p_i \in \operatorname{prox}_{\nu h_i}(q + x)_i - x_i$;

2. otherwise, $|p_i| = \Delta$ by construction, and

$$\begin{aligned}\operatorname*{prox}_{\nu\psi}(q)_i &= \arg\min_{v_i = x_i \pm \Delta}(\tfrac{1}{2}\nu^{-1}(v_i - (x_i + q_i))^2 + h_i(v_i)) - x_i \\ &= \arg\min_{s_i = \pm\Delta} \tfrac{1}{2}\nu^{-1}(s_i - q_i)^2 + h_i(x_i + s_i) \subseteq \{-\Delta, \Delta\}.\end{aligned}$$

In such cases, the definition of convexity implies that set of bound-constrained solutions includes the projection of the unconstrained solutions into the bounds. Because the objective of (5.26) is strictly convex, equality holds:

$$\operatorname*{prox}_{\nu\psi}(q)_i = \{\operatorname*{proj}_{[x_i - \Delta, x_i + \Delta]}(\operatorname*{prox}_{\nu h_i}(q + x)_i)\} - x_i = \operatorname*{proj}_{[-\Delta,\Delta]}(\operatorname*{prox}_{\nu h_i}(q + x)_i - x_i),$$

For example, let $h(x) = \lambda\|x\|_1$. Then,

$$\operatorname*{prox}_{\nu\psi}(q)_i = \operatorname*{proj}_{[-\Delta,\Delta]}(\operatorname*{prox}_{\nu\lambda|\cdot|}(q + x)_i - x_i) = \operatorname*{proj}_{[-\Delta,\Delta]}\left(\begin{cases} q_i - \nu\lambda & x_i + q_i > \nu\lambda \\ -x_i & |x_i + q_i| \le \nu\lambda \\ q_i + \nu\lambda & x_i + q_i < -\nu\lambda \end{cases}\right)$$

$$= \operatorname*{proj}_{[-\Delta,\Delta]}\left(\operatorname*{proj}_{[q_i - \nu\lambda, q_i + \nu\lambda]}(-x_i)\right).$$

When h is nonconvex, there may be a greater variety of cases. For instance, if $h(x) = \lambda\|x\|_0$, a global solution of (5.26) may be one of the bounds, or either of the unconstrained local minimizers q and $-x$ if they lie inside the bounds. A simple strategy consists in evaluating the objective of (5.26) at those four points and choosing one with lowest objective value.

5.4.2 $p = 2$, $h(x) = \lambda\|x\|_1$

When using other norms to define the trust region, additional computations are required. For certain norms, we can dualize h to solve (5.26). We focus on $h(x) = \lambda\|x\|_1$ with an ℓ_2-norm trust-region throughout because the ℓ_2-norm is standard in the literature, and is used in section 5.6.1.

First, we rewrite the scaled ℓ_1-norm using its conjugate:

$$\lambda\|x+s\|_1 = \sup_{w\in\lambda\mathbb{B}_\infty} w^T(x+s),$$

recharacterizing (2.1b) and (5.25) as

$$\min_s \sup_{w\in\lambda\mathbb{B}_\infty} \tfrac{1}{2}\nu^{-1}\|s-q\|^2 + w^T(x+s) + \chi(s;\Delta\mathbb{B}_2). \tag{5.27}$$

Strong duality holds in this case since the objective is convex, piecewise linear-quadratic, and the primal solution is attained. We interchange the order of minimization and maximization and complete squares in s and in w to obtain

$$\sup_{w\in\lambda\mathbb{B}_\infty} \min_s \tfrac{1}{2}\nu^{-1}\|s-q+\nu w\|^2 + \chi(s;\Delta\mathbb{B}_2) - \tfrac{1}{2}\nu^{-1}\|x+q-\nu w\|^2 + \tfrac{1}{2}\nu^{-1}\|x+q\|^2. \tag{5.28}$$

The solution of the inner problem is

$$s(w) := \operatorname*{proj}_{\Delta\mathbb{B}_2}(q-\nu w). \tag{5.29}$$

We substitute (5.29) back into (5.28) to rewrite the dual objective as

$$\sup_{w\in\lambda\mathbb{B}_\infty} \tfrac{1}{2}\nu^{-1}\operatorname{dist}(q-\nu w;\Delta\mathbb{B}_2)^2 - \tfrac{1}{2}\nu^{-1}\|x+q-\nu w\|^2 + \tfrac{1}{2}\nu^{-1}\|x+q\|^2. \tag{5.30}$$

The change of variable

$$y = q - \nu w, \tag{5.31}$$

transforms (5.30) into

$$\min_{q-\nu\lambda\mathbf{1}\le y\le q+\nu\lambda\mathbf{1}} \tfrac{1}{2}\nu^{-1}\left(\|y+x\|^2 - \operatorname{dist}(y;\Delta\mathbb{B}_2)^2\right), \tag{5.32}$$

where $\mathbf{1}$ is a vector of all ones. As the value function of (5.27) with respect to s, the objective of (5.32) is convex [120, Proposition 2.22]. The first-order optimality conditions of (5.32) are

$$0 \in x + \frac{y}{\max\{1, \|y\|/\Delta\}} + \nu\partial\chi(y; [q - \nu\lambda\mathbf{1}, q + \nu\lambda\mathbf{1}]). \tag{5.33}$$

Once we have an optimal solution of (5.32) , denoted y^+, we can evaluate (5.29) at the corresponding w^+ to obtain

$$s = \operatorname*{proj}_{\Delta\mathbb{B}_2}(y^+).$$

which solves (5.26). To characterize y^+ more explicitly, we work backwards from properties of the solution. There are only two possibilities to consider: y^+ is in the trust region, and y^+ is outside of the trust region.

1. if $\|y^+\| < \Delta$, $\operatorname{dist}(y^+; \Delta\mathbb{B}_2) = 0$, and (5.32) and (5.33) simplify:

$$s = y^+ = \operatorname*{proj}_{[q-\nu\lambda\mathbf{1}, q+\nu\lambda\mathbf{1}]}(-x),$$

   where we used (5.29) and (5.31);

2. if $\|y^+\| \geq \Delta$, (5.33) becomes

$$0 \in x + \frac{\Delta}{\|y\|}y + \nu\partial\chi(y; [q - \nu\lambda\mathbf{1}, q + \nu\lambda\mathbf{1}]).$$

   Multiplying through by $\|y\|/\Delta$ yields

$$0 \in y + \frac{\|y\|}{\Delta}x + \frac{\nu\|y\|}{\Delta}\partial\chi(y; [q - \nu\lambda\mathbf{1}, q + \nu\lambda\mathbf{1}]). \tag{5.34}$$

   Suppose first that $\eta := \|y^+\|$ is known. A solution y^+ to (5.34) can be obtained by solving

$$\min_{y \in [q-\nu\lambda\mathbf{1}, q+\nu\lambda\mathbf{1}]} \tfrac{1}{2}\|y + \frac{\eta}{\Delta}x\|^2$$

   which can be written in closed form as

$$y = \operatorname*{proj}_{[q-\nu\lambda\mathbf{1}, q+\nu\lambda\mathbf{1}]}\left(-\frac{\eta}{\Delta}x\right). \tag{5.35}$$

Taking the norm of each side of (5.35) gives a scalar root finding equation that characterizes η:

$$\eta = \left\| \operatorname*{proj}_{[q-\nu\lambda\mathbf{1}, q+\nu\lambda\mathbf{1}]} \left(-\frac{\eta}{\Delta} x\right) \right\|.$$

Once we have solved for $\eta = \|y^+\|$, we obtain y^+ from (5.35), and, using (5.29),

$$s = \operatorname*{proj}_{\Delta\mathbb{B}_2} \left(\operatorname*{proj}_{[q-\nu\lambda\mathbf{1}, q+\nu\lambda\mathbf{1}]} \left(-\frac{\eta}{\Delta} x\right) \right) = \left(\operatorname*{proj}_{[q-\nu\lambda\mathbf{1}, q+\nu\lambda\mathbf{1}]} \left(-\frac{\eta}{\Delta} x\right) \right) \frac{\Delta}{\eta}.$$

5.5 *A quadratic regularization variant*

We now describe a variant of the trust-region algorithm of the previous sections inspired by the modified Gauss-Newton scheme proposed by Nesterov [104] in the context of nonlinear least-squares problems. Here again, Cartis et al. [36] establish a complexity of $O(\epsilon^{-2})$ iterations to attain a near-optimality condition under the assumption that h is convex and globally Lipschitz continuous. In the sequel, we obtain the same complexity bound under Problem Assumption 5.2.1. The quadratic regularization method decribed below is closely related to the standard proximal gradient method with the exception that it employs an adaptive steplength. It may be used as an alternative to a linesearch-based proximal gradient method such as those of Li and Lin [89] and Boţ et al. [30].

In the quadratic regularization method, we use the linear model

$$\varphi(s;x) = f(x) + \nabla f(x)^T s \approx f(x+s) \tag{5.36}$$

together with a model of $\psi(s;x)$ that satisfies Model Assumption 5.2.2. The first difference is that in the present setting, the Lipschitz constant of $\nabla\varphi(\cdot;x)$ is $L(x) = 0$ for all $x \in \mathbb{R}^n$. The second difference is that we must now assume that $\psi(\cdot;x)$ is prox-bounded. In particular, there exists $\lambda_x \in \mathbb{R} \cup \{+\infty\}$ such that $\psi(\cdot;x) + \frac{1}{2}\lambda_x^{-1}\|\cdot\|^2$ is bounded below [120, Exercise 1.24c]. We refer to the supremum of all such λ_x as the *threshold of boundedness* of $\psi(\cdot;x)$. If $\psi(\cdot;x)$ is itself bounded below, we may choose $\lambda_x = +\infty$. At x, we define

$$p(\sigma;x) := \underset{s}{\text{minimize}}\ \ m(s;x,\sigma), \tag{5.37a}$$

$$P(\sigma;x) := \arg\min_s\ m(s;x,\sigma), \tag{5.37b}$$

where

$$m(s;x,\sigma) := \varphi(s;x) + \psi(s;x) + \tfrac{1}{2}\sigma\|s\|^2, \tag{5.38}$$

and $\sigma > 0$ is a regularization parameter. From x, the method computes a step $s \in P(\sigma;x)$. As earlier, let us also define

$$\xi(\sigma;x) := f(x) + h(x) - p(\sigma;x) \geq 0. \tag{5.39}$$

If we combine (5.36) with (5.38), we may write

$$m(s;x,\sigma) = \tfrac{1}{2}\sigma\|s+\sigma^{-1}\nabla f(x)\|^2 + \psi(s;x) + f(x) - \tfrac{1}{2}\sigma^{-1}\|\nabla f(x)\|^2, \tag{5.40}$$

where the last two terms are independent of s. In (5.40), we recognize a model of the form (5.5), so that minimizing (5.38) amounts to performing a single step of the proximal gradient method with step size $1/\sigma$ and Lipschitz constant $L = 0$. The decrease guaranteed by the proximal gradient method is given by (5.6), i.e.,

$$\xi(x;\sigma) = f(x) + h(x) - m(s;x,\sigma) \geq \tfrac{1}{2}\sigma\|s\|^2, \tag{5.41}$$

so that

$$f(x) + h(x) - (\varphi(s;x) + \psi(s;x)) \geq \sigma\|s\|^2, \tag{5.42}$$

provided that $\sigma > 1/\lambda_x$. Because of (5.42), there is no need for a sufficient decrease assumption such as (5.9b) in the quadratic regularization method.

In view of (5.40), Proposition 2.2.1 applies to (5.37). In particular, $p(\sigma;x)$ is continuous in (σ,x), and $P(\sigma;x)$ is nonempty and compact for all $\sigma > 1/\lambda_x$.

By Proposition 2.2.3, for any $\sigma > 1/\lambda_x$, if $s \in P(\sigma;x)$, then $0 \in \nabla f(x) + \partial\psi(s;x) + \sigma s$. Thus, we have the following optimality result.

Lemma 5.5.1. *Let Model Assumption 5.2.2 be satisfied, $\psi(\cdot;x)$ be prox-bounded, and let $\sigma > 1/\lambda_x$. Then $\xi(\sigma;x) = 0 \iff 0 \in P(\sigma;x) \implies x$ is first-order stationary for (5.1).*

As in the trust-region context, we require that the difference between the model and the actual objective be bounded by a multiple of $\|s_k\|^2$:

Step Assumption 5.5.1. *There exists $\kappa_{\mathrm{m}} > 0$ such that for all k,*

$$|f(x_k+s_k) + h(x_k+s_k) - \varphi_k(s_k;x_k) - \psi(s_k;x_k)| \leq \kappa_{\mathrm{m}}\|s_k\|^2. \tag{5.43}$$

Once a step s has been computed, its quality is assessed by comparing the decrease in $\varphi(\cdot;x) + \psi(\cdot;x)$ with that in the objective $f + h$, similarly to Algorithm 16. If both are

in strong agreement, σ decreases. Otherwise, σ increases. We state the overall algorithm as Algorithm 17. Contrary to Algorithm 16, it is possible that, at certain iterations, $\sigma_k \leq 1/\lambda_{x_k}$, and in such a situation, it is possible that $m(s_k; x_k, \sigma_k) = -\infty$, and therefore that $\varphi(s_k; x_k) + \psi(s_k; x_k) \leq m(s_k; x_k, \sigma_k) = -\infty$. Because h is proper, $h(x_k + s_k)$ is either finite or $+\infty$. In such a case, thanks to extended arithmetic rules, $\rho_k = 0$, s_k is rejected, and σ_k is increased. After a finite number of such increases, we obtain $\sigma_k > 1/\lambda_{x_k}$.

Algorithm 17 Nonsmooth quadratic regularization algorithm.

1: Choose constants $0 < \eta_1 \leq \eta_2 < 1$ and $0 < \gamma_3 \leq 1 < \gamma_1 \leq \gamma_2$.

2: Choose $x_0 \in \mathbb{R}^n$ where h is finite, $\sigma_0 > 0$, compute $f(x_0) + h(x_0)$.

3: **for** $k = 0, 1, \ldots$ **do**

4: Define $m(s; x_k, \sigma_k)$ as in (5.38) satisfying Model Assumption 5.2.2 with $L = 0$.

5: Compute a solution s_k of (5.37) such that Step Assumption 5.5.1 holds.

6: Compute the ratio

$$\rho_k := \frac{f(x_k) + h(x_k) - (f(x_k + s_k) + h(x_k + s_k))}{\varphi(0; x_k) + \psi(0; x_k) - (\varphi(s_k; x_k) + \psi(s_k; x_k))}.$$

7: If $\rho_k \geq \eta_1$, set $x_{k+1} = x_k + s_k$. Otherwise, set $x_{k+1} = x_k$.

8: Update the regularization parameter according to

$$\sigma_{k+1} \in \begin{cases} [\gamma_3 \sigma_k,\ \sigma_k] & \text{if } \rho_k \geq \eta_2, \\ [\sigma_k,\ \gamma_1 \sigma_k] & \text{if } \eta_1 \leq \rho_k < \eta_2, \\ [\gamma_1 \sigma_k,\ \gamma_2 \sigma_k] & \text{if } \rho_k < \eta_1. \end{cases}$$

We now combine (5.42) with Step Assumption 5.5.1 into the following result.

Theorem 5.5.2. *Let Model Assumption 5.2.2 and Step Assumption 5.5.1 be satisfied, $\psi(\cdot; x_k)$ be prox-bounded for each $k \in \mathbb{N}$ with threshold of boundedness λ_{x_k}, and let*

$$\sigma_{\text{succ}} := \kappa_{\text{m}}/(1 - \eta_2) > 0. \tag{5.44}$$

If x_k is not first-order stationary and $\sigma_k \geq \max(1/\lambda_{x_k}, \sigma_{\text{succ}})$, then iteration k is very successful and $\sigma_{k+1} \leq \sigma_k$.

Proof. Let s_k be the step computed at iteration k of Algorithm 17. Because x_k is not first-order stationary, $s_k \neq 0$. Step Assumption 5.5.1 and (5.42) combine to yield

$$|\rho_k - 1| = \frac{|f(x_k + s_k) + h(x_k + s_k) - (\varphi(s_k; x_k) + \psi(s_k; x_k))|}{\varphi(0; x_k) + \psi(0; x_k) - (\varphi(s_k; x_k) + \psi(s_k; x_k))} \leq \frac{\kappa_{\text{m}} \|s_k\|^2}{\sigma_k \|s_k\|^2}.$$

After simplifying by $\|s_k\|^2$, we obtain $\sigma_k \geq \sigma_{\text{succ}} \implies \rho_k \geq \eta_2$. □

If there exists $\lambda > 0$ such that $\lambda_{x_k} \geq \lambda$ for all $k \in \mathbb{N}$, Theorem 5.5.2 ensures existence of a constant $\sigma_{\max} > 0$ such that

$$\sigma_k \leq \sigma_{\max} := \min(\sigma_0, \gamma_2 \max(1/\lambda, \sigma_{\text{succ}})) > 0 \quad \text{for all } k \in \mathbb{N}. \tag{5.45}$$

We paraphrase the above assumption by stating that $\psi(\cdot; x_k)$ is *uniformly prox-bounded*.

A result analogous to Theorem 5.2.5 holds for Algorithm 17. We omit the proof, as it is nearly identical.

Theorem 5.5.3. *Let Model Assumption 5.2.2 and Step Assumption 5.5.1 be satisfied, and $\psi(\cdot; x_k)$ be uniformly prox-bounded. If Algorithm 17 only generates finitely many successful iterations, $x_k = x^*$ for sufficiently large k and x^* is first-order critical.*

According to Proposition 2.2.1 part 2, and the identification $\nu = \sigma^{-1}$, $p(\sigma; x)$ increases as σ increases, so that $\xi(\sigma; x)$ decreases as σ increases, and (5.45) yields

$$\xi(\sigma_k; x_k) \geq \xi(\sigma_{\max}; x_k) \quad \text{for all } k \in \mathbb{N}. \tag{5.46}$$

Lemma 5.5.1, (5.42) and (5.46) suggest using $\xi(\sigma_{\max}; x_k)^{\frac{1}{2}}$ as stationarity measure.

Let $\epsilon > 0$ be a tolerance set by the user and consider the sets (5.15). We are now in position to establish complexity results analogous to those obtained for Algorithm 16. The proof is nearly identical and is omitted.

Theorem 5.5.4. *Let Model Assumption 5.2.2 and Step Assumption 5.5.1 be satisfied, and $\psi(\cdot; x_k)$ be uniformly prox-bounded. Assume there are infinitely many successful iterations and that $f(x_k) + h(x_k) \geq (f + h)_{\text{low}}$ for all $k \in \mathbb{N}$. Then, for all $\epsilon \in (0, 1)$,*

$$|\mathcal{S}(\epsilon)| = O(\epsilon^{-2}), \quad |\mathcal{U}(\epsilon)| = O(\epsilon^{-2}), \quad |\mathcal{S}(\epsilon)| + |\mathcal{U}(\epsilon)| = O(\epsilon^{-2}). \tag{5.47}$$

5.6 Implementation and numerical results

Algorithms 16 and 17 are implemented in Julia [22] and are available at `github.com/UW-AMO/TRNC`, along with scripts to reproduce our experiments. Our design allows the user to choose a method to compute a step, an important feature given the nonstandard $\psi_k + \chi_k$ operator.

We compare the performance of Algorithm 16 (TR) to other proximal quasi-Newton routines: PANOC [129] and ZeroFPR [132]. PANOC can be viewed as a proximal gradient descent scheme accelerated by limited-memory BFGS steps. It performs proximal gradient iterations with a backtracking linesearch, and then 20 quasi-Newton steps computed using the proximal gradient method. ZeroFPR is similar, but takes a fixed number of quasi-Newton steps between each proximal gradient step; it defaults to proximal gradient descent if no progress is made during the inner quasi-Newton steps. To compare, we count gradient evaluations as well as proximal operator evaluations, but in our example problems, proximal evaluations are far cheaper than gradients.

In the following experiments, we set $\psi(s; x_k) := h(x_k + s)$. Our stopping criteria for Algorithm 16 is $\xi(\Delta_k; x_k, \nu_k)^{1/2}$, which we use as a proxy for the first-order error measure $\nu_k^{-1}\xi(\Delta_{\min}; x_k, \nu_k)^{1/2}$ defined in (5.7). We set $\Delta_0 := 1.0$. We compute trust-region steps using the proximal-gradient (PG) method with step length chosen as in Corollary 5.3.3, denoted TR-PG in figures and tables. The user could choose accelerated variants for the subproblem, including our quadratic regularization procedure Algorithm 17 (R2), signified by TR-R2. In our experiments, the latter performed similarly to the proximal gradient method, although it typically required fewer inner iterations. We use proximal operators that include both $\psi(\cdot; x_k)$ and the indicator of the trust region as described in section 5.4. The criticality measure used in the inner PG iterations is the norm of the subgradient (5.20b), while that used in the R2 inner iterations is $\xi(\sigma_k; x_k)^{1/2}$, which is a proxy for $\xi(\sigma_{\max}; x_k)^{1/2}$. We set the inner tolerance to

$$\min(0.01,\, \xi(\Delta_k; x_k + s_{k,1}, \nu_k)^{\frac{1}{2}})\, \xi(\Delta_k; x_k + s_{k,1}, \nu_k),$$

which is inspired from inexact Newton methods to encourage fast local convergence. Note

that ξ is computed with the first step $s_{k,1}$ from Line 7 of Algorithm 16.

We use automatic differentiation as implemented in the ForwardDiff package [118] to obtain $\nabla f(x)$ and construct limited-memory quasi-Newton approximations by way of the LinearOperators package [109]. Below, we use LSR1 and LBFGS approximations with memory 5 for the BPDN and ODE examples, respectively.

5.6.1 LASSO/BPDN

The first set of experiments concerns LASSO/basis pursuit de-noise (BPDN) problems, which arise in statistical [133] and compressed sensing [50] applications. We seek to recover a sparse signal $x_{\text{true}} \in \mathbb{R}^n$ given observed noisy data $b \in \mathbb{R}^m$. x_{true} is a sparse vector containing mostly zeros and 10 values of ± 1 where both the index of the nonzero entry and $\pm$ are randomly generated.

We set $m = 200$, $n = 512$, $b := Ax_{\text{true}} + \varepsilon$ where $\varepsilon \sim \mathcal{N}(0, .01)$ and A to have orthonormal rows—A^T is generated by taking the Q factor in the thin QR decomposition of a random $n \times m$ matrix. To recover x, we solve

$$\underset{x}{\text{minimize}}\ \tfrac{1}{2}\|Ax - b\|_2^2 + h(x). \tag{5.48}$$

We first consider $h(x) = \lambda\|x\|_p$ for $p \in \{0, 1\}$ with $\lambda = 0.1\|A^T b\|_\infty$ in the vein of [142], and employ both the ℓ_2 and ℓ_∞ norms to define the trust region. We also consider $h(x) = \chi(x; \lambda\mathbb{B}_0)$ with $\lambda = 10$ and an ℓ_∞-norm trust region. We set the maximum number of inner iterations to 5000 and $\epsilon = 10^{-3}$. The quasi-Newton model is defined by a limited-memory SR1 approximation with memory 5. All algorithms use $x_0 = 0$.

Table 5.1 and Figure 5.1 summarize our results. Table 5.1 shows that Algorithm 16 performs comparably to PANOC and ZeroFPR in terms of parameter fit, it performs significantly fewer gradient evaluations and significantly more proximal operator evaluations. Thus there is an advantage when proximal evaluations are cheap relative to gradient evaluations, especially in situations where the proximal operator of ψ is simpler or cheaper than that of h. All algorithms yield nearly identical solution quality. The objective value history in

Figure 5.1 shows a steeper initial decrease for Algorithm 16 with shorter tails in all cases. Results with R2 as subproblem solver are nearly identical though R2 performed fewer inner iterations than PG.

