DYNAMIC COMPRESSIVE SENSING: SPARSE RECOVERY ALGORITHMS FOR STREAMING SIGNALS AND VIDEO

A Dissertation
Presented to
The Academic Faculty

by

Muhammad Salman Asif

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Electrical and Computer Engineering

Georgia Institute of Technology
August 2013

DYNAMIC COMPRESSIVE SENSING: SPARSE RECOVERY ALGORITHMS FOR STREAMING SIGNALS AND VIDEO

Approved by:

Professor Justin Romberg, Advisor
School of Electrical and Computer Engineering
Georgia Institute of Technology

Professor James McClellan
School of Electrical and Computer Engineering
Georgia Institute of Technology

Professor Mark Davenport
School of Electrical and Computer Engineering
Georgia Institute of Technology

Professor Sung Ha Kang
School of Mathematics
Georgia Institute of Technology

Professor Patricio Vela
School of Electrical and Computer Engineering
Georgia Institute of Technology

Date Approved: June 20, 2013

Dedicated to my parents.

ACKNOWLEDGEMENTS

While not exactly getting as far as to stand on the proverbial shoulders of the giants, I had the good fortune to look at the world while standing in a close neighborhood of the said giants. In that process I got inspired and motivated to work on many interesting problems for which I received the guidance and the encouragement from people with whom I worked with and learned from. I have also been fortunate to receive tremendous support and encouragement from the friends and family during my graduate studies. I thank you all from the depth of my heart for everything.

The biggest influence on the work in this thesis, and my research and thinking process in general, has been from my advisor Justin Romberg. He has been an inspiring figure and a role model for me since I came to Georgia Tech and attended his first seminar there. Everything in this thesis is a direct result of preliminary ideas from Justin, our long discussions on the white board, and the offhand comments he made at times when I went to his office (almost always unannounced) to discuss a problem. In this process I learned tremendously from Justin on how to describe, analyze, or write a problem (and its solution), and the wealth of knowledge he exposed me to is my prized possession. It has been an absolute pleasure and honor to be his student, and I am immensely grateful to him for everything.

I would like to thank all the members in my thesis committee for reading my thesis, exchanging several ideas, and providing constructive feedback. I would also like to express my sincere gratitude to all the professors at Georgia Tech from whom I learned about mathematics, engineering, writing, and much more. Many thanks to my teachers from earlier education institutions, who inspired me to ask questions and pursue higher education.

I want to thank Petros Boufounos for providing me the opportunity to work as an intern at MERL in Cambridge, where I spent one of my best Summers. Many thanks to Felix Fernandes for providing me the opportunity to work as an intern in Samsung research laboratories in Dallas for one Summer, which was an amazing experience.

I thoroughly enjoyed the companionship of my fellow students in Justin's (extended) group, Adam, Aditya, Ali, Aurèle, Chris, Darryl, Han, Steve, and William. Thank you all; I will dearly miss our long, multi-topic discussions. A very special thanks to my closest friend (also in terms of distance) for a long time, Umair, who had to suffer from my dozen or so alarms every morning, evening, etc. Thanks *patti* for all the good and spoiled times. Farasat and Omer, I cherish your friendship and appreciate the support and affection you have always offered me and also that I have at times extracted from you. Many thanks to Ehsanullah for being there whenever I had to pour my heart out to someone.

The biggest thank you goes to all my family members: My mother and father, who always trusted in me and allowed me to come this far in the pursuit of my dreams; *Ammi* and *Abu*, it is only because of your encouragement, love, and prayers that I could do this or anything else. Many thanks to my sisters, brother, and brothers-in-law for their consistent love and support. I would also like to thank my in-laws for their support and prayers. I owe a debt of gratitude to my wife, Wajiha, for her patient support, much-needed distractions, and unending love. The years spent with her have been filled with joy and happiness, the biggest of all is our son Noraiz who was born while I was working on the final parts of this thesis. Thank you *b*v!