Table 5.1: BPDN results (5.48) with Algorithm 16 and a proximal gradient subsolver (TR-PG), PANOC and ZeroFPR (ZFP). $\Delta\mathbb{B}_p$ indicates the norm used in the trust region. The true value of $h(\cdot)/\lambda$ is 10 for $\|\cdot\|_1$ and $\|\cdot\|_0$, but 0 for $\chi(\cdot;\lambda\mathbb{B}_0)$.

		$h = \lambda\|\cdot\|_1$, $\Delta\mathbb{B}_2$			$h = \lambda\|\cdot\|_0$, $\Delta\mathbb{B}_\infty$			$h = \chi(\cdot;\lambda\mathbb{B}_0)$, $\Delta\mathbb{B}_\infty$		
	True	TR-PG	PANOC	ZFP	TR-PG	PANOC	ZFP	TR-PG	PANOC	ZFP
$f(x)$	0.020	0.005	0.005	0.005	0.019	0.019	0.019	0.019	0.019	0.019
$h(x)/\lambda$	10/0	10.750	10.767	10.750	10	10	10	0	0	0
$\|x - x_{\text{true}}\|_2/\|A\|$	0	0.134	0.141	0.133	0.055	0.055	0.056	0.054	0.056	0.055
∇f evals		24	78	45	14	69	23	6	12	10
$\text{prox}_{\nu\psi}$ calls		270	52	95	90	36	57	32	6	14

5.6.2 A nonlinear inverse problem

We next consider an inverse problem consisting in recovering the regularized solution to a system of nonlinear ODEs. We seek parameters $x_{\text{true}} \in \mathbb{R}^n$ given observed noisy data $b = F(x_{\text{true}}) + \varepsilon$ where $F : \mathbb{R}^n \to \mathbb{R}^m$ and $\varepsilon \sim \mathcal{N}(0, 0.1)$. The data generating mechanism F is given by the FitzHugh [54] and Nagumo et al. [103] model for neuron activation

$$\frac{\mathrm{d}V}{\mathrm{d}t} = (V - V^3/3 - W + x_1)x_2^{-1}, \quad \frac{\mathrm{d}W}{\mathrm{d}t} = x_2(x_3V - x_4W + x_5), \tag{5.49}$$

which, if $x_1 = x_4 = x_5 = 0$, becomes the Van der Pol [144] oscillator

$$\frac{\mathrm{d}V}{\mathrm{d}t} = (V - V^3/3 - W)x_2^{-1}, \quad \frac{\mathrm{d}W}{\mathrm{d}t} = x_2(x_3V). \tag{5.50}$$

Both models are highly nonlinear and ill-conditioned.

We use initial conditions $(V, W) = (2, 0)$ and discretize the time interval $[0, 20]$ at 0.2 second increments. For given x, let $V(t; x)$ and $W(t; x)$ be solutions of (5.49). Define variables $v_i(x) \approx V(t_i; x)$, $w_i(x) \approx W(t_i; x)$, $i = 1, \ldots, n+1$ where $n = 20/0.2 = 100$. We set $F(x) := (v(x), w(x))$, where $v(x) := (v_1(x), \ldots, v_{n+1}(x))$ and $w(x) := (w_1(x), \ldots, w_{n+1}(x))$. We generate b using $x_{\text{true}} = (0, 0.2, 1, 0, 0)$, which corresponds to a solve of the Van der Pol oscillator. To recover x, we solve

$$\underset{x}{\text{minimize}} \ \tfrac{1}{2}\|F(x) - b\|_2^2 + h(x), \tag{5.51}$$

with $h(x) = \|x\|_0$. ODE solves are performed with the DifferentialEquations.jl package [115], which features an mechanism for choosing the solver, and provides $\nabla v(x)$ and $\nabla w(x)$ by way of automatic differentiation. We set $\epsilon = 10^{-3}$ in all methods, the maximum iterations to 500, and use an LBFGS approximation of the Hessian. For Algorithm 16, the maximum number of inner iterations is 5000.

Table 5.2 summarizes our results and Figure 5.2 shows overall data fit and objective function traces. Algorithm 16 with either PG or R2 as subsolver, as well as ZeroFPR, correctly identified the nonzero pattern of x with reasonable error in the nonzero elements. PANOC performs well initially, but its linesearch routine terminates prematurely as it generates a step length that is below a preset tolerance of 10^{-7}. At that point, PANOC terminates. ZeroFPR performs well, but needs many iterations to decrease the objective value to the same level as Algorithm 16. As in section 5.6.1, Algorithm 16 converges with significantly fewer gradient evaluations than ZeroFPR, though with a significant number of proximal operator evaluations. However, gradient evaluations in (5.49) are far more expensive and time consuming than proximal evaluations. Figure 5.2 also reveals that the final iterate generated by Algorithm 16 and ZeroFPR results in trajectories that are visually indistinguishable from those associated with the exact solution. Algorithm 16 with Algorithm 17 as a subsolver reaches a similar solution as ZeroFPR, but requires much fewer proximal and gradient evaluations. The results appear in Table 5.2. Plots are nearly identical to those in Figure 5.2, and are hence omitted.

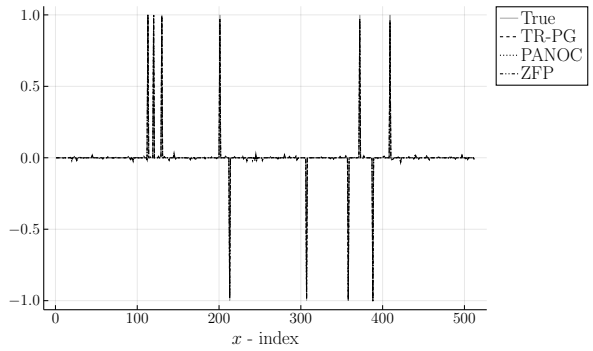
Table 5.2: Results for Algorithm 16 with proximal gradient (TR-PG) and Algorithm 17 (TR-R2) subsolvers, PANOC, and ZeroFPR applied to (5.49) with $h = \|\cdot\|_0$, $\Delta\mathbb{B}_\infty$ and LBFGS approximation.

Parameters										
True	TR-PG	TR-R2	PANOC	ZFP	Measure	True	TR-PG	TR-R2	PANOC	ZFP
0	0	0	0.840	0	$f(x)$	1.058	1.078	1.266	73.888	1.048
0.2	0.170	0.130	0.690	0.188	$h(x)$	2	2	3	5	3
1.0	1.136	1.408	0.952	1.048	$\|\|x - x_{\text{true}}\|\|_2$	0	0.139	0.427	1.636	0.051
0	0	0.107	0.983	0.010	∇f evals		76	61	43	422
0	0	0	0.874	0	$\text{prox}_{\nu\psi}$ calls		60143	22617	30	421

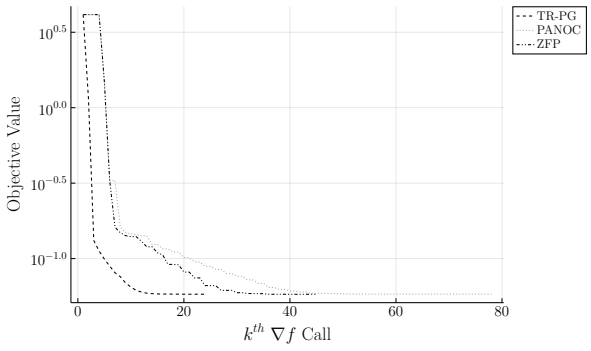
We also compare Algorithm 17 to our own implementation of a standard proximal gradient with linesearch on (5.51). We set the stopping tolerance for both to 10^{-3}. Table 5.3 summarizes our results and Figure 5.3 shows overall data fit and objective function traces. Both Algorithm 17 and proximal gradient descent converge much slower than Algorithm 16, where we use curvature information. Neither algorithm correctly identified the nonzero pattern of x within 5000 iterations, although Algorithm 17 descends considerably faster than proximal gradient descent, and attains the stopping tolerance. Figure 5.3 reveals that the final iterate generated by Algorithm 17 is closer to the solution than that of proximal gradient descent, though both terminated far from the correct answer.

Table 5.3: Results for Algorithm 17 (R2) and proximal gradient descent (PG) applied to (5.49) with $h = \|\cdot\|_0$.

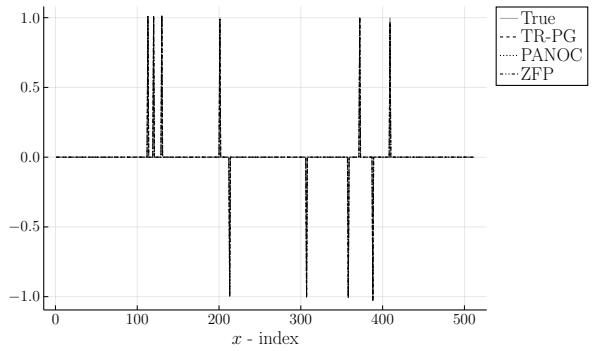
Parameters						
True	R2	PG	Measure	True	R2	PG
0	0	0.228	$f(x)$	1.058	3.852	24.246
0.200	0.142	0.245	$h(x)$	2	4	5
1.000	1.392	1.083	$\|\|x - x_{\text{true}}\|\|_2$	0	0.737	1.045
0	0.621	0.916	∇f evals		3892	5010
0	0.022	0.440	$\text{prox}_{\nu\psi}$ calls		8891	5009



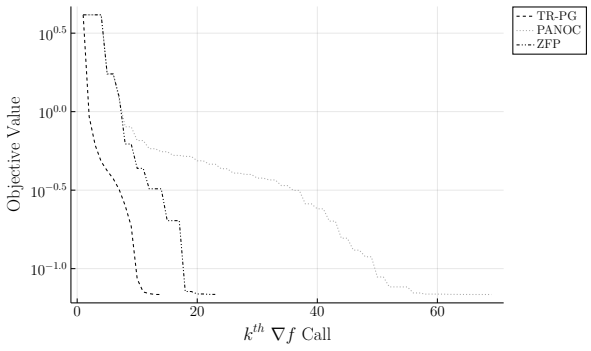
(a) Signal: $h = \lambda\|\cdot\|_1$, $\Delta\mathbb{B}_2$



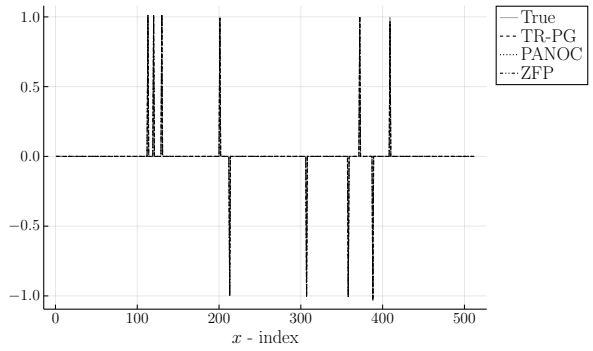
(b) History: $h = \lambda\|\cdot\|_1$, $\Delta\mathbb{B}_2$



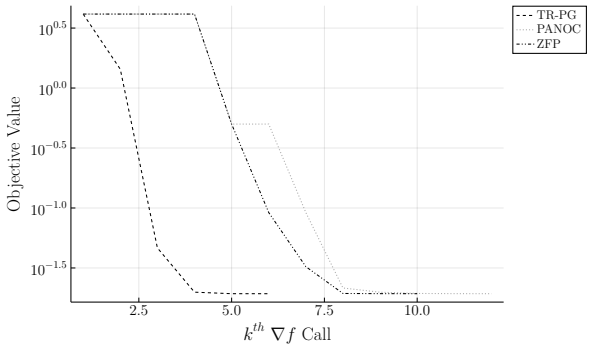
(c) Signal: $h = \lambda\|\cdot\|_0$, $\Delta\mathbb{B}_\infty$



(d) History: $h = \lambda\|\cdot\|_0$, $\Delta\mathbb{B}_\infty$

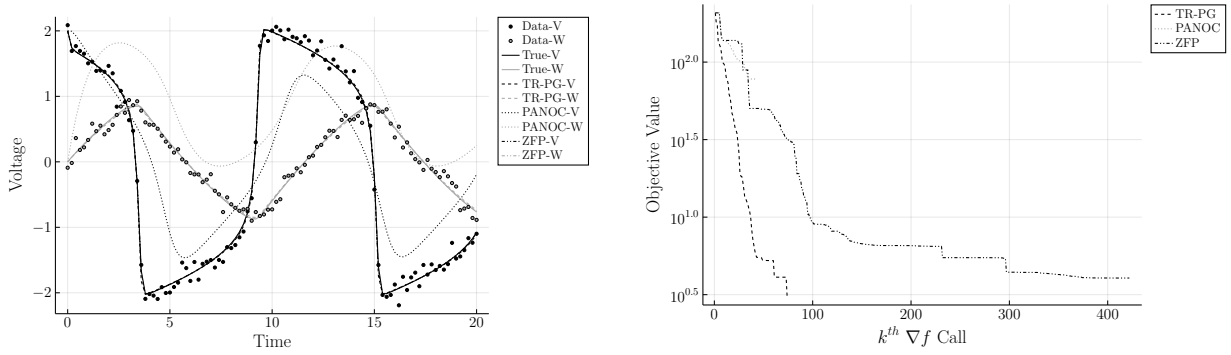


(e) Signal: $h = \chi(\cdot;\lambda\mathbb{B}_0)$, $\Delta\mathbb{B}_\infty$



(f) History: $h = \chi(\cdot;\lambda\mathbb{B}_0)$, $\Delta\mathbb{B}_\infty$

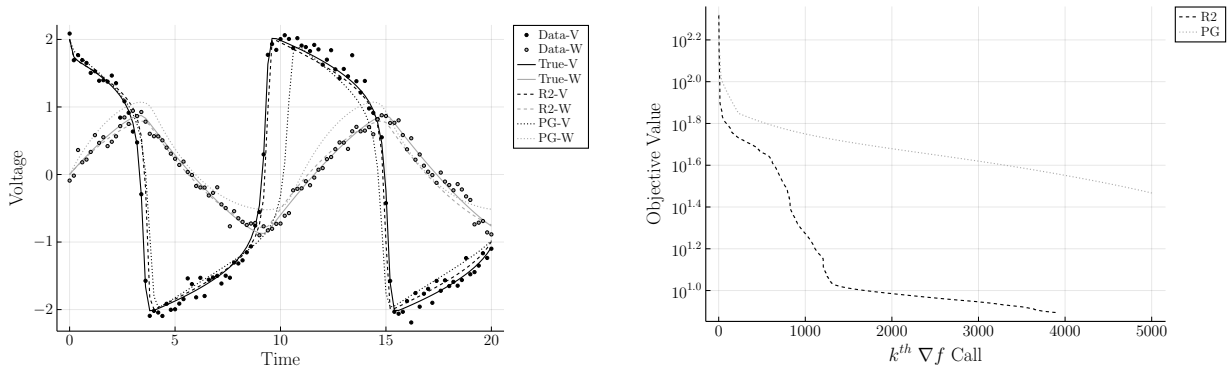
Figure 5.1: BPDN results (5.48) with Algorithm 16 and a proximal gradient subsolver (TR-PG), PANOC and ZeroFPR (ZFP): Signal plots (left) and objective value history (right). $\Delta\mathbb{B}_p$ indicates the norm used to define the trust region.



(a) Solution with data

(b) Objective Function (5.51) history

Figure 5.2: Solution of (5.49) with $h(x) = \|x\|_0$ in (5.51), $\Delta\mathbb{B}_\infty$ and LBFGS approximation.



(a) Solution with data

(b) Objective Function (5.51) history

Figure 5.3: Solution of (5.49) for $h = \|\cdot\|_0$ in (5.51) with PG with linesearch and Algorithm 17.

5.7 Discussion and perspectives

We demonstrated the performance of trust-region methods using nonconvex LSR1 quasi-Newton models against two linesearch methods using LBFGS models, and observed faster convergence curves with fewer gradient evaluations. Many regularizers in (5.1) have a closed-form or efficiently-computable proximal operator, whose cost is often dominated by that of a function or gradient evaluation in a large inverse problem.

The worst-case iteration complexity bound of Algorithm 16 matches the best known bound for trust-region methods in smooth optimization. Algorithm 17, a first-order method that is related to the proximal gradient method with adaptive steplength, does not require prior knowledge or estimation of a Lipschitz constant, and has a straightforward complexity analysis similar to that of Algorithm 16. In practice, using curvature information in Algorithm 16 proved useful for efficiently estimating highly nonlinear nonsmooth models. Convergence of trust-region methods for smooth optimization can be established even if Hessian approximations are unbounded, provided they do not deteriorate too fast. It may be possible to generalize our analysis along similar lines.

Interesting directions left to future work include implementation and analysis for *inexact* function, gradient, and proximal operator evaluations, and extensions of our results to cubic regularization, and more general nonlinear stepsize control-type methods, such as those of [59].

Chapter 6

GUIDING ADMM CONVERGENCE WITH FILTER METHODS

6.1 Introduction and Problem Formulation

As opposed to Chapter 5, we tackle nonlinear problems but with splitting algorithms. The Alternating Direction Method of Multipliers (ADMM) Algorithm 5 was briefly described in Chapter 2. ADMM has had much success in convex optimization [90], but minimizing some nonconvex functions produces mixed results. ADMM has been shown to succeed for some applications; NMF/C, phase retrieval, compressed sensing, and image restoration [148]. While insights have been made into why ADMM converges for these problems [148], it has also been shown to diverge on 3-block convex problems. In addition, most (exceptions being [63]) assume that the constraints in the block variables are linear. Indeed, ADMM has only been shown to converge for nonlinear constraints in very specific applications [63].

In this work, we have made some insights as to guiding ADMM convergence for nonconvex, nonlinear block problems with nonlinear constraints. We do this primarily by leveraging filter methods for an existing problem class. We address optimization problems of the form

$$\min_{\mathbf{x}\in\mathbb{R}^n} \; f(\mathbf{x}) \quad \text{s.t. } c(\mathbf{x}) = 0, \; \mathbf{x} \in \Omega, \tag{6.1}$$

where $f : \mathbb{R}^n \to \mathbb{R}$ and $c : \mathbb{R}^n \to \mathbb{R}^m$ are twice continuously differentiable. Our variable is $\mathbf{x} := [x_0; \ldots; x_p] \in \mathbb{R}^n$ where $n = \sum_{i=0}^{m} n_i$. By $\Omega \subseteq \mathbb{R}^n$, we mean a simple feasible set; that is, one which admits an efficient and well-defined projection (2) It is well known that proj_Ω is well defined (i.e. that it exists and is unique) when the underlying set Ω is nonempty, closed, and convex. In this application, we focus on simple bound constraints $\Omega = \{\mathbf{x} : l \leq \mathbf{x} \leq u\}$, for which we have the projection

$$\left[\underset{\Omega}{\text{proj}}(\mathbf{x})\right]_i = \min\{\max\{l_i, x_i\}, u_i\} \qquad i = 1, \ldots, n. \tag{6.2}$$

Simple restriction operators also fit within this framework. If we let $S \subset \{1, 2, \ldots, n\}$ to be the set of observed entries, and define a sampling operator as we do with $\mathcal{A}$ in Chapter 4. One can write Ω to accomodate such a sampling operator in its projection:

$$\Omega_S = \{x_i : l_i \leq x_i \leq u_i\}_{i=1}^n, \quad \begin{cases} l_i = -\infty, u_i = \infty, & i \in S \\ l_i = u_i = 0, & i \notin S \end{cases}. \tag{6.3}$$

This implies that the sampling operator $\mathcal{A}$ is (6.2) with Ω_S

$$\mathcal{A}(\mathbf{x}) = \underset{\Omega_S}{\text{proj}}(\mathbf{x}). \tag{6.4}$$

The first-order optimality conditions of (6.1) for a local minimum $\mathbf{x}_*$ can be expressed by

$$\underset{\Omega}{\text{proj}}(\mathbf{x}_* - \nabla_{\mathbf{x}}\mathcal{L}(\mathbf{x}_*, y)) = \mathbf{x}_* \tag{6.5a}$$

$$c(\mathbf{x}_*) = 0. \tag{6.5b}$$

This is equivalent to enforcing

$$\frac{\partial \mathcal{L}}{\partial x_i}(\mathbf{x}_*) \begin{cases} \geq 0 & x_{*,i} = l_i \\ = 0, & x_{*,i} \in (l_i, u_i) \\ \leq 0, & x_{*,i} = u_i \end{cases}$$

for $\mathbf{x}_*$ being a local minimizer.

6.2 Augmented Lagrangian Filter Algorithm

Given a penalty parameter $\rho \geq 0$, the augmented Lagrangian for (6.1) is defined by

$$\mathcal{L}_\rho(\mathbf{x}, y) = f(\mathbf{x}) - y^T c(\mathbf{x}) + \tfrac{\rho}{2}\|c(\mathbf{x})\|^2 = \mathcal{L}(\mathbf{x}, y) + \tfrac{\rho}{2}\|c(\mathbf{x})\|^2. \tag{6.6}$$

For a given penalty parameter ρ_k and multipliers y_k, minimizing $\mathcal{L}_{\rho_k}(\mathbf{x}, y_k)$ over $x \in \Omega$ yields the simply constrained subproblem.

$$\min_{\mathbf{x}\in\Omega} \mathcal{L}_{\rho_k}(\mathbf{x}, y_k). \tag{6.7}$$

which may not have a unique solution. A basic augmented Lagrangian scheme (approximately) solves (6.6) to obtain $\mathbf{x}_{k+1}$ and then updates the multipliers y either via the first-order update

$$y_{k+1} = y_k - \rho_k c(\mathbf{x}_{k+1}) \tag{6.8}$$

or by keeping $y_{k+1} = y_k$ and increasing ρ_k. Now, we introduce terms η_k and ω_k that represent infeasibility and first order error, respectively. We define these as

$$\eta(x) := \|c(\mathbf{x})\| \tag{6.9a}$$

$$\omega(x, y) := \|\operatorname*{proj}_{\Omega}(\mathbf{x} - \nabla_{\mathbf{x}}\mathcal{L}(\mathbf{x}, y)) - \mathbf{x}\| \tag{6.9b}$$

where $\omega_0(\mathbf{x}, y)$ is the dual feasibility error of the NLP (6.1). We also note that first-order multiplier update implies that

$$\begin{aligned}\nabla_{\mathbf{x}}\mathcal{L}(\mathbf{x}_{k+1}, y_{k+1}) &= \nabla f(\mathbf{x}_{k+1}) - \nabla c(\mathbf{x}_{k+1})^T y_k + \rho_k \nabla c(\mathbf{x}_{k+1})^T c(\mathbf{x}_{k+1}) \\ &= \nabla \mathcal{L}_{\rho_k}(\mathbf{x}_{k+1}, y_k)\end{aligned}$$

which means

$$\omega_0(\mathbf{x}_{k+1}, y_{k+1}) = \omega_{\rho_k}(\mathbf{x}_{k+1}, y_k).$$

Hence, we can monitor the dual infeasibility error of the original problem whilst solving the augmented Lagrangian. For our proposed algorithm, we consider *optimal* stopping criteria to be the relative first order error ω_{ρ_k} (6.9b), that is

$$\frac{\|\operatorname{proj}_{\Omega}(\mathbf{x} - \nabla_{\mathbf{x}}\mathcal{L}_0(\mathbf{x}_k, y_k)) - \mathbf{x}\|}{\|\operatorname{proj}_{\Omega}(\mathbf{x} - \nabla_{\mathbf{x}}\mathcal{L}_0(\mathbf{x}_0, y_0)) - x\|} = \frac{\omega_0(\mathbf{x}_k, y_k)}{\omega_0(\mathbf{x}_0, y_0)} \leq \varepsilon \tag{6.10}$$

where $\mathbf{x}_0$ is the initial iterate. Next, we define what an augmented Lagrangian Filter is and how a point is accepted.

Definition 9 (Augmented Lagrangian Filter and Acceptance)**.** *A filter $\mathcal{F}$ is a list of pairs $(\eta_l, \omega_l) := (\eta(\mathbf{x}_l), \omega_0(\mathbf{x}_l, y_l))$ such that no pair dominates another pair, i.e., there exists no pairs $(\eta_l, \omega_l), (\eta_k, \omega_k)$, $l \neq k$ such that $\eta_l \leq \eta_k$ and $\omega_l \leq \omega_k$. A point $(\mathbf{x}_k, y_k)$ is acceptable to the filter $\mathcal{F}$ iff*

$$\eta_k := \eta(\mathbf{x}_k) \leq \beta\eta_l \quad \textit{or} \quad \omega_k := \omega_0(\mathbf{x}_k, y_k) \leq \omega_l - \gamma\eta(\mathbf{x}_k), \ \forall\ (\eta_l, \omega_l) \in \mathcal{F} \tag{6.11}$$

where $0 < \gamma, \beta < 1$.

Definition 9 is depicted in Figure 6.1. In essence, the filter is used to enforce a reasonable feasibility to convergence ration; if a point minimizes ω to much without also changing feasibility, then a linesearch routine is activated. Points are accepted when they are either more feasible or have smaller ω than all other points, i.e. are not dominated by another entry.

We assume our Lagrangian is separable with a block structure with $i = 1, \dots p$ blocks. For k^{th} out iteration $\mathcal{L}_k = \mathcal{L}_{\rho_k}(\mathbf{x}_k, y_k)$, update i^{th} block from the $(j-1)^{th}$ to j^{th} iterate with the notation

$$\begin{aligned}
\mathcal{L}_{k,ij} &= \mathcal{L}_{\rho_k}(\hat{\mathbf{x}}_{j,<i}, \hat{x}_{j-1,i}, \hat{\mathbf{x}}_{j-1,>i}, y_k) \\
&\underbrace{\text{intermediate } i^{th} \text{ step}}_{(j-1) \to\ j} \\
\hat{\mathcal{L}}_{k,ij} &= \mathcal{L}_{\rho_k}(\hat{\mathbf{x}}_{j,<i}, \hat{x}_{j,i}, \hat{\mathbf{x}}_{j-1,>i}, y_k).
\end{aligned}$$

The Lagrangian evaluated at the current point is $\mathcal{L}_k$, while $\mathcal{L}_{k,ij}$ is the intermediate i^{th} step, and $\hat{\mathcal{L}}_{k,ij}$ is the intermediate end step for the i^{th} block. This means we are optimizing each intermediate block step that satisfies

$$\mathcal{L}_{k,ij} - \hat{\mathcal{L}}_{k,ij} \geq \sigma \|\nabla_i \mathcal{L}_{k,ij}\|_2^2$$

where ∇_i is the gradient with respect to x_i. A telescoping sum implies

$$\mathcal{L}_k - \mathcal{L}_k(\hat{\mathbf{x}}_j, y_k) \geq \sigma \sum_{i=1}^{m} \|\nabla_i \mathcal{L}_{k,ij}\|_2^2$$

which is not sufficient reduction.

With this in hand, we state our ADMM Filter in Algorithm 18.

Nominally, one chooses positive constants that get exponentiated such that ρ_k always increases, $\xi = \{1.1, 2, \dots, 10, \dots\}$. Currently, we use the update scheme

$$\xi = \max\left(1.1, \frac{\hat{\eta_j}^2}{\Delta \mathcal{L}_{\rho_k}^{in}}\right) \tag{6.12}$$

where

$$\Delta \mathcal{L}_{\rho_k}^{in} = \mathcal{L}_{\rho_k}(\mathbf{x}_k, y_k) - \mathcal{L}_{\rho_k}(\hat{\mathbf{x}}_j, y_k)$$

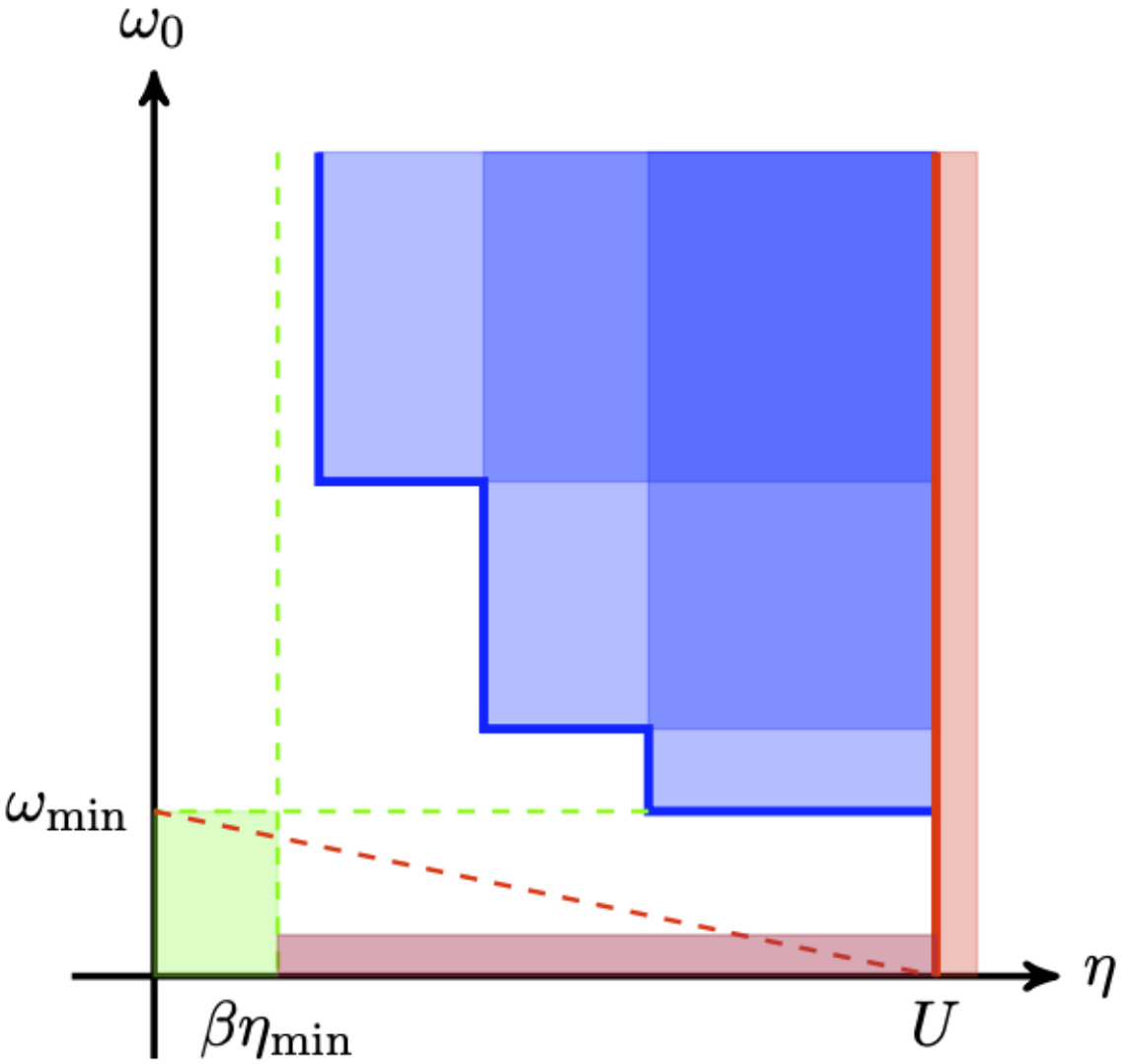


Figure 6.1: Example acceptance from [145]. Example of an augmented Lagrangian filter $\mathcal{F}$ with three entries. The filter is in blue, the dashed green line shows the envelope in feasibility η, and the upper bound U (red line) is implied by the sloping envelope condition (6.9a) and $\omega_0 \geq 0$. Values above and to the right of the filter are not acceptable. The green area shows the set of filter entries that are guaranteed to be acceptable, and the shaded purple area is the set of entries that trigger the switch to restoration.

Algorithm 18 ADMM-F for (6.6)

Input: functions and data.
Choose $\mathbf{x}_0,\ y_0,\ \rho_0$.
while $(\mathbf{x}_k, y_k)$ not optimal (6.10) **do**
    $j \leftarrow 0,\ \texttt{Restflag} \leftarrow 0$
    Declare temporary variable: $\hat{\mathbf{x}}_j \leftarrow \mathbf{x}_k$
    while (η_j, ω_j) not acceptable to $\mathcal{F}_k$ **do**
        while $\mathcal{L}_{\rho_k}(\mathbf{x}_k, y_k) - \mathcal{L}_{\rho_k}(\hat{\mathbf{x}}_j, y_k) < \sigma\omega_{\rho_k}(\mathbf{x}_k, y_k)$ **do**
            for $i = 1 \dots p$ with p blocks **do**
                $\hat{x}_{j+1,i} \leftarrow \arg\min_{x_i} \mathcal{L}_{\rho_k}(\mathbf{x}_{j+1,<i}, x_{j,i}, \mathbf{x}_{j,>i}, y_k)$
            $\hat{\mathbf{x}}_{j+1} \leftarrow \hat{\mathbf{x}}_{j+1,1:p}$, and $j \leftarrow j+1$
        if Restoration condition (6.14) **then** holds
            $\texttt{Restflag} \leftarrow 1$
            Find $\hat{\mathbf{x}}_{j+1}$ s.t. $(\hat{\eta}_{j+1}, \hat{\omega}_{j+1})$ acceptable, or $\hat{\mathbf{x}}_{j+1} \leftarrow \arg\min_x \|c(\mathbf{x})\|^2 \quad \text{s.t. } \mathbf{x} \in \Omega$
            $j \leftarrow j+1$
        Compute $\omega_{\rho_k}(\hat{\mathbf{x}}_j, y_k), \eta(\hat{\mathbf{x}}_j, y_k)$
    $(\eta_k, \omega_k) \leftarrow (\omega_{\rho_k}(\hat{\mathbf{x}}_j, y_k), \eta(\hat{\mathbf{x}}_j, y_k))$
    if $\texttt{Restflag} = 1$ **then**
        Increase Penalty $\rho_{k+1} \leftarrow \max(\rho_k, \xi\rho_k)$
    else
        $\mathbf{x}_{k+1} \leftarrow \hat{\mathbf{x}}_j$
        $y_{k+1} \leftarrow y_k - \rho_k c(\hat{\mathbf{x}}_j)$
        Add (η_k, ω_k) to $\mathcal{F}_k$
    $k \leftarrow k+1$

and $\frac{\hat{\eta_j}^2}{\Delta\mathcal{L}^{in}_{\rho_k}}$ taken from [145, Equation 4.21]. The augmented Lagrangian method is a dual method, and ρ_k can be interpreted as the stepsize of the steepest descent direction, or as the first order multiplier update. It remains to develop the exact filter specifications as well as how we measure sufficient reduction in the augmented Lagrangian. Currently, we were thinking

$$\mathcal{L}_{\rho_k}(\mathbf{x}_k, y_k) - \mathcal{L}_{\rho_{k+1}}(\mathbf{x}_{k+1}, y_{k+1}) \geq \sigma\omega(\mathbf{x}_k, y_k), \tag{6.13}$$

for $\sigma \in (0, 1)$, but ADMM is difficult in that the gradient of the Lagrangian is taken in blocks at different points. This equation is very aggressive and sees the best performance increases when ρ_k is a poor penalty parameter. The standard reduction is given in Vanaret and Leyffer [145, Equation 3.15 & 3.16]:

$$\hat{\eta}(\mathbf{x}) = \eta(\hat{\mathbf{x}}_{j+1}) \geq \beta U \tag{6.14a}$$

$$\omega_{\rho_k}(\hat{\mathbf{x}}_{j+1}, y_k) =\leq \epsilon \quad \text{and} \quad \eta(\hat{\mathbf{x}}_{j+1}) \geq \beta\eta_{\min}. \tag{6.14b}$$

Equation (6.13) is the sufficient reduction condition from the augmented Lagrangian Filter, which follows easily from a Cauchy-step condition of the inner minimization of the augmented Lagrangian. We require that the block coordinate descent part of the algorithm to achieve first-order sufficient reduction in Line 7 of Algorithm 18. However, we do not minimize jointly in blocks of $\mathbf{x}$, and hence we may not get this relationship exactly. In fact, we are only guaranteed this in some circumstances.

6.2.1 Block Coordinate Descent Convergence

In this discussion, we summarize the convergence criteria for block coordinate descent methods, and try to apply them for the BCD Algorithm 4 part of Algorithm 18. Convergence of block coordinate descent methods typically require that the function f be strictly convex (or quasiconvex or hemivariate) differentiable, and has bounded level sets (if we take into account bound constraints) [139]. This has since been relaxed to pseudoconvexity (convex around local minima), which allows f to have a nonunique minimum along coordinate directions.

Convergence has also been shown for non(pseudo)convex f, such as when f is quadratic, or when f is strictly pseudoconvex in each of the $p-2$ coordinate blocks, or when f has unique minima in each coordinate block. In [139], the author considers nondifferentiable (nonconvex) functions but where the nondifferentiable part of f is separable. In the BCD of Algorithm 18, we perform a BCD method on the primal variables until

$$\mathcal{L}_{\rho_k}(\mathbf{x}_k, y_k) - \mathcal{L}_{\rho_k}(\hat{\mathbf{x}}_j, y_k) =: \Delta\mathcal{L}^{in}_{\rho_k} > \hat{\omega}_{\rho_k}(\hat{\mathbf{x}}_j, y_k). \tag{6.15}$$

Currently, the restoration switching condition is

$$\sigma\omega_{\rho_k}(\hat{\mathbf{x}}_j, y_k) < \Delta\mathcal{L}^{in}_{\rho_k} < \rho_k\eta(\hat{\mathbf{x}}_j)^2$$

which essentially increases the ρ_k along the Pareto curve $(\rho_k\eta(\hat{\mathbf{x}}_j)^2, \sigma\omega_{\rho_k}(\hat{\mathbf{x}}_j, y_k))$. This relationship is where we decided to use (6.12) in our ρ_k update. It is believed, as of writing this thesis, that this result is enough for the convergence of the inner-part of Algorithm 18. Future work will resolve this in the context of [145].

6.3 Nonconvex Bilinear Optimization: Nonnegative Matrix Factorization and Completion

Next, we turn to the motivating example for this endeavor - ADMM for NMF/C. While we use this as an application, and indeed extent our algorithm to nonlinear NMF/C examples, we by no means try to out-perform NMF/C-specific algorithms. Indeed, we show that while our algorithm has more flexibility, we generally converge slower than the current state-of-the-art [3] (Figure 6.2).

6.3.1 Matrix Decomposition

We can consider the following matrix decomposition model

$$\min_{X,Y,Z} p(X) + \sum_{i=1}^{m-1} \|Y_i - Y_{i+1}\|_2 + \|Z\|_F^2 \quad \text{s.t. } V = X + Y + Z, \tag{6.16}$$

where $X, Y, Z, V \in \mathbb{R}^{n\times m}$, Y_i is the i^{th} column of Y, $\|\cdot\|_F$ is the Frobenius norm, and $p(X)$ is any lower bounded, lower semi-continuous penalty function. For example, this could be the Schatten-q quasi-norm $\|X\|_q$ $q \in (0,1)$

$$\|A\|_q = \sum_{i=1}^{n} \sigma_i^q(A),$$

where $\sigma_i(A)$ is the i^{th} largest singular value of A. This is nonconvex for $q \in (0,1)$. ADMM for this particular problem would take the form of Algorithm 19. The general form of

Algorithm 19 ADMM for (6.16)

Input: functions $p, \|\cdot\|_F, \rho > 0$.

Choose X_0, Y_0, Z_0, Λ_0.

for $k = 0, 1, \ldots$ **do**

$X_{k+1} = \arg\min_X p(X) + \frac{\rho}{2}\|X + Y_k + Z_k - V + \rho^{-1}\Lambda_k\|_F^2$

$Y_{k+1} = \arg\min_Y \sum_{i=1}^m \|Y_i - Y_j\| + \frac{\rho}{2}\|X_{k+1} + Y + Z_k - V + \rho^{-1}\Lambda_k\|_F^2$

$Z_{k+1} = \arg\min_Z \|Z\|_F^2 + \frac{\rho}{2}\|X_{k+1} + Y_{k+1} + Z - V + \rho^{-1}\Lambda_k\|_F^2$

$\Lambda_{k+1} = \Lambda_k + \rho(X_{k+1} + Y_{k+1} + Z_{k+1} - V)$

these problems contains the nonconvex constraint $XY = Z$ which is not simple addition as in (6.16). We first consider Nonnegative Matrix Factorization (NMF) [20, 85, 86]

$$\min_{X,Y,Z} \frac{1}{2}\|Z - M\|_F^2 \quad \text{s.t. } Z = XY,\ X, Y \geq 0, \tag{6.17}$$

where $M \in \mathbb{R}^{P\times Q}$ is the data matrix and the two factors are $X \in \mathbb{R}^{P\times K}, Y \in \mathbb{R}^{K\times Q}$ with $K \ll \min(Q, P)$ and $M = XY$ with both $X, Y \geq 0$. Hence, $\Omega = \{x :\ X, Y \geq 0\}$.

6.3.2 Nonnegative Matrix Factorization

ADMM can be used to solve (6.17), in addition to other methods. This is given in Algorithm 20 and [63]. In [63], the authors claim that the convergence of other algorithms, namely in [148], are based on nonstandard assumptions that the successive difference of the

iterates goes to zero, which are impossible to verify *a priori*, as they are assumptions on the progress of a method rather than the problem itself. The authors [63] also show that their

Algorithm 20 ADMM for (6.17)

1: **Input:** functions and data $\|\cdot\|_F, M$.
2: Choose X_0, Y_0, Z_0, Λ_0.
3: **for** $k = 0, 1, \ldots$ **do**
4: $\quad Y_{k+1} = \arg\min_{Y \geq 0} \frac{\rho}{2}\|Z_k - X_k Y + \rho^{-1}\Lambda_k\|_F^2$
5: $\quad (X_{k+1}, Z_{k+1}) = \arg\min_{X \geq 0, Z} \|Z - M\|_F^2 + \frac{\rho}{2}\|Z - XY_{k+1} + \rho^{-1}\Lambda_k\|_F^2$
6: $\quad \Lambda_{k+1} = \Lambda_k + \rho(Z_{k+1} - X_{k+1}Y_{k+1})$

scheme produces iterates that converge, for specific ρ values. We suggest a method that choses ρ automatically based on the filter acceptance critera Definition 9.

ADMM Filter for NMF (6.17)

Here, we establish our own scheme to solve Equation (6.17). In this, we write down the augmented Lagrangian

$$\mathcal{L}_\rho(X, Y, Z, \Lambda) = \|M - Z\|_F^2 + \operatorname{tr}\Big(\Lambda^T(Z - XY)\Big) + \frac{\rho}{2}\|Z - XY\|_F^2 \tag{6.18}$$

where $\operatorname{tr}\Big(\Lambda^T(Z - XY)\Big)$ is the Frobenius inner product. We also define η and ω to be

$$\eta(x) := \|Z - XY\| \tag{6.19a}$$

$$\omega_\rho(x, y) := \left\| \operatorname*{proj}_{\Omega}([X, Y, Z]^T - \nabla_{X,Y,Z}\mathcal{L}_{\rho_k}(X, Y, Z, \Lambda)) - [X, Y, Z]^T \right\| \tag{6.19b}$$

Our scheme, which we denote ADMM-F or ADMM Filter, is described in Algorithm 21. We also note that the competition [63] first bound the successive difference of the dual variables by the successive different in the primal ones, and then show that the augmented Lagrangian is a decreasing function that is bounded from below. Namely, [63, Lemma 1] shows the successive bounding and then [63, Lemma 2] bounds the successive difference of the augmented Lagrangian function.

Algorithm 21 ADMM-F for (6.17)

Input: functions and data $\|\cdot\|_F, M$.

Choose $x_0 = [X_0, Y_0, Z_0],\ y_0 = \Lambda_0,\ \rho_0$.

while $(\mathbf{x}_k, y_k)$ not optimal (6.10) **do**

  $j \leftarrow 0,\ \texttt{Restflag} \leftarrow 0$

  Declare temporary variable: $\hat{\mathbf{x}}_j \leftarrow \mathbf{x}_k$

  while (η_j, ω_j) not acceptable to $\mathcal{F}_k$ **do**

    while $\mathcal{L}_{\rho_k}(\mathbf{x}_k, y_k) - \mathcal{L}_{\rho_k}(\hat{\mathbf{x}}_j, y_k) < \sigma\omega_{\rho_k}(\mathbf{x}_k, y_k)$ **do**

      Block 1: $\hat{Y}_{j+1} \leftarrow \arg\min_{Y\geq 0} \mathcal{L}_{\rho_k}(\hat{X}_j, Y, \hat{Z}_j, \Lambda_k)$

      Block 2: $(\hat{X}_{j+1}, \hat{Z}_{j+1}) \leftarrow \arg\min_{X\geq 0,\ Z} \mathcal{L}_{\rho_k}(X, \hat{Y}_{j+1}, Z, \Lambda_k)$

      $\hat{\mathbf{x}}_{j+1} \leftarrow [\hat{X}_{j+1}, \hat{Y}_{j+1}, \hat{Z}_{j+1}]$, and $j \leftarrow j+1$

    if Restoration condition (6.14) for (6.19a) & (6.19b) **then**

      $\texttt{Restflag} \leftarrow 1$

      Find $\hat{\mathbf{x}}_{j+1}$ s.t. $(\hat{\eta}_{j+1}, \hat{\omega}_{j+1})$ acceptable, or $\hat{\mathbf{x}}_{j+1} \leftarrow \arg\min_x \|Z - XY\|^2$ s.t. $Y, X \geq 0$

      $j \leftarrow j+1$

    Compute $\omega_{\rho_k}(\hat{\mathbf{x}}_j, y_k), \eta(\hat{\mathbf{x}}_j, y_k)$ given by (6.19a) and (6.19b)

  if $\texttt{Restflag} = 1$ **then**

    Increase Penalty $\rho_{k+1} \leftarrow \max(\rho_k, \xi\rho_k)$ (6.12)

  else

    $\mathbf{x}_{k+1} \leftarrow \hat{\mathbf{x}}_j$

    $y_{k+1} \leftarrow \Lambda_k + \rho_k(\hat{Z}_j - \hat{X}_j\hat{Y}_j)$

    Add $(\eta_{k+1}, \omega_{k+1})$ to $\mathcal{F}_k$

  $k \leftarrow k+1$

6.3.3 Restoration Phase

In Algorithm 21, 'feasibility restoration' is an open question. Since notation is becoming difficult, we designate $\hat{\mathbf{x}}$ to be a potential update produced by a designated algorithm. As before, we let $(\hat{X}_{j+1}, \hat{Y}_{j+1}, \hat{Z}_{j+1})$ be the potential $(j+1)^{th}$ iterates (note the $\hat{\cdot}$) of the inner BCD problem of Algorithm 21. Currently, we assume $\hat{\eta}_j \geq U_\eta$ where

$$U_\eta := \max(\omega_{\min}/\gamma, \beta\eta_{\min})$$

with $\hat{\eta}_j = \|\hat{Z}_{j+1} - \hat{X}_{j+1}\hat{Y}_{j+1}\|_F^2 \geq U_\eta$. Here, $\omega_{\min}$ is the smallest first-order error of any filter entry; that is $\omega_{\min} := \min\{\omega_l : (\eta_l, \omega_l) \in \mathcal{F}\}$, with $\eta_{\min}$ being the corresponding η value. We note that γ, β are both constants in the interval $(0, 1)$ determined beforehand. Assume that $\hat{Z}_{j+1} \neq \hat{X}_{j+1}\hat{Y}_{j+1}$ is nonlinearly infeasible and that this iteration invokes the feasibility restoration phase. We know that $\tilde{Z} = \hat{X}_{j+1}\hat{Y}_{j+1}$ is feasible. We consider a linesearch-type routine to solve for $\hat{\eta}_j$. In it, we define our 'new' feasibility constraints as

$$\eta(\alpha) := \|\hat{Z}_{j+1} - \alpha(\tilde{Z} - \hat{Z}_{j+1}) - \hat{X}_{j+1}\hat{Y}_{j+1}\|_F^2 \tag{6.20}$$

$$\omega(\alpha) := \|\min\{l - \hat{\mathbf{x}}_{j+1}(\alpha), \max\{\hat{\mathbf{x}}_{j+1}(\alpha) - u, \nabla_x \mathcal{L}_{\rho_k}(\hat{\mathbf{x}}_{j+1}(\alpha), y_k)\}, \}\|_2 \tag{6.21}$$

where

$$\hat{\mathbf{x}}_{j+1}(\alpha) = [\hat{X}_{j+1}, \hat{Y}_{j+1}, \hat{Z}_{j+1} + \alpha(\tilde{Z} - \hat{Z}_{j+1})].$$

So what we do now is run a bisection method on α until we reach *filter acceptance* Definition 9, and then continue if proceeding entries in $\alpha+ = .01$ dominates the previous ones.