Finally, I would like to acknowledge and express my gratitude for the financial support from the Higher Education Commission of Pakistan during my graduate studies. It was indeed a generous award from the homeland that gave me so much more and that is now, more than ever, in need of a return on such investments and a relief from so many miseries. In the words of Faiz, *Hum dekhain gey* (we shall see).

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

This thesis presents compressive sensing algorithms that utilize system dynamics in the sparse signal recovery process. These dynamics may arise due to a time-varying signal, streaming measurements, or an adaptive signal transform. Compressive sensing theory has shown that under certain conditions, a sparse signal can be recovered from a small number of linear, incoherent measurements. The recovery algorithms, however, for the most part are static: they focus on finding the solution for a fixed set of measurements, assuming a fixed (sparse) structure of the signal.

In this thesis, we present a suite of sparse recovery algorithms that cater to various dynamical settings. The main contributions of this research can be classified into the following two categories: 1) Efficient algorithms for fast updating of ℓ_1-norm minimization problems in dynamical settings. 2) Efficient modeling of the signal dynamics to improve the reconstruction quality; in particular, we use inter-frame motion in videos to improve their reconstruction from compressed measurements.

Dynamic ℓ_1 updating: We present homotopy-based algorithms for quickly updating the solution for various ℓ_1 problems whenever the system changes slightly. Our objective is to avoid solving an ℓ_1-norm minimization program from scratch; instead, we use information from an already solved ℓ_1 problem to quickly update the solution for a modified system. Our proposed updating schemes can incorporate time-varying signals, streaming measurements, iterative reweighting, and data-adaptive transforms. Classical signal processing methods, such as recursive least squares and the Kalman filters provide solutions for similar problems in the least squares framework, where each solution update requires a simple low-rank update. We use *homotopy* continuation for updating ℓ_1 problems, which requires a series of rank-one updates along the

so-called homotopy path.

Dynamic models in video: We present a compressive-sensing based framework for the recovery of a video sequence from incomplete, non-adaptive measurements. We use a linear dynamical system to describe the measurements and the temporal variations of the video sequence, where adjacent images are related to each other via inter-frame motion. Our goal is to recover a quality video sequence from the available set of compressed measurements, for which we exploit the spatial structure using sparse representations of individual images in a spatial transform and the temporal structure, exhibited by dependencies among neighboring images, using inter-frame motion. We discuss two problems in this work: low-complexity video compression and accelerated dynamic MRI. Even though the processes for recording compressed measurements are quite different in these two problems, the procedure for reconstructing the videos is very similar.

CHAPTER I

INTRODUCTION

Recent advances in technology have enhanced our ability to sample, analyze, store, and transmit more information, both in terms of quantity and variety. For instance, a human body can be imaged in high detail and quality using computational tomography scans or magnetic resonance imaging, high-dynamic range and high-speed images and videos can be produced using computational photography, and high-definition videos can be stored or broadcast on hand-held devices. Several factors have contributed to these phenomena over decades: innovations in sensor technology, improvements in the circuits and hardware design, high-performance computing resources, sophisticated signal analysis and elaborate compression schemes, and as always the demand for better and bigger pictures.

In many cases, sensing and imaging devices record indirect measurements of the desired signals. The recovery of the signal in turn requires solving an inverse problem, and any knowledge about the signal structure helps in the solution. In an increasing number of applications, the measurements available for the signal of interest are highly incomplete, and the resulting inverse problems become ill-posed. What makes signal recovery possible in such cases is the use of some prior knowledge that the underlying signal exhibits a simple, low-dimensional structure. In this regard, recent work in compressive sensing and related areas has raised and addressed fundamental questions concerning the amount and the content of measurements that are sufficient for signal recovery, the representative signal models that are most suitable, and the recovery methods that are fast, accurate, and robust [33, 34, 37, 55].