Lemma 6.3.1 (Filter Acceptance for α)**.** *At a filter unacceptable point given by the inner iteration of Algorithm 21 and $\eta(\alpha)$ given by Equation (6.20), there exists an $\alpha_{\min} \in (0, 1)$ such that $\hat{\mathbf{x}}_{j+1}(\alpha_{\min})$ is a filter acceptable point.*

Proof. By definition of $\eta(x)$ and $c(x) = Z - XY$, we have that the feasibility is a continuous function and $\eta(x) = 0$ iff $Z = XY$. We define $\hat{\mathbf{x}}_{j+1}(\alpha) = \hat{\mathbf{x}}_{j+1} + \alpha d$ where $d = [0, 0, (\hat{X}_{j+1}\hat{Y}_{j+1} - \hat{Z}_{j+1})]^T$, i.e. zeros for X and Y coordinates and exact feasibility for

the Z coordinate. This direction is also monotonic since as $\frac{d}{d\alpha}\|Z+\alpha(XY-Z)-XY\|^2 = (\alpha-1)\|XY-Z\|^2 \leq 0$ for $\alpha \in [0,1]$, thus, we have that as $\alpha \to 1$, $Z \to XY$ monotonically. Hence, for $\alpha = 1$ we have that $\eta(\alpha) = 0$ where $\alpha = 0$ leaves us within infeasibility territory by definition of restoration. Now, we have $\eta(\hat{\mathbf{x}}_{j+1}(\alpha))$ is the curve given by our line-search. By the Intermediate Value Theorem and monotonicity of $\eta(x)$, there exists an $\alpha_{\min} \in (0,1)$ such that $\eta(\hat{\mathbf{x}}_{j+1}(\alpha_{\min})) = \eta_{\min} > 0$, and since $\eta(\hat{\mathbf{x}}_{j+1}(\alpha_{\min}+\varepsilon)) < \eta_{\min},\ \forall \varepsilon > 0$, our linesearch is guaranteed to create a feasible point if we draw any $\alpha > \alpha_{\min}$. □

6.4 Nonnegative Matrix Factorization and Completion

One can also consider the NMF problem with missing values. This problem can be originally formulated as

$$\min_{X,Y} \frac{1}{2}\|\mathcal{A}(XY-M)\|_F^2 \quad \text{s.t. } X,Y \geq 0$$

where $S \subset \{1,2,\ldots,n\}$ contains the indices of known entries and $\mathcal{A}$ (recall (6.4)) denotes the projection onto the observed set S. In this case, S would denote the set of observed entries of the data M which we want to exactly match.

To solve this problem, we would reformulate the above to solve the problem

$$\min_{X,Y,Z,W} \frac{1}{2}\|Z-W\|_F^2 \quad \text{s.t. } X,Y \geq 0,\ Z = XY,\ \mathcal{A}(W-M) = 0. \tag{6.22}$$

Here, we define $\Omega = \{x :\ X,Y \geq 0,\ \mathcal{A}(W-M) = 0\}$, and ω, η as

$$\eta(x) := \|Z - XY\| \tag{6.23a}$$

$$\omega_\rho(x,y) := \left\| \operatorname*{proj}_{\Omega}([X,Y,Z,W]^T - \nabla_{X,Y,Z,W}\mathcal{L}_{\rho_k}(X,Y,Z,W,\Lambda)) - [X,Y,Z,W]^T \right\| \tag{6.23b}$$

A typical routine to solve (6.22) is given in Algorithm 22 in [63].

We produce a similar scheme to Algorithm 21, which is again modified from Algorithm 18. Note that the same restoration condition is utilized, since the nonlinear $c(x)$ is the same in both NMF and NMFC.

Algorithm 22 ADMM for (6.22)

1: **Input:** functions and data $\|\cdot\|_F, M$.

2: Choose X_0, Y_0, Z_0, U_0.

3: **for** $k = 0, 1, \ldots$ **do**

4: $(Y_{k+1}, W_{k+1}) = \arg\min_{Y\geq 0, \mathcal{P}_\Omega(W-M)=0} \frac{1}{2}\|Z_k - W\|_F^2 + \frac{\rho}{2}\|Z_k - X_k Y + \rho^{-1}\Lambda_k\|_F^2$

5: $(X_{k+1}, Z_{k+1}) = \arg\min_{X\geq 0, Z} \frac{1}{2}\|Z - W_{k+1}\|_F^2 \frac{\rho}{2}\|Z_k - X Y_{k+1} + \rho^{-1}\Lambda_k\|_F^2$

6: $\Lambda_{k+1} = \Lambda_k + \rho(Z_{k+1} - X_{k+1}Y_{k+1})$

6.5 Chemical Spectrum Application

The main application of this work is to find distributions of chemicals that occur in measured spectrum analysis. We assume that each chemical follows a Gaussian distribution, and would like to ascertain the nonnegative combination of Gaussians that reproduce the spectra data. We let μ, σ be the mean and standard deviation of each Gaussian, respectively. The *Intensity Function* as a function of wave number w and concentration c, with rank constraint K. Let m be the number of concentrations c. Our model then becomes

$$\hat{I}(w_i, c_j) = \sum_{k=1}^{K} \underbrace{h_{k,j}}_{f_k(c_j)} \underbrace{\exp\left(-\tfrac{1}{2}\left(\tfrac{w_i-\mu_k}{\sigma_k}\right)^2\right)}_{g_k(w_i)} = W_{i,:}H_{:,j} \tag{6.24}$$

where

$$W_{i,:} := W_{i,:}(\mu, \Sigma) = \left[\exp\left(-\tfrac{1}{2}\left(\tfrac{w_i-\mu_1}{\sigma_1}\right)^2\right), \ldots, \exp\left(-\tfrac{1}{2}\left(\tfrac{w_i-\mu_K}{\sigma_K}\right)^2\right)\right]$$

and $H_{:,j}$ is every row entry in the j^{th} column of the matrix H. This lends itself to a problem similar to the nonnegative matrix factorization problem

$$\min_{\mu,\Sigma,H} \sum_{i,j} (d_{i,j} - I(w_i, c_j))^2 \quad \text{s.t.} \quad H \geq 0 \tag{6.25}$$

where $I(w_i, c_j)$ is the calculated function and $d_{i,j}$ is the observed data. We can reformulate this into a nonlinear program similar to the nonnegative matrix factorization, given by

$$\min_{\mu,\Sigma,H,Z} \|D - Z\|_F^2 \quad \text{s.t.} \quad H \geq 0, \; Z = WH. \tag{6.26}$$

Algorithm 23 ADMM-F for (6.22)

Input: functions and data $\|\cdot\|_F, M$.

Choose $\mathbf{x}_0 = [X_0, Y_0, Z_0, W_0],\ y_0 = \Lambda_0,\ \rho_0$.

while $(\mathbf{x}_k, y_k)$ not optimal (6.10) **do**

  $j \leftarrow 0,\ \texttt{Restflag} \leftarrow 0$

  Declare temporary variable: $\hat{\mathbf{x}}_j \leftarrow \mathbf{x}_k$

  while (η_j, ω_j) not acceptable to $\mathcal{F}_k$ **do**

    while $\mathcal{L}_{\rho_k}(\mathbf{x}_k, y_k) - \mathcal{L}_{\rho_k}(\hat{\mathbf{x}}_j, y_k) < \sigma\omega_{\rho_k}(\mathbf{x}_k, y_k)$ **do**

      Block 1: $(\hat{Y}_{j+1}, \hat{W}_{j+1}) \leftarrow \arg\min_{Y\geq 0, \mathrm{proj}_\Omega(W-M)=0} \mathcal{L}_{\rho_k}(\hat{X}_j, Y, \hat{Z}_j, W, \Lambda_k)$

      Block 2: $(\hat{X}_{j+1}, \hat{Z}_{j+1}) \leftarrow \arg\min_{X\geq 0,\ Z} \mathcal{L}_{\rho_k}(X, \hat{Y}_{j+1}, Z, \hat{W}_{j+1}, \Lambda_k)$

      $\hat{\mathbf{x}}_{j+1} \leftarrow [\hat{X}_{j+1}, \hat{Y}_{j+1}, \hat{Z}_{j+1}, \hat{W}_{j+1}]$, and $j \leftarrow j+1$

    if Restoration condition (6.14) for (6.23a) & (6.23b) **then**

      $\texttt{Restflag} \leftarrow 1$

      Find $\hat{\mathbf{x}}_{j+1}$ s.t. $(\hat{\eta}_{j+1}, \hat{\omega}_{j+1})$ is acceptable, or $\hat{\mathbf{x}}_{j+1} \leftarrow \arg\min_x \|X - ZY\|^2$ s.t. $X, Y \geq 0, \mathcal{A}(W - M) = 0$

      $j \leftarrow j+1$

    Compute $\omega_{\rho_k}(\hat{\mathbf{x}}_j, y_k), \eta(\hat{\mathbf{x}}_j, y_k)$

  if $\texttt{Restflag} = 1$ **then**

    Increase Penalty $\rho_{k+1} \leftarrow \max(\rho_k, \xi\rho_k)$ (6.12)

  else

    $\mathbf{x}_{k+1} \leftarrow \hat{\mathbf{x}}_j$

    $y_{k+1} \leftarrow \Lambda_k + \rho_k(\hat{Z}_j - \hat{X}_j\hat{Y}_j)$

    Add $(\eta_{k+1}, \omega_{k+1})$ to $\mathcal{F}_k$

  $k \leftarrow k+1$

where $D \in \mathbb{R}^{m\times n}$ matrix of data filled by the intensity function, and $W \in \mathbb{R}^{m\times K}, H \in \mathbb{R}^{K\times n}$. We need to take the derivative of

$$\begin{aligned}\mathcal{L}_{\rho_k}(x,y) &= f(x) + y^T c(x) + \tfrac{\rho_k}{2}\|c(x)\|^2 \\ &= \|D - Z\|_F^2 + \langle \Lambda, Z - WH\rangle_F + \tfrac{\rho}{2}\|Z - WH\|_F^2 \\ &= \sum_{i,j}(d_{ij} - Z_{ij})^2 + \sum_{i,j}\Lambda_{ij}(Z_{ij} - W_{i,:}H_{:,j}) + \tfrac{\rho}{2}\sum_{i,j}(Z_{ij} - W_{i,:}H_{:,j})^2\end{aligned}$$

with respect to μ, Σ, and H. Noting that $\mathcal{L}_{\rho_k}(x,y)$ is singleton, we can say that with respect to one entry of the mean vector μ_l

$$\frac{\partial \mathcal{L}}{\partial \mu_l} = \sum_{i,j}\left(0 - \Lambda_{ij}\tfrac{\partial(W_{i,:}H_{:,j})}{\partial \mu_l} - \rho(Z_{ij} - W_{i,:}H_{:,j})\tfrac{\partial(W_{i,:}H_{:,j})}{\partial \mu_l}\right).$$

This requires derivative of $(W_{i,:}H_{:,j})$ with respect to μ_l, which we can write down as

$$\begin{aligned}\tfrac{\partial(W_{i,:}H_{:,j})}{\partial \mu_l} &= \tfrac{\partial}{\partial \mu_l}\sum_{k=1}^{L} e^{-\frac{1}{2}\left(\frac{w_i - \mu_k}{\sigma_k}\right)^2} H_{kj} \\ &= \left(\tfrac{w_i - \mu_l}{\sigma_l^2}\right) e^{-\frac{1}{2}\left(\frac{i-\mu_l}{\sigma_l}\right)^2} H_{lj}\end{aligned}$$

which we can plug back in to get

$$\frac{\partial \mathcal{L}}{\partial \mu_l} = \sum_{i,j}\left(-\Lambda_{ij} - \rho(Z_{ij} - (W_{i,:}H_{:,j}))\right)\left(\tfrac{i-\mu_l}{\sigma_l}\right) e^{-\frac{1}{2}\left(\frac{w_i-\mu_l}{\sigma_l^2}\right)^2} H_{lj}. \tag{6.27}$$

and similarly for σ_l but with $\sigma_l^{-1}\left(\tfrac{i-\mu_l}{\sigma_l}\right)^2 e^{-\frac{1}{2}\left(\frac{i-\mu_l}{\sigma_l}\right)^2} H_{lj}$. The derivatives of H and Z are the same as their counterparts in the nonnegative matrix factorization section.

6.6 Preliminary Numerical Results & Conclusions

Figure 6.3 shows preliminary numerical results for NMFC Algorithm 23 on an image of Argonne National Lab's logo. Figure 6.4 the same type of results for NMF Algorithm 21 on the same image. Both of these results converge fairly quickly, as shown in Figure 6.2. However, we sacrifice some speed for generalizability; ANLS and E-ANLS [3] converge faster

but are not generalizable and cannot solve (6.25) or (6.26). The results for the latter are shown in Figure 6.5. Here, we reduce the rank from 22 to 9 and allow for σ and μ to be optimized (as opposed to other formulations that hold these quantities fixed). We can see the reconstructed signal performs very well in Figure 6.5(b), and the means and standard deviations shift in Figure 6.5(c). Final and total filter entries are given by Figure 6.5(e) and Figure 6.5(f).

We again note that there is still much to accomplish in this line of thought that was not accomplished by the time this thesis was due. Namely,

1. Unifying the BCD [139] and the Filter [145] convergence theory;

2. Determining initial guesses for ρ_k, or generally examining ρ behavior;

3. Finding optimal Gaussians for the spectrum analysis, which has many applications in chemistry;

4. Generally describing Filter behavior.

Indeed, the primary work that needs to be accomplished is unifying the theory between filter and BCD methods. There are several interesting extensions as well. The first is investigating penalty parameter behavior. Most (if not all analysis) assumes that *rho* either stabilizes or trends to infinity to enforce feasibility. We have found that in some numerical experiments it is beneficial to actually *decrease rho*. While there is little theory behind penalty parameter choice for most functions (usually just lower bound constraints for problems with computable Hessians), it would be beneficial to see if the filter can handle such a scheme without entering a closed loop. Another mathematical exploration is describing filter behavior. Figure 6.5(f) shows how oddly the nonlinear NMF filter moves as it converges to a minimum. Such analysis on the Pareto-front of feasibility and first-order optimality has not been seriously studied (to the best of my knowledge), and warrants a thorough investigation.

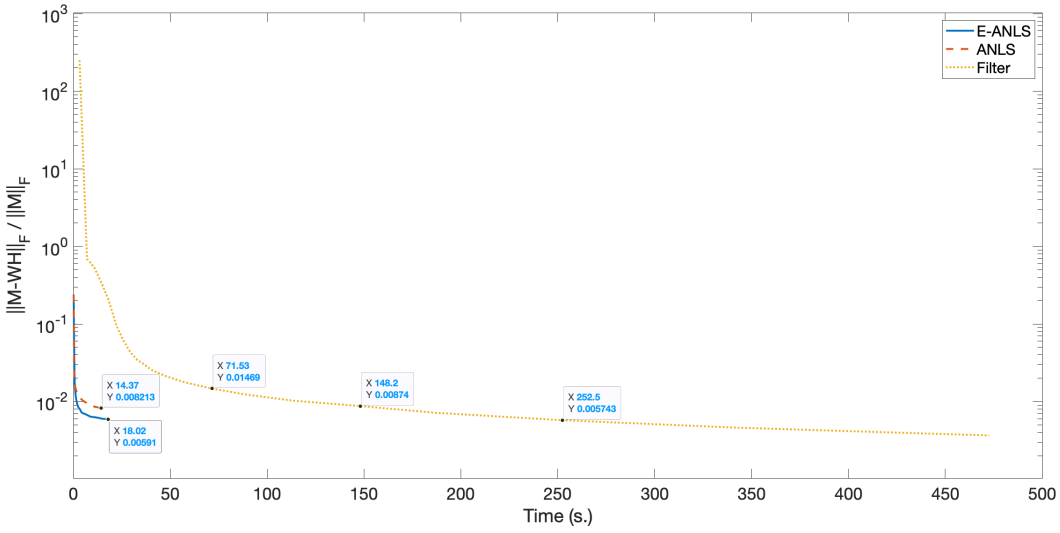
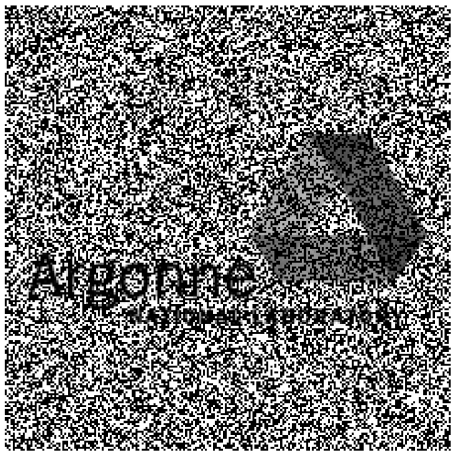


Figure 6.2: Comparison to ANLS and E-ANLS. We are much slower overall.

(a) True



(b) Data - black spots are excluded points

(c) XY

(d) Z

Figure 6.3: NMFC for ANL Image.

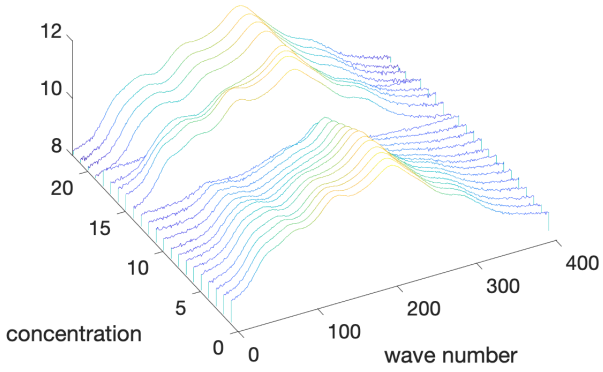
(a) True

(b) Data

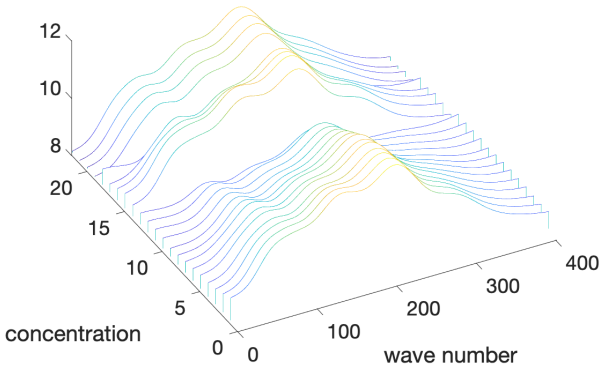
(c) XY

(d) Z

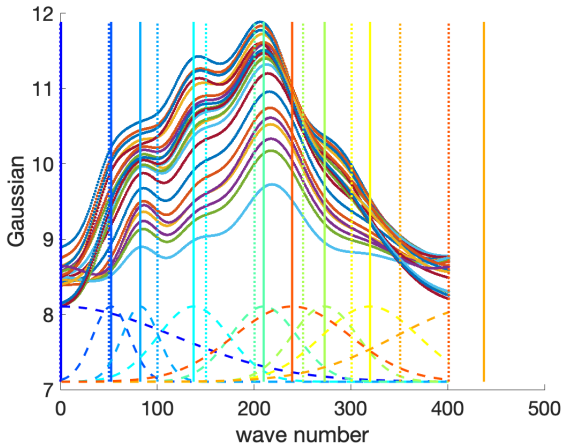
Figure 6.4: NMF for ANL Image.



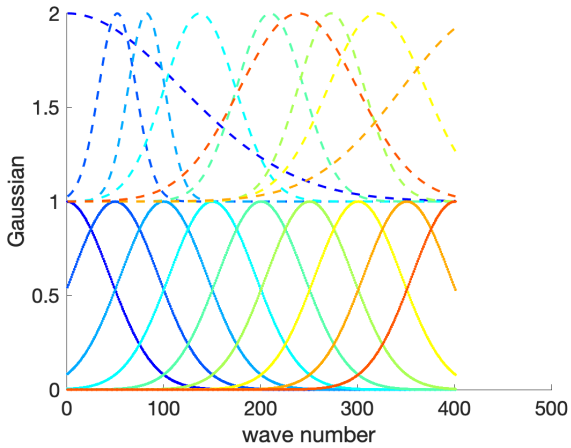
(a) Data



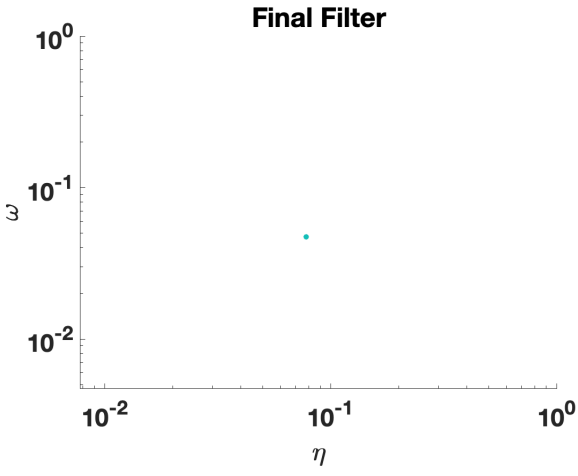
(b) XY



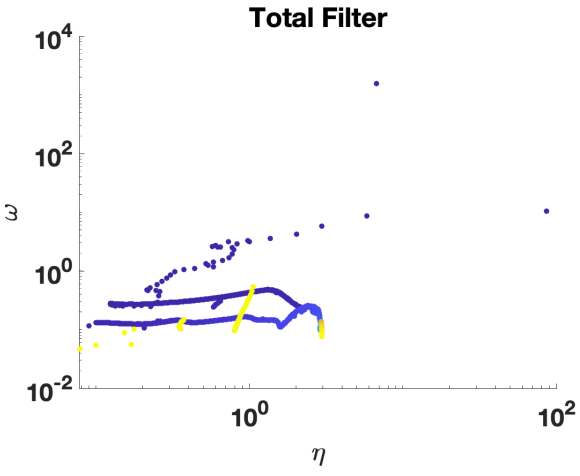
(c) Gaussian with Signal



(d) Prior/bottom vs posterior/top



(e) Final Filter



(f) Total Filter

Figure 6.5: NMF for Chemical Spectra.

Chapter 7

CONCLUSIONS, FUTURE WORK AND EXTENSIONS

7.1 Conclusions

The previous chapters have explored nonsmooth, nonlinear optimization problems with primarily the proximal gradient technique. The primary contribution is twofold: 1) an algorithm for a general class of basis pursuit denoise problems with nonsmooth constraints; 2) a fusion of trust region and proximal methods for nonlinear, nonsmooth regularized problems. These contributions were built on the fundamental properties of proximal operators and trust regions. These tools are the building blocks of many optimization routines, and the methods developed throughout this work have wide-ranging impacts for the larger inverse problem community.

7.1.1 Summary

In Chapter 3 of this work, we applied proximal gradient and splitting techniques to a novel regularized matrix factorization problem with applications to travel time tomography. We showed how local smoothness structure can be combined with low rank constraints using level-set optimization formulations, and develop a new relaxation algorithm that can efficiently solve these joint problems. In the seismology setting, we use the approach to interpolate missing stations and de-noise observed stations. The new approach is competitive with LBFGS and FISTA, and offers new functionality to interpolate observed data using both smoothness and low rank structure in the presence of data fitting constraints.

In Chapter 4, we extend the work in Chapter 3 to a more general set of problems that occur in basis pursuit denoise problems. We provide a general solver for level-set formulations, with applications in matrix completion and sparsity promotion in data interpolation and denoising. This is a new flexible algorithmic framework that targets *nonsmooth* level-set constraints, including ℓ_1, ℓ_∞, and even ℓ_0 norms. These constraints give greater flexibility for modeling deviations in observation and denoising, and have significant impact on the solution. Measuring error in the ℓ_1 and ℓ_0 norms makes the result more robust to large outliers, while matching many observations exactly. We demonstrated the approach for basis

pursuit denoise (BPDN) problems as well as for extensions of BPDN to matrix factorization, with applications to interpolation and denoising of 4D seismic data. The new methods are particularly promising for seismic applications, where the amplitude in the data varies significantly, and measurement noise in low-amplitude regions can wreak havoc for standard Gaussian error models.

In Chapter 5, we combine the nonsmooth analysis developed earlier with trust region methods. We develop a trust-region method for minimizing the sum of a smooth term f and a nonsmooth term h, both of which can be nonconvex. This is done via a trust region method for the smooth term f, and pushing the nonsmooth term into the subproblem where we solve it simultaneously with picking the descent direction. The model coincides with $f+h$ in value and subdifferential at the center. We establish global convergence to a first-order stationary point when f satisfies a smoothness condition that holds, in particular, when it has Lipschitz-continuous gradient. The model of h has minimal assumptions: it is required to be proper, lower-semi-continuous and prox-bounded. Under these weak assumptions, we establish a worst-case $O(1/\epsilon^2)$ iteration complexity bound that matches the best known complexity bound of standard trust-region methods for smooth optimization. We detail a special instance in which we use a limited-memory quasi-Newton model of f and compute a step with the proximal gradient method, resulting in a practical proximal quasi-Newton method. We establish similar convergence properties and complexity bound for a quadratic regularization variant, and provide an interpretation as a proximal gradient method with adaptive step size for nonconvex problems. We describe our Julia implementations and report numerical results on inverse problems from sparse optimization and signal processing. Our trust-region algorithm exhibits promising performance and compares favorably with linesearch proximal quasi-Newton methods based on convex models. Extensions of this algorithm include: barrier/penalty methods, inexact function evaluations, and nonlinear least squares.

Finally, in Chapter 6 we study minimization of separable but twice-continuous cost functions with continuous constraints. These may be nonconvex and also have simple bound

constraints, which can also be the sampling operator. The issue we attempt to address here is the convergence of ADMM with nonconvex problems; ADMM has been known to fail for this class and theory that shows convergence for certain function classes rests on *a priori* information about the function that may not be testable, especially for PDEs. Another condition is penalty parameter control; previous analysis usually give minimum/lower penalty parameter bounds, but these may be difficult to compute and control schemes are almost never provided. To solve these conditions, we turn to filter methods, which manage the penalty parameter to control the feasibility and first-order stationarity. Points are accepted into the filter if they satisfy certain criteria, and this eventually pushes the solution towards the most feasible solution that is also stationary. There is still much to be done; the first is unifying the convergence theory of block coordinate descent and filter methods, as well as better elucidating filter behavior for such nonlinear problems.

7.1.2 Extensions and Future work

Next, we extrapolate on basic problem statements for several future projects specifically involving nonsmooth inverse problems. The goal here was to just give the basic ideas for several projects I plan to work on during my tenure as a John von Neumann Postdoctoral Research Fellow at Sandia National Labs. Some of them are more developed than others, but the purpose is to establish the basic ideas. All of them continue the theme of proximal methods for nonsmooth problems. We later extend them into the PDE constrained optimization realm. The roadmap is as follows: first, we discuss inexact extensions of this thesis and Chapter 5 in particular and its application to HPC; next we talk about the nonlinear least squares extension of Chapter 5, and relate that to establishing convergence theory (or descent) for relax and split solvers in Algorithm 3; finally, we briefly discuss some connections with the PDE optimization realm and conclude with a brief retrospective.

7.2 Inexactness

A huge realm of possible expansion is inexact formulations for algorithms. Algorithms stand to benefit from computing architectures. Graphics processing units (GPUs) are growing ever more prominent in numerical computing [28]; GPUs are efficient at solving large linear systems, massively parallelizable problems, and at accelerating multi- and mixed-precision computing. Routines can improve efficiency by performing inexact computations distributed across processors with different speed and precision characteristics. Such low-precision computations are faster on specialized hardware and consume less energy.

My proposal for future work in my postdoc is to quantify and exploit *inexactness*, which stems from three identifiable characteristics: 1) cost function evaluation difficulties; 2) regularizer nonsmoothness; and 3) hardware. Future expansions will focus on inexact algorithm development for nonsmooth nonconvex inverse problems. The focus is to develop algorithmic settings that best exploit inexactness in composite inverse problems and capitalize on faster architectures to address the gap between traditional nonlinear approaches and the growing size of modern problems.

As computing clusters reach exascale capacity, the opportunity arises for optimization algorithms to exploit these architectures. The *motivation* is that mixed-precision computations combined with controlled inexactness can make nonsmooth, large inverse problems computationally tractable. This project is a synthesis of proximal relaxation techniques and mixed-precision numerics; the *key objective* is to show that inexact evaluations of relaxations converge to approximate stationary points at reasonable rates with run time and evaluation savings. We will develop analysis for convergence of inexact deterministic methods with numerical error.

Algorithms are classified according to *information* required for descent; first-order algorithms need gradient information, while second-order algorithms employ Hessian information in hopes to accelerate convergence and identify a weak minimizer instead of a first-order stationary point. Certain algorithms require only function values, and may attempt

to approximate gradient or Hessian information of f or c via numerical methods such as finite differencing, which can be inaccurate. There has long been an associated trade-off between numerical complexity and iteration cost; first-order methods are cheap and have low complexity, but convergence is slow for large problems. Even so, calculating full gradient information for f or c becomes prohibitively expensive in large scale problems and nonsensical for h [28, 125]. This issue arises in physical applications; electromagnetics problems [111], ML training [61], and finite element simulations of fluid-flow [4, 140] calculate derivatives for millions of parameters. Second-order methods are even more numerically intensive; computation and storage is not worth the cost for problems with even thousands of variables and constraints. Stochastic optimization methods employ partial gradient or Hessian information while ensuring sufficient accuracy in expectation; these can also exhibit slow convergence [45, 72]. Limited-memory quasi-Newton methods are a highly successful middle ground with storage and local convergence rate trade-offs [74, 123]. The approach we propose lies within this regime; we rely on *proximal splitting relaxations* of nonsmooth and nonconvex problems to control problem features and allow nonvanishing errors in the mixed-precision setting. The practical implementation creates a general framework for tackling computationally large inverse problems on clusters and software. This project ultimately has three goals outlined below: two theoretical and one practical.

7.2.1 Inexact Trust Region Review

Often, applications will have extremely expensive or incomputable cost functions ϕ or gradients $\nabla\phi$. A host of work [28, 39, 40, 102, 106, 123] has been done to allow for inexact gradient or function evaluations. These inexact gradient evaluations can be used in the trust region subproblem, while inexact function evaluations can be allowed in the ρ_k computation. [102] (and described in [40]) introduce an inexact gradient condition that requires the gradient approximation at the trust region center to asymptotically approach the true gradient. This is given by the relationship $\|\nabla\phi(x_k) - g_k\| \to 0$ for a convergence sequence of x_k and g_k being the inexact gradient. [35] eludicate relative gradient error condition $\|\nabla\phi(x_k) - g_k\| \leq \eta\|g_k\|$

for $\eta \in (0,1)$. This conditions implies an accuracy on g_k at a particular iteration, but does not require g_k to be recomputed to a higher accuracy after a failed step. However, this does need a tight bound on the gradient error, which can be impractical. [135] uses the gradient condition $\|\nabla\phi(x_k) - g_k\| \leq \min\{\kappa_1\Delta_k, \kappa_2\}$ with $\kappa_1, \kappa_2 > 0$. This requires increased accuracy as Δ_k decreases (after rejected steps), but relies on arbitrary constants. In the sequential quadratic programming context, [65] suggest $\|\nabla\phi(x_k) - g_k\| \leq \xi \min\{\|g_k\|, \Delta_k\}$ for $\xi > 0$. The condition increases accuracy after failed iterations or near convergence. The arbitrary constants above permit the use of error indicators that can circumvent the use of tight bounds. This allows for the derivation of a function $E(x_k)$ such that $\|\nabla\phi(x_k) - g_k\| \leq \xi E(x_k)$; then [65] can satisfy $E(x_k) \leq \kappa \min\{\|g_k\|, \Delta_k\}$ for $\kappa > 0$ being any user-defined constant. Hence, xi is never fortunately computed, as it may depend on unknown/incomputable quantities such as Lipschitz constants or PDE bounds.

Similar tricks can be performed with function evaluations [35, 78]. [78] use an asymptotic condition as in [65]. This replaces the ρ_k computation with

$$\tilde{\rho}_k = \frac{\varphi(x_k) - \varphi(x_k + s_k)}{m_k(0) - m_k(s_k)}$$

for $\varphi : \mathbb{R}^n \to \mathbb{R}$ is the inexact objective model that satisfies

$$|\phi(x_k) - \phi(x_k + s_k) + \varphi(x_k + s_k) - \varphi(x_k)| \leq \kappa[\eta \min\{m_k(0) - m_k(s), r_k\}]^{1/\omega}$$

for $\sigma > 0$ and $\{r_k\}_{k=1}^{\infty} \to 0$ a forcing sequence. We can therefore use another error indicator

$$|\phi(x_k) - \phi(x_k + s) + \varphi(x_k + s) - \varphi(x_k)| \leq \sigma E_k(x)$$

where the true error can be disregarded and the inexact objective condition enforced solely based on the metric

$$E_k(x_k + s_k)^{\omega} \leq \eta \min\{m_k(0) - m_k(s), r_k\}$$

for algorithmic constants ω, η. [78] and [65] combined allow for global convergence without ϕ or $\nabla\phi$ queries.

7.2.2 Nonsmooth Relaxations:

Here, we seek to develop a general framework for inexact problem relaxations of (1.1). Trust-region methods exist as numerically efficient approximations of nonlinear functions [40] and are used frequently in the community ([4] and works therein). Each k iteration uses a surrogate quadratic model of smooth f in (1.1) with gradient ∇f and Hessian approximation $B_k = B_k^T$ that is valid within a region determined by quadratic model performance and accuracy [40]. Evaluating this quadratic model saves on cost when the true model is large and forward solutions are expensive. Problem features such as nonsmoothness or nonconvexity complicate derivative information, often requiring specialized first-order methods. Nonsmooth trust-region theory exists [40, 93] but is restrictive. In [14]/Chapter 5, we develop a proximal trust-region method for composite problems to handle nonsmoothness for nonlinear cost functions with no constraints. We address nonsmooth h with the Moreau proximal envelope (2.1a). Each iteration of (2.5) requires evaluating (2.1b), which in turn requires solving a nonsmooth optimization problem. Past work Chapter 5 elucidates the global convergence theory of trust-region proximal relaxations for unconstrained nonsmooth, nonconvex problems [14] under the assumption of exact function values and gradients. A trust-region iteration aims to decrease a model of $f+h$ inside a region centered at the current iterate. Chapter 5 uses iteration (2.5) to achieve such decrease.

Relax Assumptions on Hessian Approximation $\boldsymbol{B_k}$.

Current work Chapter 5 assumes bounded $\{B_k\}$; this can be relaxed to $\sum \|B_k\|^{-1} = \infty$ to obtain existence of a first-order critical limit point of $\{x_k\}$. We use a quasi-Newton approximation B_k in the quadratic model of f when Hessian information is inaccessible or too costly to obtain. The sequence $\{B_k\}$ is assumed to be bounded in trust-region convergence theory, which is problematic in quasi-Newton methods but can be relaxed in the smooth case [40]. This can also extend to Gauss-Newton Hessians for nonlinear least squares ML problems.

Accelerated Proximal Gradient Techniques.

Extend [14] to accelerated variants of proximal gradient [18]. Chapter 5 relies on plain proximal gradient, but accelerated variants have improved complexity bounds and typically perform better in practice [18]. We hope to establish that accelerated variants satisfy the sufficient decrease conditions stated in Chapter 5/[14] to guarantee convergence of the trust-region method.

Constrained Problems.

Extend our formulation in Chapter 5 to Penalty Methods. This was already touched on in Chapter 6, but Chapter 5 formulation readily extends to nonconvex constraints. Penalty methods are a popular approach for (1.1); examples include PDE-constrained problems [4], nonlinear functions [60], and others. They solve a sequence of unconstrained problems of the form

$$\text{minimize} \;\; f(x) + h(x) + \delta P(c(x)),$$

where $\delta > 0$ is a penalty parameter and P is a penalty function; for instance $P(y) = \|y\|_p$, an ℓ_p-norm. My framework in Chapter 5 extends readily to penalty methods under the assumption of exact function values and derivatives. Typically this is simpler than (1.1), except in cases where the value of (2.1b) is known. An example is $h(x) = \|c(x)\|_1$; a trust-region method would then require computation of the prox (2.1b) of $\|c(x_k) + \nabla c(x_k)^T s\|_1$ over descent direction s, which is known analytically. This is an extension of nonsmooth trust-region theory in [40].

7.2.3 Numerical Speed-ups

Here, we seek to provide theoretical error bounds on function and derivative information needed for convergence. This goal addresses numerical efficiency in function evaluations and derivative computation via controlled approximations. This is key for functions that are too computationally intense to evaluate. Example applications include computational fluid

dynamics, climate modeling, or earth systems; these simulations can be expensive with large numbers of parameters. Some optimization methods are resilient to certain degrees of inexactness. Newton methods [128] allows the Newton's search direction computation to be truncated while retaining favorable convergence. Such freedom is useful for directions computed via an iterative process, such as a Krylov method. Certain methods for smooth optimization also permit inexactness in objective and constraint evaluation and their derivatives while preserving convergence properties [40]. A natural extension of Goal 1 is to permit inexact function values and derivatives as in [40], and inexact computation of the proximal term (2.1b) as in [93, 119, 123, 125, 147]. A crucial component of the theory assumed in Goal 1 is that (2.1b) must be computable exactly, which is not always known; therefore, an important objective is extension into cases where the prox (2.1b) must be approximated via an iterative process. This establishes algorithm error tolerance in derivative computations, and provides theoretical understanding of functions without closed-form proximal operators.

Function and Derivative Inexactness.

I would like to generalize and control error handling in Algorithm 16 and also Algorithm 18, with the formulations below mostly geared towards the former. I allow for approximations $\tilde{f}_x \approx f(x)$, $\tilde{c}_x \approx c(x)$, $\tilde{g}_x \approx \nabla f(x)$, $\tilde{J}_x \approx \nabla c(x)$,

$$|f(x) - \tilde{f}_x| \leq \varepsilon_f, \quad \|c(x) - \tilde{c}_x\| \leq \varepsilon_c, \quad \|\nabla f(x) - \tilde{g}_x\| \leq \varepsilon_{\nabla f}, \quad \|\nabla c(x) - \tilde{J}_x\| \leq \varepsilon_{\nabla c},$$

for appropriate iteration-dependent tolerances ε_f, ε_c, $\varepsilon_{\nabla f}$ and $\varepsilon_{\nabla c} > 0$ controlled by the algorithm. Such approximations could result from evaluations in low floating-point precision. I also will generalize inexactness to nonsmooth h to preserve convergence and complexity; i.e. conditions on $\tilde{h}_x \approx h(x)$ and $\tilde{v} \in D_x \approx \partial h(x)$, where ∂h is an appropriate subdifferential.

Smooth Error Tolerances. Trust-region methods for smooth unconstrained optimization can control error tolerances so as to preserve convergence [40]. Such control can apply to the smooth term in trust-region proximal splitting algorithm Chapter 4. Smooth $f(x)$, $c(x)$

and $\nabla f(x), \nabla c(x)$, as well as search directions, can be inexact.

Nonsmooth Error Tolerances. Allow inexact solution of (2.1b) for difficult h. Each iteration requires solving an optimization subproblem involving the nonsmooth term; this is easier than solving the original problem as f and h are replaced with simpler models. In the literature, (2.1b) is almost always assumed to be solved exactly, with few exceptions [93, 125]. In practice, it seems more efficient to stop short of optimality and update

$$x_{k+1} \leftarrow \operatorname*{prox}_{\nu h}\left(x_k - \nu^{-1}\tilde{g}_{x_k}\right) + e_k, \tag{7.1}$$

where $\tilde{g}_{x_k} \approx \nabla f(x_k)$ and $e_k > 0$ measures the error in the evaluation of (2.1b). Such error is considered in [93] in a related context. As in Goal 1, the above framework can be extended to include second-order information [14, 123] and accelerated methods [125].

Iteration Cost.

Early iterations need not be accurate when far from a solution. The inexactness error in function values is updated along the iterations and depends on model decrease. As iterations progress and convergence occurs, higher accuracy is required; this can greatly benefit situations that can change between coarse or fine forward solutions, such as PDE-constrained optimization [25, 79], and peridynamics [140]. In contrast, gradient evaluations need not increase in accuracy during convergence, permitting low-accuracy derivatives. Gradient approximations are required to satisfy a condition such as the *norm test*, which requires

$$\|\nabla f(x) - \tilde{g}_x\| \leq \omega \|\tilde{g}_x\|, \quad 0 < \omega < 1.$$

Geometrically, the norm test requires that the gradient approximation be directions of sufficient descent to preserve convergence properties, following analysis performed in [10, 27, 40].

7.2.4 Software and GPU consolidation

I would like to generalize my algorithms and release software packages complete with implementation on large problems that take advantage of efficient hardware. My work in Chapter 5

lends itself to large scale, nonlinear inverse problems. Efficiency is key when cost function evaluations may be prohibitively expensive, and such problems can benefit from speed-ups on mixed-precision machines. This provides an additional source of inexactness, as GPUs move beyond the traditional IEEE 754 single and double precision to half-precision units. More examples are the BFloat16 format found on Google's tensor processing units, the quadruple precision format, posits, or qubits. As in Goal 2, early iterations can possess looser accuracy, allowing for half- or even more lenient precision. A major challenge when working in half or lower precision is scaling the problem to avoid under or overflow [70]. The framework that I propose can readily handle such scaling. For instance, evaluating $f(x)$ in single precision really produces $\tilde{f}(x)$, which satisfies

$$|f(x) - \tilde{f}(x)| \leq \epsilon_{32}|f(x)|,$$

where ϵ_{32} is the machine epsilon of single precision. For $|f(x)| \leq M$ scaled in the domain of interest, evaluations of f need only be accurate to within $10^{-7}M$ approximately. A similar reasoning holds for half and double precision, as well as other numerical representations. These errors accommodate mixed-precision arithmetic, which improves memory bandwidth pressure and increase numerical throughput. This framework also lends itself to coordinate-wise decomposable proximal operators, leading to massive parallelization on GPUs. Common decomposable $h(x)$ in (2.1b) include soft-thresholding for $h(x) = \|\cdot\|_1$, hard-thresholding for $h(x) = \|\cdot\|_0$, and indicator functions like the nonnegative orthant or box constraints.

I plan to release open-source, flexible software packages for use in optimization and the large scientific community. The algorithm for [14] is written in Julia; I plan to extend this to the CUDA environment and C++ for use on large clusters.

7.2.5 Impact on High Performance Computing

Increases in computational power and data availability have massively impacted optimization, and subsequent accessibility further increases the scientific and societal impacts. Next phases of this work involve developing and improving inexact algorithms for high-performance,

composite mixed-precision computing. Relaxations have been employed for nonlinear [40] and composite functions [123, 125], but this has not been explored in GPU settings or with nonsmoothness and nonconvexity in (1.1). Goal 1 seeks to combine such relaxations for nonsmooth problems. By developing Goal 2, we address nonconvexity so often found in the literature [10, 27, 116, 131] and extend it to inexact computations. Goal 3 seeks to account for nonlinear and nonsmooth problem structure in GPU settings, and apply it to inverse problems and numerical software.

There are a plethora of inverse problem and numerical computing applications that stand to benefit from this proposal, including optimization problems in quantum computing, peridynamics [140], uncertainty quantification [1, 101], fluid dynamics [25, 111], training ML models [61, 110], PDE-constrained optimization [79], and mechanical engineering [23]. This work could produce numerical software which can compete with or be added to packages such as Dakota, Trilinos, and Kokkos. Current work focuses on scalable optimization, multigrid/multilevel methods, and finite element solvers for PDEs. Optimization for inexact research is geared towards schemes for solving PDEs, whereas some more standard, out of the box packages focus on numerical cost and black-box optimization. My goal is to supplement this research by providing a fast framework for parameter estimation by using certain features of these solvers and computing resources. Much inexact research addreses solving PDE with finite elements and inverse problems for physical systems [25, 79, 101]. This proposal's framework addresses error of such solvers in the context of parameter estimation with difficult regularization terms, or even provide theory for solver error complexity. For instance, Dakota or Trilinos fidelity can be tailored for forward solutions of nonlinear inverse problems. Low-accuracy evaluations could take the form of early coarser-grid discretization, and higher accuracy grid refinement as parameter estimation progresses. These libraries, in addition to Kokkos, already allow for GPU evaluations, and hence are a natural testing ground for my method. This proposal supplements work in PDE-constrained optimization [25, 79] by providing theory for inexact evaluations applied to numerical multifidelity. It can also guide precision characteristics for mesh discretization in Dakota, as well as forward solves

for complicated models in the larger scientific community, such as the seismology or power grid simulations. Providing theory on problem relaxation and function/derivative error will create a ready reference to computational trade-offs optimizers and scientists make daily. An immediate impact is a greater theoretical understanding of numerical error and problem relaxations in composite regimes. A practical impact is that generalizability benefits hosts of data-intensive biological and physical applications [28, 131]. It designs a flexible algorithm and software for next-generation HPC platforms, and allows researchers to conduct parameter estimation without compromising speed. This is accomplished by creating a time- and energy-efficient optimization framework for the exascale era that accounts for errors and uncertainty. Its generalizability benefits the host of data-intensive applications in physics, machine-learning, and mathematics.

7.3 Nonlinear Least Squares

This section elucidates some preliminary results with extending our formulation in Chapter 5 to nonlinear least squares (NLS) problems. This has numerous applications to inverse problems [131] and the greater machine-learning community, which often use NLS-type formulations for minimization. The distinction here is the use of the Gauss-Newton Hessian approximation; we assume $f(x) = \|F(x)\|_2^2$, where we can compute the Jacobian of $F(x)$ and use that in our minimization routine. This can get faster convergence for some types of inverse problems [88], but has yet to be generalized to the nonsmooth regularized context.