The work presented in this thesis is motivated by similar questions on the recovery of structured, sparse signals from linear measurements, where compressive sensing ideas are applied to various *dynamic* settings. The dynamics may arise because of time-varying signals, streaming measurements, or data-adaptive transforms. The overall theme of this work is to efficiently incorporate these system dynamics in the sparse signal recovery algorithms. The main contributions of this work can be classified into the following two categories: 1) Algorithms for fast updating of ℓ_1-norm minimization based sparse recovery problems in dynamical settings. 2) Efficient modeling of signal dynamics for improving the reconstruction quality; in particular, motion-adaptive models for reconstructing videos from compressed measurements.

1.1 Dynamic ℓ_1 updating

The first part of this thesis details our contributions to the dynamic updating of ℓ_1-norm minimization programs. Solving an ℓ_1-norm minimization problem, such as basis pursuit [46], basis pursuit denoising [46], LASSO [131] and the Dantzig selector [34], has been shown to be an effective way to recover a sparse vector in many different contexts. Most of the existing schemes for solving such ℓ_1 problems are static in nature as they focus on finding the solution for a fixed set of measurements, assuming a fixed (sparse) representation of the signals. However, the same representation and reconstruction formulation is not readily applicable for a streaming system in which the signal changes over time; instead of measuring the entire signal or processing the entire set of measurements at once, these tasks are performed sequentially over short time intervals. Our objective for dynamic ℓ_1 updating is to avoid solving an ℓ_1 problem from scratch every time the system changes; instead, we want to utilize information about the current signal estimate to quickly update the solution. In this regard, we have developed a suite of homotopy algorithms for updating ℓ_1 problems

in a variety of dynamical settings, such as time-varying signals, sequential measurements, iterative reweighting in ℓ_1 norm, and streaming signals that follow a linear dynamic model.

Homotopy methods provide a general framework to solve an optimization program by continuously transforming it into a related problem for which the solution is either available or easy to compute. Starting from an available solution, a series of simple problems are solved along the so-called *homotopy path* towards the desired solution for the original problem [59, 106, 138]. The progression along the homotopy path is controlled by a transformation parameter called the *homotopy parameter*, usually varied between 0 and 1, which correspond to the two end points of the homotopy path.

The homotopy algorithms presented in this thesis update the solution for an ℓ_1 problem by first transforming it into a problem for which the solution is a known (warm-start) vector, and then following a piecewise linear path toward the final solution (for the target ℓ_1 problem). The warm-start vector can be a solution of a related ℓ_1 problem or an arbitrary vector believed to be close to the final solution, whereas updating the solution along the piecewise linear path requires a series of rank-one updates. The underlying assumption is that since small changes in the system cause small changes in the solution, updating a warm-start vector would require only a small number of (computationally simple) low-rank updates. By comparison, classical signal processing methods such as recursive least squares and the Kalman filter update solutions of similar problems in the least squares framework and they require a single low-rank update [77, 78].

Part 1 of the thesis is organized as follows. We begin with the discussion of a general ℓ_1 homotopy algorithm in Chapter 3. This algorithm provides a unified solution for dynamic ℓ_1 updating as it can update the ℓ_1 problem in various dynamical settings. In Chapter 4, we discuss dynamic ℓ_1 updating when the sparse signal changes

while the measurement matrix remains fixed [14]. In Chapter 5, we discuss dynamic ℓ_1 updating as measurements of a fixed signal are sequentially added to the system. In Chapter 6, we discuss dynamic ℓ_1 updating of weights in the ℓ_1 problem and an algorithm to adaptively select weights while solving a sparse recovery problem from scratch [12]. In Chapter 7, we discuss dynamic ℓ_1 updating for a streaming system in which measurements, sparse signal, and sparse representation change over time and the streaming signal is iteratively estimated over short, sliding interval [13]. In Chapter 8, we discuss dynamic ℓ_1 updating for a streaming signal that varies according to a linear dynamic model. This work is in the spirit of an ℓ_1-regularized Kalman filter [13]. In Chapter 9, we discuss how a simple change in the ℓ_1-homotopy algorithm imposes the positivity constraints on the solution of the ℓ_1 programs. In Chapter 10, we discuss a dynamic updating algorithm for the Dantzig selector, where we apply similar principles of ℓ_1-homotopy to the primal and dual formulations of the Dantzig selector program. In Chapter 11, we discuss dynamic ℓ_1 updating for streaming measurements in the context of decoding by linear programming [38], where we want to recover an arbitrary signal from *coded* measurements that are corrupted by sparse errors.