7.3.1 Problem Introduction

We consider the problem

$$\underset{x}{\text{minimize}} \;\; f(x) + h(x), \qquad f(x) = \tfrac{1}{2}\|F(x)\|_2^2, \tag{7.2}$$

where $F : \mathbb{R}^n \to \mathbb{R}^m$ is continuously differentiable and $h : \mathbb{R}^n \to \mathbb{R}$ is proper and lower semi-continuous, but may be nonsmooth and nonconvex. We use $J(x)$ to denote the Jacobian of F at x. We again use the limiting subdifferential Definition 5 and Proposition 2.2.3 In particular, our stationary point is

$$x \text{ is first-order stationary for (7.2)} \quad \Longleftrightarrow \quad 0 \in J(x)^T F(x) + \partial h(x). \tag{7.3}$$

7.3.2 Linear Least Squares

Here, we describe how the models change from Chapter 5 from general nonsmooth regularized problems to linear least squares regularized problems. Note that here, we have no trust region yet. This approach is analogous to the quadratic regularization in section 5.3. For

fixed $x \in \mathbb{R}^n$, define

$$\varphi(s;x) := \tfrac{1}{2}\|J(x)s + F(x)\|_2^2 \tag{7.4a}$$

$$\psi(s;x) \approx h(x+s) \tag{7.4b}$$

$$m(s;x,\sigma) := \varphi(s;x) + \tfrac{1}{2}\sigma\|s\|^2 + \psi(s;x), \tag{7.4c}$$

where $\sigma \geq 0$. Consider the parametric problem and its optimal set

$$p(\sigma;x) := \min_s \; m(s;x,\sigma) \leq \varphi(0;x) + \psi(0;x) = f(x) + h(x) \tag{7.5a}$$

$$P(\sigma;x) := \arg\min_s \; m(s;x,\sigma), \tag{7.5b}$$

The form of (7.5) is representative of a Levenberg-Marquardt subproblem for (7.2) in which f and h are modeled separately.

We make the following additional assumption that ψ is proper, lsc, and level bounded (Model Assumption 5.2.1). By Proposition 2.2.3,

$$s \in P(\sigma;x) \quad \Longrightarrow \quad 0 \in \nabla\varphi(s;x) + \sigma s + \partial\psi(s;x).$$

We define

$$\xi(\sigma;x) := (f+h)(x) - p(\sigma;x). \tag{7.6}$$

The following stationarity criterion follows directly from the definitions above.

Lemma 7.3.1. *Let Model Assumption 5.2.1 be satisfied and $\sigma > 0$. Then $\xi(\sigma;x) = 0 \iff 0 \in P(\sigma;x) \implies x$ is first-order stationary for (7.2). In addition, x is first-order stationary for (7.2) if and only if $s = 0$ is first-order stationary for (7.4c).*

Proof. Note first that $\xi(\sigma;x) = 0 \iff p(\sigma;x) = (f+h)(x) = \varphi(0;x)+\psi(0;x)$, which occurs if and only if $0 \in P(\sigma;x)$. Proposition 2.2.3 then implies $0 \in \partial m(0;x,\sigma) = \nabla\varphi(0;x)+\partial\psi(0;x)$ and is equivalent to (7.3). □

The next result states some properties of (7.5) similar to those of Proposition 5.2.1.

Proposition 7.3.2. *Let Model Assumption 5.2.1 be satisfied. The domain of $p(\cdot;x)$ and $P(\cdot;x)$ is $\{\sigma \mid \sigma \geq 0\}$. In addition,*

1. *$p(\cdot;x)$ is proper lsc and for each $\sigma \geq 0$, $P(\sigma;x)$ is nonempty and compact;*

2. *if $\{\sigma_k\} \to \bar{\sigma} \geq 0$ in such a way that $\{p(\sigma_k;x)\} \to p(\bar{\sigma};x)$, and for each k, $s_k \in P(\sigma_k;x)$, then $\{s_k\}$ is bounded and all its limit points are in $P(\bar{\sigma};x)$;*

3. *if $\varphi(\cdot;x) + \psi(\cdot;x)$ is strictly convex, $P(\sigma;x)$ is single-valued;*

4. *$p(\cdot;x)$ is continuous at any $\bar{\sigma} > 0$ and $\{p(\sigma_k;x)\} \to p(\bar{\sigma};x)$ holds in part 2 if $\bar{\sigma} > 0$.*

Proof. Parts 1–2 follow from applying [120, Theorem 1.17] by noting that (7.4c) is level-bounded in s locally uniformly in σ because $\psi(\cdot;x)$ is level bounded. Part 4 also follows from [120, Theorem 1.17] by noting that (7.4c) is continuous in σ at any $\bar{\sigma} > 0$. □

By Proposition 7.3.2 part 4, $\xi(\cdot;x)$ is continuous at any $\bar{\sigma} > 0$.

The first step of the proximal gradient method applied to (7.4c) from $s_0 = 0$ computes

$$s_1 \in \operatorname*{prox}_{\nu\psi(\cdot;x)}(-\nu J(x)^T F(x)),$$

where $0 < \nu < 1/(\sigma_{\max}(J)^2 + \sigma) < 1/\sigma$ is a step length. The decrease achieved by this first step is

$$(f+h)(x) - (\varphi(s_1;x) + \psi(s_1;x)) \geq \tfrac{1}{2}(\nu^{-1} - \sigma)\|s_1\|^2 \geq \tfrac{1}{2}\nu^{-1}\|s_1\|^2,$$

and therefore,

$$\xi(\sigma;x) \geq \tfrac{1}{2}\sigma\|s_1\|^2. \tag{7.7}$$

The algorithm in this part will be similar to Algorithm 17, but with a linear least squares adaptation. It remains to be developed in full; here we just elucidate some preliminary theory and move on the nonlinear least squares example.

7.3.3 Nonlinear Least Squares

A trust-region approach

We first briefly comment on the applicability of Algorithm 16 to (7.2). We assume that each $f_i : \mathbb{R}^n \to \mathbb{R}$ is $\mathcal{C}^1$, so that we satisfy Step Assumption 5.2.1 in Chapter 5. A natural model for f about x is the Gauss-Newton model (7.4a), which satisfies $\varphi(0; x) = f(x)$ and $\nabla_s \varphi(0; x) = \nabla f(x) = J(x)^T F(x)$. In Algorithm 16, the first proximal gradient step s_1 is computed by solving

$$\underset{s}{\text{minimize}} \ \tfrac{1}{2}\|F(x)\|_2^2 + \left(J(x)^T F(x)\right)^T s + \tfrac{1}{2}\nu^{-1}\|s\|^2 + \psi(s; x) \qquad \text{subject to } \|s\| \leq \Delta,$$

i.e.,

$$s_1 \in \underset{\nu\psi(\cdot;x)+\chi(\cdot;\Delta)}{\text{prox}} (-\nu J(x)^T F(x)),$$

where $0 < \nu < 1/\|J(x)\|^2$. Subsequent steps continue the proximal gradient iterations to compute an approximate solution of

$$\underset{s}{\text{minimize}} \ \tfrac{1}{2}\|J(x)s + F(x)\|_2^2 + \psi(s; x) \qquad \text{subject to } \|s\| \leq \min(\beta\|s_1\|, \Delta), \tag{7.8}$$

where $\beta \geq 1$, which is a trust-region variant of the method of [88] and [98] for nonconvex regularized least-squares problems. Step Assumption 5.2.1 holds provided ∇f is Lipschitz continuous or each f_i is $\mathcal{C}^2$ with bounded Hessian, and their convergence results and iteration complexity bound hold.

7.3.4 Levenberg-Marquardt

We now examine the alternative implementation of the method of Levenberg and Marquardt in which the model (7.4c) is employed to compute a step. Specifically, consider Algorithm 24.

Our main working assumption is the following.

Problem Assumption 7.3.1. *The residual F is $\mathcal{C}^1$ on the level set $\Omega := \{x \in \mathbb{R}^n \mid f(x) \leq f(x_0)\}$.*

Algorithm 24 Nonsmooth regularized Levenberg-Marquardt Method.

1: Choose constants $0 < \eta_1 \leq \eta_2 < 1$ and $0 < \gamma_3 \leq 1 < \gamma_1 \leq \gamma_2$.

2: Choose $x_0 \in \mathbb{R}^n$ where h is finite, $\sigma_0 > 0$, compute $F(x_0)$ and $h(x_0)$.

3: **for** $k = 0, 1, \ldots$ **do**

4: Define $m(s; x_k, \sigma_k)$ as in (7.4c).

5: Compute an approximate solution s_k of (7.5b).

6: Compute the ratio

$$\rho_k := \frac{f(x_k) + h(x_k) - (f(x_k + s_k) + h(x_k + s_k))}{\varphi(0; x_k) + \psi(0; x_k) - (\varphi(s_k; x_k) + \psi(s_k; x_k))}.$$

7: If $\rho_k \geq \eta_1$, set $x_{k+1} = x_k + s_k$. Otherwise, set $x_{k+1} = x_k$.

8: Update the regularization parameter according to

$$\sigma_{k+1} \in \begin{cases} [\gamma_3 \sigma_k,\ \sigma_k] & \text{if } \rho_k \geq \eta_2, \\ [\sigma_k,\ \gamma_1 \sigma_k] & \text{if } \eta_1 \leq \rho_k < \eta_2, \\ [\gamma_1 \sigma_k,\ \gamma_2 \sigma_k] & \text{if } \rho_k < \eta_1. \end{cases}$$

An auxiliary assumption that is slightly stronger but likely to be required for our assumptions on the model to be satisfied is the following.

Problem Assumption 7.3.2. *The residual F and its Jacobian J are Lipschitz continuous on Ω.*

Under Problem Assumption 7.3.2, it is not difficult to show that ∇f is Lipschitz continuous on Ω, i.e., there exists $L > 0$ such that

$$f(x+s) \leq f(x) + \nabla f(x)^T s + \tfrac{1}{2} L \|s\|^2 \quad \text{for all } x,\ x+s \in \Omega. \tag{7.9}$$

We emphasize that in what follows, knowledge of L, or an estimate thereof, is not required. Our next assumption on the model is the following.

Model Assumption 7.3.1. *There exists a constant $\kappa_{\mathrm{m}} > 0$ such that for all x and $s \in \mathbb{R}^n$, $|(f+h)(x+s) - (\varphi+\psi)(s;x)| \leq \kappa_{\mathrm{m}} \|s\|^2$.*

Model Assumption 7.3.1 is essentially an assumption on the nonsmooth part ψ of the model. Indeed, (7.4a) and (7.9) combine to yield

$$\begin{aligned} f(x+s) - \varphi(s;x) &\leq f(x) + \nabla f(x)^T s + \tfrac{1}{2} L \|s\|^2 - \tfrac{1}{2} \|F(x) + J(x)s\|^2 \\ &= \tfrac{1}{2} L \|s\|^2 - \tfrac{1}{2} \|J(x)s\|^2 \\ &\leq \tfrac{1}{2} L \|s\|^2, \end{aligned}$$

where we used the definition of $f(x)$ and the identity $\nabla f(x) = J(x)^T F(x)$. In particular, Model Assumption 7.3.1 is satisfied with $\kappa_{\mathrm{m}} = L$ if we select $\psi(s;x) := h(x+s)$.

For simplicity, we use the shorthands $F_k := F(x_k)$ and $J_k := J(x_k)$. Our first result ensures that σ_k is bounded above in Algorithm 24, and follows from Theorem 5.5.2.

Theorem 7.3.3. *Let Problem Assumption 7.3.1 and Model Assumptions 5.2.1 and 7.3.1 be satisfied, and let*

$$\sigma_{\mathrm{succ}} := 2\kappa_{\mathrm{m}} / (1 - \eta_2) > 0. \tag{7.10}$$

If x_k is not first-order stationary and $\sigma_k \geq \sigma_{\mathrm{succ}}$, then iteration k is very successful and $\sigma_{k+1} \leq \sigma_k$.

Proof. Let s_k be the step computed at iteration k of Algorithm 24. Because x_k is not first-order stationary, $s_k \neq 0$. Because s_k is an approximate solution of (7.5b), we must have

$$\begin{aligned}\varphi(0; x_k) + \psi(0; x_k) &\geq \varphi(s_k; x_k) + \tfrac{1}{2}\sigma_k\|s_k\|^2 + \psi(s_k; x_k) \\ &= \tfrac{1}{2}\|F_k\|^2 + (J_k^T F_k)^T s_k + \tfrac{1}{2}\|J_k s_k\|^2 + \tfrac{1}{2}\sigma_k\|s_k\|^2 + \psi(s_k; x_k),\end{aligned}$$

and therefore,

$$\psi(0; x_k) - \psi(s_k; x_k) \geq (J_k^T F_k)^T s_k + \tfrac{1}{2}\|J_k s_k\|^2 + \tfrac{1}{2}\sigma_k\|s_k\|^2. \tag{7.11}$$

In addition,

$$\varphi(0; x_k) - \varphi(s_k; x_k) = -(J_k^T F_k)^T s_k - \tfrac{1}{2}\|J_k s_k\|^2. \tag{7.12}$$

We combine (7.11) and (7.12) into

$$\varphi(0; x_k) + \psi(0; x_k) - (\varphi(s_k; x_k) + \psi(s_k; x_k)) \geq \tfrac{1}{2}\sigma_k\|s_k\|^2. \tag{7.13}$$

Model Assumption 7.3.1 and (7.13) combine to yield

$$|\rho_k - 1| = \frac{|f(x_k + s_k) + h(x_k + s_k) - (\varphi(s_k; x_k) + \psi(s_k; x_k))|}{\varphi(0; x_k) + \psi(0; x_k) - (\varphi(s_k; x_k) + \psi(s_k; x_k))} \leq \frac{2\kappa_{\mathrm{m}}\|s_k\|^2}{\sigma_k\|s_k\|^2}.$$

After simplifying by $\|s_k\|^2$, we obtain $\sigma_k \geq \sigma_{\mathrm{succ}} \implies \rho_k \geq \eta_2$. □

Note that Theorem 7.3.3 does not explicitly include Problem Assumption 7.3.2 in its assumptions, though it is likely to be required for Model Assumption 7.3.1 to hold. Interestingly, Theorem 7.3.3 holds without assuming that the step s_k satisfies a sufficient decrease condition. Upon examination of the proof, the reason turns out to be that any step that results in simple decrease in $m(s; \sigma, x)$ results in sufficient decrease in $\varphi(\cdot; x) + \psi(\cdot; x)$, independently of the method used to compute s_k.

Theorem 7.3.3 ensures existence of a constant $\sigma_{\max} > 0$ such that

$$\sigma_k \leq \sigma_{\max} := \min(\sigma_0, \gamma_2\sigma_{\mathrm{succ}}) > 0 \quad \text{for all } k \in \mathbb{N}. \tag{7.14}$$

Our first result concerns the situation where a finite number of successful iterations occur. The proof is almost identical to that of [40, Theorem 6.4.4] and Theorem 5.2.5 and is omitted.

Theorem 7.3.4. *Let Problem Assumption 7.3.1 and Model Assumptions 5.2.1 and 7.3.1 be satisfied. If Algorithm 24 only generates finitely many successful iterations, then $x_k = x^*$ for all sufficiently large k and x^* is first-order critical.*

Because $p(\sigma; x)$ increases when σ increases, $\xi(\sigma; x)$ decreases when σ increases. Thus, it follows from (5.45) that

$$\xi(\sigma_k; x_k) \geq \xi(\sigma_{\max}; x_k) \quad \text{for all } k \in \mathbb{N}. \tag{7.15}$$

Lemma 7.3.1 and (7.15) suggest using $\xi(\sigma_{\max}; x_k)^{\frac{1}{2}}$ as stationarity measure.

For a stopping tolerance $\epsilon \in (0, 1)$, we seek to determine $k(\epsilon) \in \mathbb{N}$ such that

$$\xi(\sigma_{\max}; x_k)^{\frac{1}{2}} > \epsilon \quad \text{for all } k > k(\epsilon) \quad \text{and} \quad \xi(\sigma_{\max}; x_{k(\epsilon)})^{\frac{1}{2}} \leq \epsilon. \tag{7.16}$$

Define the sets similar to (5.15)

$$\mathcal{S} := \{k \in \mathbb{N} \mid \rho_k \geq \eta_1\}, \tag{7.17a}$$

$$\mathcal{S}(\epsilon) := \{k \in \mathcal{S} \mid k < k(\epsilon)\}, \tag{7.17b}$$

$$\mathcal{U}(\epsilon) := \{k \in \mathbb{N} \mid k \notin \mathcal{S} \text{ and } k < k(\epsilon)\}. \tag{7.17c}$$

In order to conduct the complexity analysis, it is necessary to assume that the step computation at stage 5 of Algorithm 24 is related to $\xi(\sigma_k; x_k)$. We make the following assumption.

Step Assumption 7.3.1. *There exists $\kappa_{\text{mdc}} \in (0, 1)$ such that s_k computed at stage 5 of Algorithm 24 satisfies*

$$\varphi(0; x_k) + \psi(0; x_k) - (\varphi(s_k; x_k) + \psi(s_k; x_k)) \geq \kappa_{\text{mdc}} \xi(\sigma_k; x_k). \tag{7.18}$$

Step Assumption 7.3.1 is similar to sufficient decrease conditions used in trust-region methods—see [40]. Chapter 5 provide a concrete use of such condition in a trust-region method for nonsmooth regularized optimization. Clearly, the sufficient decrease assumption is satisfied after a single step of the proximal gradient method applied to (7.4c). In view

of (7.7), it is also satisfied at a minimizer of (7.4c). Thus, in step 5 of Algorithm 24, one strategy is to continue the proximal-gradient iterations until a stopping condition is attained. The following results parallel those of Chapter 5, which are in turn inspired from those of [36] and references therein.

Lemma 7.3.5. *Let Problem Assumption 7.3.1 and Model Assumptions 5.2.1 and 7.3.1 be satisfied and s_k be computed according to Step Assumption 7.3.1. Assume there are infinitely many successful iterations and that $f(x) + h(x) \geq (f + h)_{\text{low}}$ for all $x \in \mathbb{R}^n$. Then, for all $\epsilon \in (0, 1)$,*

$$|\mathcal{S}(\epsilon)| \leq \frac{(f + h)(x_0) - (f + h)_{\text{low}}}{\eta_1 \kappa_{\text{mdc}} \epsilon^2} = O(\epsilon^{-2}). \tag{7.19}$$

Proof. The proof is identical to that of Lemma 5.2.6 in Chapter 5. □

Lemma 7.3.6. *Under the assumptions of Lemma 7.3.5,*

$$|\mathcal{U}(\epsilon)| \leq \frac{\log(\sigma_{\max}/\sigma_0)}{\log(\gamma_1)} + |\mathcal{S}(\epsilon)| \frac{|\log(\gamma_3)|}{\log(\gamma_1)} = O(\epsilon^{-2}). \tag{7.20}$$

Proof. For each $k \in \mathcal{U}(\epsilon)$, $\sigma_{k+1} \geq \gamma_1 \sigma_k$, while for each $k \in \mathcal{S}(\epsilon)$, $\sigma_{k+1} \geq \gamma_3 \sigma_k$. Thus if $k(\epsilon)$ is the iteration for which (7.16) occurs for the first time,

$$\sigma_0 \gamma_1^{|\mathcal{U}(\epsilon)|} \gamma_3^{|\mathcal{S}(\epsilon)|} \leq \sigma_{k(\epsilon)-1} \leq \sigma_{\max}.$$

Taking logarithms, we have

$$|\mathcal{U}(\epsilon)| \log(\gamma_1) + |\mathcal{S}(\epsilon)| \log(\gamma_3) \leq \log(\sigma_{\max}/\sigma_0).$$

Rearranging and recalling that $0 < \gamma_3 < 1$ yields (7.20). □

Combining Lemmas 7.3.5 and 7.3.6 yields the overall iteration complexity bound.

Theorem 7.3.7. *Under the assumptions of Lemma 7.3.5,*

$$|\mathcal{S}(\epsilon)| + |\mathcal{U}(\epsilon)| = O(\epsilon^{-2}). \tag{7.21}$$

Stated differently, Theorem 7.3.7 ensures that either $(f+h)(x_k) \to -\infty$ or that $\liminf \xi(\sigma_{\max}; x_k) = 0$. Because of the difficulty in developing proximal operators for (7.8), we would also like to develop the relax-and-split routine (Algorithm 3) for the subproblem solution. This requires proving that relax-and-split satisfies Theorem 7.3.3 at some finite iteration of this subproblem solver. We outline the problem statement below, but it remains to be proven.

7.3.5 Relax-and-Split solver

We now develop a fast relaxation-based method to solve (7.8) that extends Algorithm 3. We introduce auxiliary variables z and a quadratic relaxation, and consider

$$\underset{s,z}{\text{minimize}} \ \tfrac{1}{2}\|J(x)s+F(x)\|_2^2+\psi(z;x)+\frac{1}{2\gamma}\|s-z\|^2 \quad \text{subject to } \|z\| \le \min(\beta\|s_1\|, \Delta), \tag{7.22}$$

To understand the structure of (7.22), we can first consider the conceptual approach were we partially minimize in s, and iterate over z. Partially minimizing in s yields a strongly convex quadratic program in z, giving us the value function optimization problem

$$\min_z Q_\gamma(z) + \psi(z;x) \quad \text{subject to } \|z\| \le \min(\beta\|s_1\|, \Delta), \tag{7.23}$$

where

$$\begin{aligned} s(z) &= \left(J^TJ + \frac{1}{\gamma}I\right)^{-1}\left(J^TF + \frac{1}{\gamma}z\right) \\ Q_\gamma(z) &= \tfrac{1}{2}\|J(x)s(z)+F(x)\|_2^2 + \frac{1}{2\gamma}\|s(z)-z\|^2. \end{aligned} \tag{7.24}$$

We can then solve (7.23) using proximal gradient. Every time z is updated, we must update $s(z)$ also, using (7.24). The derivative of Q_γ with respect to z is easily obtained:

$$\nabla Q_\gamma(z) = \frac{1}{\gamma}(z - s(z)).$$

The practical implementation of the algorithm is given by

$$\begin{aligned} s^+ &= \arg\min_s \tfrac{1}{2}\|J(x)s+F(x)\|_2^2 + \frac{1}{2\gamma}\|s-z\|^2 \\ z^+ &= \underset{\gamma(\psi+\chi(\Delta\mathbb{B})}{\text{prox}}(s^+) \end{aligned} \tag{7.25}$$

The method can be analyzed and converges quickly, since Q_γ is much better conditioned than the original problem in (7.22), see [153]. The difficulty is now to account for γ in the overall analysis. Also we have had good performance with only a few CG iterations for the s^+ update, and that is another interesting thing to consider here.

7.3.6 RS Analysis

We can write RS as proximal gradient descent on Q_γ. Define $\mathbf{z} = [s, z]$, which yields the new formulation

$$\underset{\mathbf{z}}{\text{minimize}} \underbrace{\left\| \begin{bmatrix} J & 0 \\ \gamma^{-1/2} I & -\gamma^{-1/2} I \end{bmatrix} \mathbf{z} + \begin{bmatrix} F \\ 0 \end{bmatrix} \right\|^2}_{G(\mathbf{z})} + \boldsymbol{\Psi}(\mathbf{z})$$

and proximal gradient with step-size α is given by

$$\mathbf{z}_{k+1} = \underset{\alpha\Psi}{\text{prox}}(\mathbf{z}_k - \alpha \nabla G(\mathbf{z}))$$

Bounding distance between s and z

First we observe that $z_k \in \Delta\mathbb{B}$ for all k, by the iteration (7.25). From this same iteration, we have

$$s_k = \arg\min_s \frac{1}{2}\|Js + F\|^2 + \frac{1}{2\nu}\|s - z^k\|^2$$

where the dependence of J and F on x has been suppressed to simplify exposition. Taking the gradient and setting it equal to 0 yields

$$\begin{aligned} s^k &= \left(J^T J + \frac{1}{\nu} I\right)^{-1} (-J^T F + \frac{1}{\nu} z^k) \\ &= -\left(J^T J + \frac{1}{\nu} I\right)^{-1} J^T F + \frac{1}{\nu}\left(\frac{1}{\nu} I + J^T J\right)^{-1} z^k \\ &= -\left(J^T J + \frac{1}{\nu} I\right)^{-1} J^T F + \left(I - \nu J^T (I + \nu J J^T)^{-1} J\right) z^k \end{aligned}$$

where we applied the first variant of the Woodbury formula. From here, we get

$$s^k - z^k = -\left(J^T J + \frac{1}{\nu} I\right)^{-1} J^T F - \nu J^T (I + \nu J J^T)^{-1} z^k$$

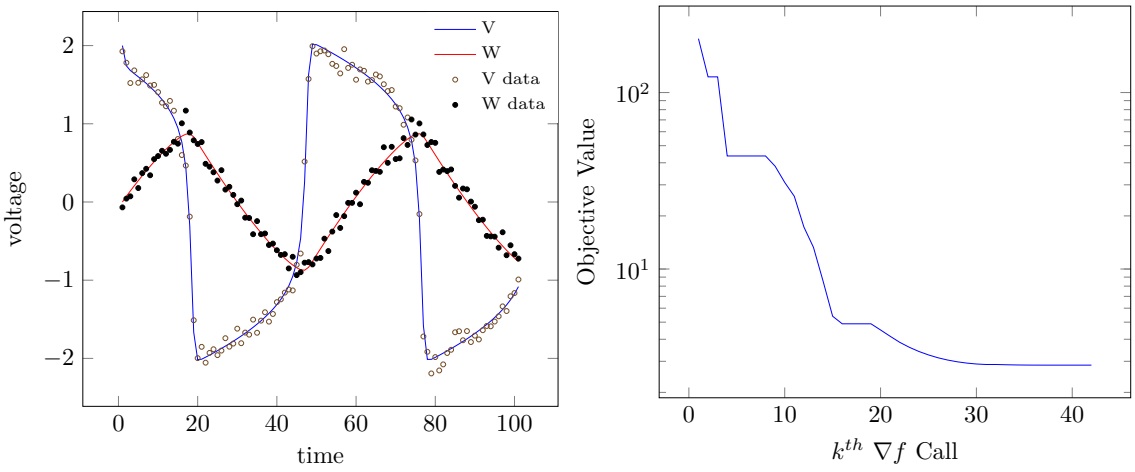
$$\begin{aligned} \|s^k - z^k\| &\leq \nu \left(\left\| \left(\nu J^T J + I \right)^{-1} J^T \right\| \|F\| + \left\| J^T (I + \nu J J^T)^{-1} J \right\| \Delta \right) \\ &\leq \nu \left(\left\| J^T \right\| \|F\| + \left\| J^T J \right\| \Delta \right) \end{aligned} \tag{7.26}$$

giving us uniform control of the distance from s^k to z^k given F and Δ. The trick is to bound the distance between Q_γ and $\|Js + F\|_2^2$, which still requires some work to complete.

7.3.7 Preliminary Results, Conclusions & Future Directions

We have run algorithms detailed in both section 7.3.2 and section 7.3.3 on the Fitzhugh-Nagumo ODE inverse problem detailed in section 5.6. Interestingly, we have been able to adapt QR (Algorithm 17) instead of PG (Algorithm 1), which actually reduces the number of inner solves. These are detailed for Algorithm 24 only in Figure 7.1, as it the solution fails for the QR variant in section 7.3.2. The latter is primarily because the ODE has parameter regions where the solution cannot exist; we note that this is not a failing of the methodology, but rather that of the problem itself. A way to address this is to expand the existing methodology to include bound constraints, which we have mentioned before in section 7.2.

Figure 7.1: Fitzhugh-Nagumo solution (5.49) for $h(x) = \lambda\|x\|_0$ in (5.51) with ℓ_∞-norm, LMTR.



(a) Solution for LMTR

(b) Objective Descent for LMTR

There are still several things to accomplish in this work. The first is to finish the theory, which again involves some complexity analysis as well as showing that Theorem 5.5.2 and Theorem 7.3.3 are satisfied for the subproblem we use. The second is develop a package for download; we have preliminary numerical results with algorithms that we have added to our repository https://github.com/UW-AMO/TRNC. Another is finish up the proofs for both the relax and split formulations as well as the LM methods. The last is demonstrate the efficacy of our methods on new problems; one of these will be discussed shortly.

7.4 *Relax and Split Tuning Parameters*

The question left unsolved by Chapters 3 and 4 is how to adjust η correctly. One thought is to look at the "exact relaxation" or the identity

$$\|x\|_2 = \arg\min_{\sigma} \frac{1}{\sigma}\|x\|^2 + \sigma. \tag{7.27}$$

Here, we take advantage of the relationship 7.27 to augment the formulation to yield

$$\min_{x,w,\sigma} \phi(w) + \varphi(x) + \frac{1}{\eta}\left(\frac{1}{\sigma}\|\mathcal{C}(x) - w\|^2 + \sigma\right). \tag{7.28}$$

Perhaps the most intuitive algorithm would be to implement a gradient descent/proximal gradient descent in the sense of PAM or PALM. This would requires solving for x via some means dependent on the convexity of $\varphi(x)$. The new version of Algorithm 3 would now include a parameter update, give in Algorithm 25.

Algorithm 25 Prox-gradient for (7.28).

Input: x_0, w_0, σ_0

Initialize: $k = 0$

while not converged **do**

$x_{k+1} \leftarrow \arg\min_x \varphi(x) + \frac{1}{\eta\sigma_k}\|x - w_k\|^2$

$w_{k+1} \leftarrow \text{prox}_{\eta\sigma_{k+1}\phi}(\mathcal{C}(x_{k+1}))$

$\sigma_{k+1} \leftarrow \|\mathcal{C}(x_{k+1}) - w_{k+1}\|_2$

$k \leftarrow k + 1$

Output: w_k, σ_k, x_k

However, early numerical examples have proven unfruitful. It might be more conducive to create a restart scheme for different η values, or relating it to the gap between the splitting variable and the relaxed part of the function.

7.5 Optimization algorithms as ODE timesteppers for gradient flows

This section briefly discusses how some of the methods from this work can be used in the realm of PDE optimization. Indeed, this relationship can be mutually beneficial, as it is possible to interpret trust region methods as controlled gradient-flow problems.

Just as proximal gradient descent can be likened to backwards Euler for numerical gradient flow, so too can trust region methods be analyzed in a similar sense. Splitting methods have a long and rich history in PDE optimization [15], and many optimization techniques have been adapted or at least have roots in older PDE methods. ADMM, for instance, was initially developed as a PDE solution routine [91]. We would like to explore whether or not the trust region method is a dynamical system in disguise. Initially, we stick to smooth, convex functions to simplify the analysis, and then extend this work to Chapter 5. Say we want to solve the nonlinear optimization problem

$$\min_u J(u) \tag{7.29}$$

where $J : V \to \mathbb{R}$ is a smooth convex functional defined on some reflexive Banach space V. We let V be a Sobolev space and is therefore distinct from its dual space V^*. If $M : V \to V^*$ is any self-adjoint, positive-definite linear operator and τ is some time scale, then we can form the ODE

$$\tau M \dot{u} = -\mathrm{d}J(u). \tag{7.30}$$

Different optimization methods can be thought of as different choices for M. For a problem posed in $L^2(\Omega)$ for some domain Ω, often M is the canonical injection of the space into its dual. Newton's method amounts to using $M = \mathrm{d}^2 J(u)$. One can then think of practical descent methods as a particular choice of timestepping scheme for this ODE. An undamped descent method would correspond to using a constant timestep δt and updating u as

$$u_{n+1} = u_n - \frac{\delta t}{\tau} M^{-1} \mathrm{d}J(u_n), \tag{7.31}$$

leaving the possibility that timestep δt_n could be updated via timestep. Another variant can be derived by applying a linearly implicit Euler discretization to the pure gradient flow from

equation (7.30):

$$u_{n+1} = u_n - \left(\frac{\tau}{\delta t} M + \mathrm{d}^2 J(u_n)\right)^{-1} \mathrm{d}J(u_n) \tag{7.32}$$

which looks more like the implicit regularization afforded by trust region methods. Small timesteps give us something closer to gradient descent, while large timesteps give us something closer to Newton's method. Future interpretations would be instead using a higher-order scheme such as RK4/5, which is also an adaptive method. A question we would like to explore is: can adaptive timestepping schemes for ODE help us design better line search methods?

Newton's method in practice typically has an eigenvalue of the objective's second derivative very close to zero. Trust region methods [40] help regularize away some of the difficulty by requiring that the norm of the step is bounded by some factor Δ, and adjusting Δ based on the decrease in the objective. The norm of the step in a Banach space is induced by some self-adjoint, positive-definite operator K. The appropriate generalization of the trust region condition is that

$$\tau \|\dot{u}\|_K \leq \Delta \tag{7.33}$$

at all times. The Lagrange multiplier λ enforces this constraint, giving us the differential-algebraic equation

$$\tau(M + \lambda K)\dot{u} = -\mathrm{d}J(u), \tag{7.34}$$

$$\lambda(\Delta - \tau\|\dot{u}\|_K) = 0, \tag{7.35}$$

$$\lambda \geq 0, \tag{7.36}$$

$$\Delta - \tau\|\dot{u}\|_K \geq 0. \tag{7.37}$$

This is essentially an ODE with inequality constraints, but can be thought of as a mechanical system with "impacts", see [130]. A naive explicit Euler discretization of this ODE yields a perfectly acceptable inequality-constrained optimization problem for the state of the system at the next timestep.

This form allows us to keep the rate of change of u less than some fixed value of Δ.

Adjusting Δ can be thought of as a *discrete-time control* problem. At discrete timesteps τ, we adjust the value of Δ based on the ratio of the actual to predicted decrease in J:

$$\rho_{(n+1)\tau} = \frac{J(u_{n\tau}) - J(u_{(n+1)\tau})}{-\left\langle \mathrm{d}J(u_{n\tau}), u_{(n+1)\tau} - u_{n\tau} \right\rangle + \frac{1}{2}\left\langle \mathrm{d}^2 J(u_{n\tau})(u_{(n+1)\tau} - u_{n\tau}), u_{(n+1)\tau} - u_{n\tau} \right\rangle}. \quad (7.38)$$

We can then think of $\Delta_{n\tau}$ as a control that keeps $\rho_{n\tau}$ above some threshold value. The rest would be determined essentially by Algorithm 9. This is preliminary and there is still much to the be done, such as form an explicit problem statement, pick several other algorithms to interpret, and explore numerical tests for PDE inverse problems. In the long-term, we would like to extend this to nonsmooth problems.

7.6 Brief Retrospective

In this work, we discuss multiple optimization algorithms and attempt to connect them together via the common theme of nonsmooth optimization and inverse problem techniques. There are many routines that perform variations on these tasks. ADMM can solve nonconvex, nonsmooth functions but requires model assumptions that are difficult to establish in many regimes, still fails even for some convex cases, and also requires linear constraints. TR optimization has been the workhorse for many nonconvex inverse optimization problems but often fails to handle nonsmooth regularizers. Proximal algorithms have also been around for decades (even centuries if you consider the backwards Euler interpretation) and are very generalizable. However, they also rely on information that one may not be privy to (such as Lipschitz constants) and are first order methods typically featuring slow convergence in difficult settings.

Chapter 5 is perhaps the most rewarding and intriguing development; typically nonsmooth optimiation relies on first order methods such as PG, and if second order methods (or indeed any method which makes use of curvature information) exist, they are only for convex functions or are practically impossible to implement. The latter condition arises due to the nature of the Moreau Envelope (2.1a) - the inner product changes from an ℓ_2 norm to that induced by the Hessian. Developing closed form proximal operators of these are even

more difficult than ours in section 5.4. However, in Chapter 5, we've seemingly finagled out of this bind; not only can trust regions now handle nonsmooth functions, but nonsmooth functions now incur the cost-effectiveness and nice convergence properties of trust regions. As discussed earlier and in section 7.2 and section 7.3, this opens the door to the plethora of trust region extensions and has given me enough work to do beyond the forseeable future. A common trope in many optimization papers is overselling the title - *nonsmooth, nonconvex* functions but they apply to separate pieces; here, we've managed to at least let *nonconvexity* apply to both f and h.

Another (hopefully) bountiful research area is the inexact extensions. A common trope amongst deterministic algorithms (which the scope is limited to currently) is that they are inefficient on modern computing architectures. With the popularization of machine learning and the alarmingly large amount of data being collected by/on everyone, the community has seen the rapid development stochastic algorithms. These (typically) have even worse convergence than their deterministic counterparts but are numerically faster, since one only needs access to partial information that converges to minima in expectation. Stochastic versions of trust regions and proximal gradient descent exist and are widely used, and much of the above work can likely be translated to the stochastic regime. Similar issues with those arise; smoothness and convexity are desired, and theory for nonsmoothness and nonconvexity tend to become niche in scope. Hopefully, inexact regimes can sidestep some of these issues for the time being. Such things have already been employed for trust region algorithms extensively [4] and have enjoyed success as part of Sandia's ROL package.

Another rich area of development is making the connections between optimization algorithms and their numerical PDE predecessors. Have already been alluded to several times in this work, it seems that there has been a fairly sizable disconnection between the optimization and PDE communities, when at one point they used to be the same. Evidence of this can be seen in ADMM, which was originally developed as a PDE solution scheme [91], or in the backwards Euler interpretation of the Moreau Envelop [10] that the community has been slowly re-evaluating/discovering. There seems to be much room for development

here; indeed, talks at PDE conferences has generally had favorable receptions to regularizing nonsmooth terms, but much work needs to be done to make this convincingly practical.

This brings me to my last point; pushing algorithms to be made readily available, runnable, and downloadable for the general public. It can be a rather thankless task sometimes, but too often are papers written without consideration for algorithm development. Hopefully, this is incentivized for researchers in the future. Language choice often is a difficult problem, as it essentially chooses the audience for the package. This is becoming a bit less true as Python starts to interact more with C++/C, and as languages such as Julia become more developed. Most of the work described here is written in Matlab or Julia; moving into the PDE optimization community will certainly lead to more C++ utilization. Either way it looks like I have my work cut out for me.

BIBLIOGRAPHY

[1] B.M. Adams, H.T. Banks, M. Davidian, Hee-Dae Kwon, H.T. Tran, S.N. Wynne, and E.S. Rosenberg. HIV dynamics: Modeling, data analysis, and optimal treatment protocols. *Journal of Computational and Applied Mathematics*, 184(1), 2005.

[2] Fred Aminzadeh, N Burkhard, L Nicoletis, Fabio Rocca, and K Wyatt. SEG/EAEG 3-D modeling project: 2nd update. *The Leading Edge*, 13(9):949–952, 1994.

[3] Andersen Man Shun Ang and Nicolas Gillis. Accelerating nonnegative matrix factorization algorithms using extrapolation. *Neural Computation*, 31(2):417–439, 2019. doi: 10.1162/neco_a_01157.

[4] Harbir Antil, Drew P Kouri, Martin D. Lacasse, and Denis Ridzal. Inexact trust-region methods for pde-constrained optimization. In Daniel Sprin, editor, *Frontiers in PDE-Constrained Optimization*, pages 83–121. Springer, 2018.

[5] Aleksandr Aravkin and Damek Davis. Trimmed statistical estimation via variance reduction. *Math. Oper. Res.*, 45(1):292–322, 2020. doi: 10.1287/moor.2019.0992.

[6] Aleksandr Aravkin, Stephen Becker, Volkan Cevher, and Peder Olsen. A variational approach to stable principal component pursuit. *UAI Proceedings*, 2014.

[7] 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):237–266, 2014.

[8] Aleksandr Y Aravkin, James V Burke, Dmitriy Drusvyatskiy, Michael P Friedlander,

and Scott Roy. Level-set methods for convex optimization. *To appear in Mathematical Programming, Series B.*, 2018.

[9] Hédy Attouch, Jérôme Bolte, Patrick Redont, and Antoine Soubeyran. Proximal alternating minimization and projection methods for nonconvex problems: An approach based on the Kurdyka-Łojasiewicz inequality. *Mathematics of Operations Research*, 35 (2):438–457, 2010.

[10] Hedy Attouch, Jérôme Bolte, and Benar Fux Svaiter. Convergence of descent methods for semi-algebraic and tame problems: proximal algorithms, forward–backward splitting, and regularized Gauss-Seidel methods. *Math. Program.*, 137(1-2):91–129, 2013. doi: 10.1007/s10107-011-0484-9.

[11] G. Banjac and P. J. Goulart. A novel approach for solving convex problems with cardinality constraints. *IFAC-PapersOnLine*, 50(1):13182–13187, 2017.

[12] Robert Baraldi, Rajiv Kumar, and Aleksandr Aravkin. Basis pursuit denoise with nonsmooth constraints. *IEEE T. Signal Proces.*, 67(22):5811–5823, 2019. doi: 10.1109/TSP.2019.2946029.

[13] Robert Baraldi, Carl Ulberg, Rajiv Kumar, Kenneth Creager, and Aleksandr Aravkin. Relaxation algorithms for matrix completion, with applications to seismic travel-time data interpolation. *Inverse Problems*, 35(10):105009, 2019.

[14] Robert Baraldi, Aleksandr Aravkin, and Dominique Orban. A proximal quasi-newton trust-region method for nonsmooth regularized optimization. *SIAM Journal of Optimization - in revision*, 2020. URL https://eartharxiv.org/repository/view/2267/.

[15] Heinz H. Bauschke and Patrick L. Combettes. *Convex Analysis and Monotone Operator Theory in Hilbert Spaces*. Springer Science, 2011. doi: 10.1007/978-3-319-48311-5.

[16] A. Beck and Y. C. Eldar. Sparsity constrained nonlinear optimization: Optimality conditions and algorithms. *SIAM Journal of Optimization*, 23(3):1480–1509, 2013.

[17] Amir Beck. *First Order Methods in Optimization*. SIAM, Philadelphia, USA, 2017. doi: 10.1137/1.9781611974997.

[18] Amir Beck and Marc Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM J. Imaging Sci.*, 2(1):183–202, 2009. doi: 10.1137/080716542.

[19] Bradley M Bell and James V Burke. Algorithmic differentiation of implicit functions and optimal values. In *Advances in Automatic Differentiation*, pages 67–77. Springer, 2008.

[20] Michael W. Berry, Murray Browne, Amy N. Langville, V. Paul Pauca, and Robert J. Plemmons. Algorithms and applications for approximate nonnegative matrix factorization. *Computational Statistics & Data Analysis*, 52(1):155–173, 2007. doi: 10.1016/j.csda.2006.11.006.

[21] Dimitri P. Bertsekas. *Constrained Optimization and Lagrange Multiplier Methods*. Academic Press, 1982.

[22] Jeff Bezanson, Alan Edelman, Stefan Karpinski, and Viral B Shah. Julia: A fresh approach to numerical computing. *SIAM Rev.*, 59(1):65–98, 2017. URL https://doi.org/10.1137/141000671.

[23] Barron J. Bichon, Michael S. Eldred, Sankaran Mahadevan, and John M. McFarland. Efficient Global Surrogate Modeling for Reliability-Based Design Optimization. *Journal of Mechanical Design*, 135(1), 12 2012. ISSN 1050-0472. doi: 10.1115/1.4022999. URL https://doi.org/10.1115/1.4022999. 011009.

[24] Thomas Blumensath and Mike E Davies. Iterative hard thresholding for compressed sensing. *Appl. Comput. Harmon. A.*, 27(3):265–274, 2009.

[25] Pavel Bochev, Denis Ridzal, Marta D'Elia, Mauro Perego, and Kara Peterson. Optimization-based, property-preserving finite element methods for scalar advection equations and their connection to algebraic flux correction. *Computer Methods in Applied Mechanics and Engineering*, 367:112982, 2020. ISSN 0045-7825. doi: https://doi.org/10.1016/j.cma.2020.112982. URL http://www.sciencedirect.com/science/article/pii/S0045782520301651.

[26] J. Bolte, S. Sabach, and M. Teboulle. Proximal alternating linearized minimization for nonconvex and nonsmooth problems. *Math. Program.*, 1(146):459—-494, 2014. doi: 10.1007/s10107-013-0701-9.

[27] Jérôme Bolte, Shoham Sabach, and Marc Teboulle. Proximal alternating linearized minimization for nonconvex and nonsmooth problems. *Mathematical Programming*, 146(1-2):459–494, 2014.

[28] Léon Bottou, Frank E. Curtis, and Jorge Nocedal. Optimization methods for large-scale machine learning. *SIAM Review*, 60(2):223–311, 2018.

[29] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine Learning*, 3(1):1–122, 2010.

[30] R. I. Boţ, E. R. Csetnek, and S.C. László. An inertial forward–backward algorithm for the minimization of the sum of two nonconvex functions. *EURO J. Comput. Optim.*, 1(4):3–25, 2016. doi: 10.1007/s13675-015-0045-8.

[31] E. J. Candès and Y. Plan. Matrix completion with noise. *Proceedings of the IEEE*, 98 (6):925–936, June 2010.

[32] Emmanuel J. Candès and Benjamin Recht. Exact matrix completion via convex optimization. *Foundations of Computational Mathematics*, 9(6):717–723, Apr 2009.

[33] Emmanuel J Candès and Terence Tao. Near-optimal signal recovery from random projections: universal encoding strategies. *IEEE Transactions on Information Theory*, 52(12):5406–5425, 2006.

[34] Emmanuel J Candès, Li Xiaodong, Yi Ma, and John Wright. Robust principal component analysis? *Journal of the ACM*, 58(3):11, 2011.

[35] Richard G. Carter. On the global convergence of trust region algorithms using inexact gradient information. *SIAM Journal on Numerical Analysis*, 28(1):251–265, 1991.

[36] Coralia Cartis, Nicholas I. M. Gould, and Ph. L. Toint. On the evaluation complexity of composite function minimization with applications to nonconvex nonlinear programming. *SIAM J. Optim.*, 21(4):1721–1739, 2011. doi: 10.1137/11082381X.

[37] Scott Shaobing Chen, David L Donoho, and Michael A. Saunders. Atomic decomposition by basis pursuit. *SIAM Journal on Scientific Computing*, 20(1):33–61, 1998.

[38] Patrick L Combettes and Jean-Christophe Pesquet. Proximal splitting methods in signal processing. In *Fixed-point algorithms for inverse problems in science and engineering*, pages 185–212. Springer, 2011. doi: 10.1007/978-1-4419-9569-8_10.

[39] A. R. Conn, N. I. M. Gould, and Ph L. Toint. Convergence of quasi-Newton matrices generated by the symmetric rank one update. *Math. Program.*, 50(1):177–195, Mar 1991. doi: 10.1007/BF01594934.

[40] A. R. Conn, N. I. M. Gould, and Ph. L. Toint. *Trust-Region Methods*. Number 1 in MOS-SIAM Series on Optimization. SIAM, Philadelphia, USA, 2000. doi: 10.1137/1.9780898719857.

[41] F. E. Curtis, D. P. Robinson, and M. Samadi. A trust region algorithm with a worst-case iteration complexity of $\mathcal{O}(\epsilon^{-3/2})$ for nonconvex optimization. *Math. Program., Series A*, 1(162):1–32, 2017. doi: 10.1007/s10107-016-1026-2.

[42] F.E. Curtis, Z. Lubberts, and D.P. Robinson. Concise complexity analyses for trust region methods. *Optim. Lett.*, 1(12):1713—-1724, 2018. doi: 10.1007/s11590-018-1286-2.

[43] 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.

[44] Damek Davis and Wotao Yin. Convergence rate analysis of several splitting schemes. In *Splitting Methods in Communication, Imaging, Science, and Engineering*, pages 115–163. Springer, 2016.

[45] Aaron Defazio, Francis Bach, and Simon Lacoste-Julien. SAGA: A fast incremental gradient method with support for non-strongly convex composite objectives. *Advances in Neural Information Processing Systems*, pages 1646–1654, 2014.

[46] Guy Demoment. Image reconstruction and restoration: Overview of common estimation structures and problems. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 37(12):259–268, December 1989.

[47] J.E. Dennis, SB.B. Li, and R.A. Tapia. A unified approach to global convergence of trust region methods for nonsmooth optimization. *Math. Program.*, 1(68):319—-346, 1995. doi: 10.1007/BF01585770.

[48] J. E. Dennis Jr. and H. H. W. Mei. Two new unconstrained optimization algorithms which use function and gradient values. *J. Optim. Theory and Applics.*, 28:453—-482, 1979. doi: 10.1007/BF00932218.

[49] David C Dobson and Fadil Santosa. Recovery of blocky images from noisy and blurred data. *SIAM Journal of Applied Mathematics*, 56(4):1181–1198, 1994.

[50] David L Donoho. Compressed sensing. *IEEE T. Inform. Theory*, 52(4):1289–1306, 2006. doi: 10.1109/TIT.2006.871582.

[51] Derek Driggs, Stephen Becker, and Aleksandr Aravkin. Adapting regularized low-rank models for parallel architectures. *SIAM Journal on Scientific Computing*, 41(1): A163–A189, 2019.

[52] B Efron, T Hastie, I Johnstone, and R Tibshirani. Least angle regression. *The Annals of Statistics*, 32(2):407–99, 2004.

[53] Jianqing Fan and Runze Li. Variable selection via nonconcave penalized likelihood and its oracle properties. *J. Am. Stat. Assoc.*, 96(456):1348–1360, 2001. doi: 10.1198/016214501753382273.

[54] Richard FitzHugh. Mathematical models of threshold phenomena in the nerve membrane. *B. Math. Biophys.*, 17(4):257–278, 1955. doi: 10.1007/BF02477753.

[55] Rina Foygel and Lester Mackey. Corrupted sensing: Novel guarantees for separating structured signals. *IEEE Transactions on Information Theory*, 60(2):1223–1247, 2014.

[56] Michael P. Friedlander and Sven Leyffer. Global and finite termination of a two-phase augmented Lagrangian filter method for general quadratic programs. *SIAM Journal on Scientific Computing*, 30(4):1706–1729, 2008. doi: 10.1137/060669930.

[57] Hong-Ye Gao and Andrew G Bruce. Waveshrink with firm shrinkage. *Stat. Sinica*, 7: 855–874, 1997. URL https://www.jstor.org/stable/i24306126.

[58] F Girosi. An equivalence between sparse approximation and support vector machines. *Neural Comp.*, 10(6):1455–1480, 1998.

[59] G.N. Grapiglia, J. Yuan, and Yx. Yuan. Nonlinear stepsize control algorithms: Complexity bounds for first- and second-order optimality. *J. Optim. Theory and Applics.*, 1(171):980—997, 2016. doi: 10.1007/s10957-016-1007-x.

[60] Joshua D. Griffin and Tamara G. Kolda. Nonlinearly Constrained Optimization Using Heuristic Penalty Methods and Asynchronous Parallel Generating Set Search. *Applied*

Mathematics Research eXpress, 2010(1):36–62, 04 2010. ISSN 1687-1200. doi: 10.1093/amrx/abq003. URL https://doi.org/10.1093/amrx/abq003.

[61] Mamikon Gulian, Maziar Raissi, Paris Perdikaris, and George Karniadakis. Machine Learning of Space-Fractional Differential Equations. *SIAM Journal of Scientific Computing*, 41(4):A2485—-A2509, 2019.

[62] Wooseok Ha and Rina Foygel Barber. Robust PCA with compressed data. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 1936–1944. Curran Associates, Inc., 2015. URL http://papers.nips.cc/paper/5705-robust-pca-with-compressed-data.pdf.

[63] Davood Hajinezhad, Tsung-Hui Chang, Xiangfeng Wang, Qingjiang Shi, and Mingyi Hong. Nonnegative matrix factorization using ADMM: Algorithm and convergence analysis. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4742–4746. IEEE, 2016. doi: 10.1109/icassp.2016.7472577.

[64] Warren Hare and Claudia Sagastizábal. Computing proximal points of nonconvex functions. *Math. Program.*, 116(1):221–258, Jan 2009. doi: 10.1007/s10107-007-0124-6. URL https://doi.org/10.1007/s10107-007-0124-6.

[65] Matthias Heinkenschloss and Luis N. Vicente. Analysis of inexact trust-region sqp algorithms. *SIAM Journal on Optimization*, 12(2):283–302, 2002.

[66] Gilles Hennenfent and Felix J Herrmann. Simply denoise: wavefield reconstruction via jittered undersampling. *Geophysics*, 73(3):19–28, 2008.

[67] Gilles Hennenfent, Lloyd Fenelon, and Felix J. Herrmann. Nonequispaced curvelet transform for seismic data reconstruction: a sparsity-promoting approach. *Geophysics*, 75(6):WB203–WB210, 12 2010.

[68] F. J. Herrmann, X. Li, A. Y. Aravkin, and T. van Leeuwen. A modified, sparsity-promoting, gauss-newton algorithm for seismic waveform inversion. In *Proceedings Volume 8138, Wavelets and Sparsity XIV*, volume 8138, pages 81380V–81380V–14, 2011. doi: 10.1117/12.893861.

[69] Felix J. Herrmann and Gilles Hennenfent. Non-parametric seismic data recovery with curvelet frames. *Geophysical Journal International*, 173(1):233–248, 2008.

[70] Nicholas J. Higham, Srikara Pranesh, and Mawussi Zounon. Squeezing a matrix into half precision, with an application to solving linear systems. *SIAM Journal on Scientific Computing*, 41(4):A2536–A2551, 2019.

[71] J. A. Hole and B. C. Zelt. 3-D finite-difference reflection traveltimes. *Geophysical Journal International*, 121(2):427–434, 1995.

[72] Sashank J. Reddi, Suvrit Sra, Barnabas Poczos, and Alexander J Smola. Proximal stochastic methods for nonsmooth nonconvex finite-sum optimization. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016. URL https://proceedings.neurips.cc/paper/2016/file/291597a100aadd814d197af4f4bab3a7-Paper.pdf.

[73] Ajinkya Kadu and Rajiv Kumar. Decentralized full-waveform inversion. Submitted to EAGE on January 15, 2018, 2018. URL https://www.slim.eos.ubc.ca/Publications/Private/Submitted/2018/kadu2018EAGEdfwi/kadu2018EAGEdfwi.pdf.

[74] N. Keskar and A. Wächter. A limited-memory quasi-Newton algorithm for bound-constrained non-smooth optimization. *Optimization Methods and Software*, 34(1):150–171, 2019.

[75] Dongmin Kim, Suvrit Sra, and Inderjit S. Dhillon. A scalable trust-region algorithm with application to mixed-norm regression. In *ICML*, pages 519–526, 2010. URL https://icml.cc/Conferences/2010/papers/562.pdf.

[76] E. Kiser, A. Levander, C. A. Zelt, I. Palomeras, K. Creager, C. W. Ulberg, B. Schmandt, S. M. Hansen, S. H. Harder, G. A. Abers, and K. Crosbie. Three-dimensional velocity models of the Mount St. Helens magmatic system using the iMUSH active-source data set. In *AGU Fall Meeting Abstracts*, Dec 2017.

[77] Eric Kiser, Alan Levander, Colin Zelt, Brandon Schmandt, and Steven Hansen. Focusing of melt near the top of the Mount St. Helens (USA) magma reservoir and its relationship to major volcanic eruptions. *Geology*, 2018.

[78] Drew P Kouri, Matthias Heinkenschloss, Denis Ridzal, and Bart G van Bloemen Waanders. Inexact objective function evaluations in a trust-region algorithm for pde-constrained op- timization under uncertainty. *SIAM Journal on Scientific Computing*, 36(6):A3011–3029, 2014.

[79] Drew Philip Kouri, Denis Ridzal, and Raymond S. Tuminaro. KKT Preconditioners for PDE-Constrained Optimization with the Helmholtz Equation. *SIAM SISC*, page Submitted, 2020.

[80] R. Kumar, O. López, D. Davis, A. Y. Aravkin, and F. J. Herrmann. Beating level-set methods for 5-D seismic data interpolation: A primal-dual alternating approach. *IEEE Transactions on Computational Imaging*, 3(2):264–274, June 2017.

[81] 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):97–114, 2015.

[82] Rajiv Kumar, Haneet Wason, and Felix J. Herrmann. Source separation for simultane-

ous towed-streamer marine acquisition — a compressed sensing approach. *Geophysics*, 80(6):WD73–WD88, 11 2015.

[83] Martin Landrø. Uncertainties in quantitative time-lapse seismic analysis. *Geophysical Prospecting*, 50(5):527–538, 2002.

[84] Martin Landrø. The effect of noise generated by previous shots on seismic reflection data. *Geophysics*, 73(3):9–17, 2008.

[85] Daniel D. Lee and H. Sebastian Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–791, 1999. doi: 10.1038/44565.

[86] Daniel D Lee and H Sebastian Seung. Algorithms for non-negative matrix factorization. In T. K. Leen, T. G. Dietterich, and V. Tresp, editors, *Advances in Neural Information Processing Systems 13*, pages 556–562. MIT Press, 2001. URL http://papers.nips.cc/paper/1861-algorithms-for-non-negative-matrix-factorization.pdf.

[87] Jason D. Lee, Yuekai Sun, and Michael A. Saunders. Proximal Newton-type methods for minimizing composite functions. *SIAM J. Optim.*, 24(3):1420–1443, 2014. doi: 10.1137/130921428.

[88] K. Levenberg. A method for the solution of certain problems in least squares. *Q. Appl. Math.*, 1(2):164–168, 1944. doi: 10.1090/qam/10666.

[89] Huan Li and Zhouchen Lin. Accelerated proximal gradient methods for nonconvex programming. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'15, pages 379–387, Cambridge, MA, USA, 2015. MIT Press. URL http://irc.cs.sdu.edu.cn/973project/result/download/2015/28.AcceleratedProximal.pdf.

[90] F. Lin, M. Fardad, and M. Jovanović. Design of optimal sparse feedback gains via the alternating direction method of multipliers. *IEEE Transactions on Automatic Control*, 58(9):2426–2431, 2013.

[91] P.L. Lions and B Mercier. Splitting algorithms for the sum of two nonlinear operators. *SIAM J. Numer. Anal.*, 16(6):964—-979, 1979. doi: 10.1137/0716071.

[92] S. Lotfi, T. Bonniot de Ruisselet, D. Orban, and A. Lodi. Stochastic damped L-BFGS with controlled norm of the Hessian approximation. In *OPT2020 Conference on Optimization for Machine Learning*, 2020. doi: 10.13140/RG.2.2.27851.41765/1.

[93] S. Lu, Z. Wei, and L. Li. A trust-region algorithm with adaptive cubic regularization methods for nonsmooth convex minimization. *Comput. Optim. Appl.*, 1(51):551–573, 2012.

[94] M. Lustig, D. Donoho, and J. Pauly. Sparse mri: The application of compressed sensing for rapid mri imaging. *Magnetic Resonance in Medicine*, 58:1182–95, 2007.

[95] J. Mairal, F. Bach, and J. Ponce. Sparse modeling for image and vision processing. *Foundations and Trends in Computer Graphics and Vision*, 8((2-3)):85–283, 2014.

[96] Stephane Mallat. *A wavelet tour of signal processing: the sparse way*. Academic press, 2008.

[97] S. D. Malone and G. L. Pavlis. Velocity structure and relocation of earthquakes at Mount St. Helens. In *EosTrans AGU*, volume 64, page 895, 1983.

[98] Donald W. Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *Journal of the Society for Industrial and Applied Mathematics*, 11(2):431–441, 1963. doi: 10.1137/0111030.

[99] JRRA Martins, Peter Sturdza, and Juan Alonso. The connection between the complex-step derivative approximation and algorithmic differentiation. In *39th Aerospace Sciences Meeting and Exhibit*, page 921, 2001.

[100] J. M. Martínez and A. C. Moretti. A trust region method for minimization of nonsmooth functions with linear constraints. *Math. Program.*, 1(76):431–449, 1997. doi: 10.1007/BF02614392.

[101] Kurt Maute, Gary Weickum, and Mike Eldred. A reduced-order stochastic finite element approach for design optimization under uncertainty. *Structural Safety*, 31(6): 450 – 459, 2009. ISSN 0167-4730. doi: https://doi.org/10.1016/j.strusafe.2009.06.004. URL http://www.sciencedirect.com/science/article/pii/S0167473009000514. Optimization under Uncertainty with Emphasis on Structural Applications.

[102] Jorge J. Moré and D. C. Sorensen. Computing a trust region step. *SIAM J. Sci. and Statist. Comput.*, 4(3):553–572, 1983. doi: 10.1137/0904038.

[103] Jinichi Nagumo, Suguru Arimoto, and Shuji Yoshizawa. An active pulse transmission line simulating nerve axon. *Proceedings of the IRE*, 50(10):2061–2070, 1962. doi: 10.1109/JRPROC.1962.288235.

[104] YU. Nesterov. Modified Gauss–Newton scheme with worst case guarantees for global performance. *Optim. Method Softw.*, 22(3):469–483, 2007. doi: 10.1080/08927020600643812.

[105] Mila Nikolova. Minimizers of cost-functions involving the nonsmooth data-fidelity terms. application to the processing of outliers. *SIAM Journal of Numerical Analysis*, 40(3):965–994, 2002.

[106] J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer, New York, 2nd edition, 2006.

[107] Peter Ochs and Thomas Pock. Adaptive fista for nonconvex optimization. *SIAM Journal of Optimization*, 29(4):2482–2503, 2019.

[108] Luca Oneto, Sandro Ridella, and Davide Anguita. Tikhonov, Ivanov and Morozov regularization for support vector machine learning. *Machine Learning*, 103(1):103–136, 2016.

[109] Dominique Orban and Able Soares Siqueira. Linearoperators.jl., February 2019.

[110] G. Pang, M. D'Elia, M. Parks, and G.E. Karniadakis. npinns: Nonlocal physics-informed neural networks for a parametrized nonlocal universal laplacian operator. algorithms and applications. *Journal of Computational Physics*, 422:109760, 2020. ISSN 0021-9991. doi: https://doi.org/10.1016/j.jcp.2020.109760. URL http://www.sciencedirect.com/science/article/pii/S0021999120305349.

[111] R. Pawlowski, E. Phipps, A. Salinger, S. Owen, C. Siefert, and Staten M. Automating embedded analysis capabilities and managing software complexity in multiphysics simulation, Part II: Application to partial differential equations. *Scientific Programming*, 20(3), 2012.

[112] W. Scott Phillips and Michael C. Fehler. Traveltime tomography: A comparison of popular methods. *Geophysics*, 56(10):1639–1649, 1991.

[113] M. J. D. Powell. A new algorithm for unconstrained optimization. In J.B. Rosen, O.L. Mangasarian, and K. Ritter, editors, *Nonlinear Programming*, pages 31–65. Academic Press, 1970. doi: 10.1016/B978-0-12-597050-1.50006-3.

[114] L. Qi and J. Sun. A trust region algorithm for minimization of locally Lipschitzian functions. *Math. Program.*, 1(66):25—-43, 1994. doi: 10.1007/BF01581136.

[115] Christopher Rackauckas and Qing Nie. Differentialequations.jl–a performant and feature-rich ecosystem for solving differential equations in Julia. *J. Open Res. Softw.*, 5(1), 2017. doi: 10.5334/jors.151.

[116] Julian Rasch and Antonin Chambolle. Inexact first-order primal–dual algorithms. *Computational Optimization and Applications*, 76:381–430, 2020.

[117] Benjamin Recht, Maryam Fazel, and Pablo A. Parrilo. Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization. *SIAM Rev.*, 52 (3):471–501, August 2010. ISSN 0036-1445. doi: 10.1137/070697835. URL http://dx.doi.org/10.1137/070697835.

[118] J. Revels, M. Lubin, and T. Papamarkou. Forward-mode automatic differentiation in Julia. *arXiv:1607.07892 [cs.MS]*, 2016. URL https://arxiv.org/abs/1607.07892.

[119] R. Tyrrell Rockafellar. Monotone operators and the proximal point algorithm. *SIAM Journal of Control and Optimization*, 14(5), 1976.

[120] R.T. Rockafellar and R.J.B. Wets. *Variational Analysis*, volume 317. Springer Verlag, 1998. doi: 10.1007/978-3-642-02431-3.

[121] Leonid I Rudin, Stanley Osher, and Emad Fatemi. Nonlinear total variation based noise removal algorithms. *Physica D*, 60:259–268, 1992.

[122] M. D. Sacchi, T. J. Ulrych, and C. J. Walker. Interpolation and extrapolation using a high-resolution Discrete Fourier transform. *IEEE Transactions on Signal Processing*, 46(1):31–38, Jan 1998.

[123] Katya Scheinberg and Xiaocheng Tang. Practical inexact proximal quasi-Newton method with global complexity analysis. *Mathematical Programming*, 160:495—-529, 2016.

[124] Mark Schmidt, Eout van den Berg, Michael Friedlander, and Kevin Murphy. Optimizing costly functions with simple constraints: A limited-memory projected quasi-newton algorithm. In David van Dyk and Max Welling, editors, *Proceedings of the Twelth International Conference on Artificial Intelligence and Statistics*, volume 5 of *Proceedings of Machine Learning Research*, pages 456–463. PMLR, 2009.

[125] Mark Schmidt, Nicolas L. Roux, and Francis R. Bach. Convergence rates of inexact proximal-gradient methods for convex optimization. In *Advances in Neural Information Processing Systems 24*, pages 1458–1466. Curran Associates, Inc., 2011.

[126] Ivan W. Selesnick and İlker Bayram. Enhanced sparsity by non-separable regularization. *IEEE Transactions on Signal Processing*, 64(9):2298 – 2313, 2016.

[127] Ivan W. Selesnick and Po-Yu Chen. Group-sparse signal denoising: Non-convex regularization, convex optimization. *IEEE Transactions on Signal Processing*, 62(13): 3464–3478, 2014.

[128] T. Steihaug. The conjugate gradient method and trust regions in large scale optimization. *SIAM J. Numer. Anal.*, 20(3):626–637, 1983. doi: 10.1137/0720042.

[129] L. Stella, A. Themelis, P. Sopasakis, and P. Patrinos. A simple and efficient algorithm for nonlinear model predictive control. In *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, pages 1939–1944, 2017. doi: 10.1109/CDC.2017.8263933.

[130] David E Stewart. *Dynamics with Inequalities: impacts and hard constraints*. SIAM, 2011.

[131] A. Tarantola. *Inverse problem theory: Methods for data fitting and model parameter estimation*. SIAM: Society for Industrial and Applied Mathematics, 1 1987.

[132] Andreas Themelis, Lorenzo Stella, and Panagiotis Patrinos. Forward-backward envelope for the sum of two nonconvex functions: Further properties and nonmonotone linesearch algorithms. *SIAM J. Optim.*, 28(3):2274–2303, 2018. doi: 10.1137/16M1080240.

[133] Robert Tibshirani. Regression shrinkage and selection via the lasso. *J. Roy. Statist. Soc. Ser. B*, 58(1):267–288, 1996. doi: 10.1111/j.2517-6161.1996.tb02080.x.

[134] Kim-Chuan Toh and Sangwoon Yun. An accelerated proximal gradient algorithm for

nuclear norm regularized least squares problems. *Pacific Journal of Optimization*, 6 (3):615–640, 2009.

[135] Philippe L. Toint. Global convergence of a class of trust-region methods for nonconvex minimization in hilbert space. *IMA Journal of Numerical Analysis*, 8(2):231–252, 1988.

[136] L. N. Trefethen and D. Bau. *Numerical Linear Algebra*. Society for Industrial and Applied Mathematics, Philadelphia, 1 edition, 1997.

[137] Ashish Tripathi, Sven Leyffer, Todd Munson, and Stefan M. Wild. Visualizing and improving the robustness of phase retrieval algorithms. *Procedia Computer Science*, 51(1):815–824, 6 2015.

[138] Joel A. Tropp. Just relax: Convex programming methods for identifying sparse signals in noise. *IEEE Transactions on Information Theory*, 52(3):1030–1050, March 2006.

[139] Paul Tseng. Convergence of a block coordinate descent method for nondifferentiable minimization. *Journal of optimization theory and applications*, 109(3):475–494, 2001.

[140] Daniel Turner, Bart van Bloemen Waanders, and Michael L. Parks. Inverse problems in heterogeneous and fractured media using peridynamics. *Journal of Mechanics of Materials and Structures*, 10(5):573–590, 2015.

[141] C. W. Ulberg, K. Creager, S. C. Moran, G. A. Abers, K. Crosbie, R. S. Crosson, R. P. Denlinger, W. A. Thelen, E. Kiser, A. Levander, and O. Bachmann. Imaging seismic zones and magma beneath Mount St. Helens with the iMUSH Broadband Array. In *AGU Fall Meeting Abstracts*. Submitted, Dec 2017.

[142] Ewout van den Berg and Michael P. Friedlander. Probing the pareto frontier for basis pursuit solutions. *SIAM J. Sci. Comput.*, 31(2):890–912, November 2008. ISSN 1064-8275. doi: 10.1137/080714488. URL http://dx.doi.org/10.1137/080714488.

[143] Ewout Van den Berg and Michael P Friedlander. Sparse optimization with least-squares constraints. *SIAM J. Optim.*, 21(4):1201–1229, 2011.

[144] Balth Van der Pol. Lxxxviii. On "relaxation-oscillations". *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11):978–992, 1926. doi: 10.1080/14786442608564127.

[145] Charlie Vanaret and Sven Leyffer. An augmented Lagrangian filter method. *Mathematical Methods of Operations Research*, 2020.

[146] John E. Vidale. Finite-difference calculation of traveltimes in three dimensions. *Geophysics*, 55(5):521–526, 1990.

[147] Silvia Villa, Saverio Salzo, Luca Baldassarre, and Allesandro Verri. Accelerated and inexact forward-backward algorithms. *SIAM Journal on Optimization*, 23(3), 2013.

[148] Yu Wang, Wotao Yin, and Jinshan Zeng. Global convergence of admm in nonconvex nonsmooth optimization. *Journal of Scientific Computing*, 78:29–63, 2018.

[149] Y.-X. Yuan. Conditions for convergence of trust region algorithms for nonsmooth optimization. *Math. Program.*, 1(31):220—228, 1985. doi: 10.1007/BF02591750.

[150] A. Yurtsever, M. Udell, J. A. Tropp, and V. Cevher. Sketchy Decisions: Convex Low-Rank Matrix Optimization with Optimal Storage. *ArXiv e-prints*, February 2017.

[151] Cun-Hui Zhang et al. Nearly unbiased variable selection under minimax concave penalty. *Ann. Stat.*, 38(2):894–942, 2010. doi: 10.1214/09-AOS729.

[152] Peng Zheng and Aleksandr Aravkin. Relax-and-split method for nonconvex inverse problems. *Inverse Problems*, 36(9):095013, 2020. doi: 10.1088/1361-6420/aba417.

[153] Peng Zheng, Travis Askham, Steven L Brunton, J Nathan Kutz, and Aleksandr Y Aravkin. A unified framework for sparse relaxed regularized regression: SR3. *IEEE Access*, 7:1404–1423, 2018. doi: 10.1109/ACCESS.2018.2886528.