1.2 Dynamic model in video

The second part of this thesis details our work on the recovery of video sequences from compressed, non-adaptive measurements. We use a linear dynamical system to describe the measurements and the temporal variation of the video sequence, where adjacent images are related to each other via inter-frame motion. Compressive sensing theory suggests that a (structured) sparse signal can be recovered from a small number of (non-adaptive) measurements. This combined acquisition and compression lessens the burden on sensing devices in the following ways: Full signal acquisition is

not necessary, additional compression is not required, and the computationally expensive task of reconstruction is performed at the decoder. These features inspired our work on the following two problems: 1) Low-complexity video compression (Chapter 12). 2) Accelerated dynamic magnetic resonance imaging (MRI) (Chapter 13). Even though the motivation and processes for recording compressed measurements are quite different in these two problems, the procedure for reconstructing the videos is very similar. Our goal is to recover a quality video sequence from the available set of compressed measurements, for which we exploit the spatial structure using sparse representation of individual images in a spatial transform and the temporal structure, exhibited by dependencies among neighboring images, using inter-frame motion.

1.2.1 Low-complexity video compression

Certain tasks in the standard video compression are either computationally expensive or infeasible for low-power devices and distributed systems. The main computational complexity in standardized video coding arises from motion estimation and transform coding blocks. To reduce the encoder complexity, we eliminated these blocks from the encoder. Our proposed encoder compresses a video sequence by recording a small number of linear, non-adaptive measurements for each image. On the decoder side, we exploit the sparsity of individual images in the wavelet representation and the motion between neighboring images in the video sequence to improve its reconstruction. We solve an optimization program that regularizes the spatial sparsity of individual images and motion-compensated differences between neighboring images. Since motion information is not readily available at the decoder, we alternate between learning motion from estimated images and using that motion information to improve image estimates. We demonstrate with an extensive set of experiments that motion-compensated regularization yields better results than frame-by-frame and

frame-difference regularized reconstructions. A comparison with some existing low-complexity encoders shows that the motion-regularized decoder outperforms methods that do not use any motion information at the decoder.

1.2.2 Accelerated dynamic MRI

In dynamic MRI, full signal acquisition is often impossible, and only a small number of the so-called k-space measurements can be acquired in a short time. Slow imaging speed in MRI poses particular challenges for dynamic cardiac imaging, in which images are often acquired during a patient's breath-holds. A complete cardiac cycle is usually recorded as a sequence of 16 to 20 images while only a small portion of the Fourier transform of each image is recorded per heartbeat. Thus, to reduce scan time, the acquisition process is accelerated by undersampling the k-space (i.e., 2-D Fourier coefficients). In our work, we treat the images in an MRI sequence as frames of a video sequence, for which undersampled Fourier measurements are provided. The recovery algorithm solves an optimization problem that involves cost functions for data mismatch, spatial sparsity of each image, and temporal sparsity of motion-compensated residuals. We alternate between estimating MR images using an estimate of the inter-frame motion and estimating inter-frame motion using an estimate of MR images. We demonstrate that, using motion-compensated regularization in the reconstruction, it is feasible to recover high-quality dynamic cardiac images with up to ten-fold acceleration [5].

1.3 Organization

We present a brief background on compressive sensing and sparse recovery algorithms in Chapter 2, where we also present a derivation of the homotopy algorithm for LASSO. In Part 1 of the thesis, Chapters 3–11, we discuss algorithms for dynamic updating of several ℓ_1-norm minimization problems. In Part 2 of the thesis, we discus low-complexity video coding in Chapter 12 and accelerated dynamic MRI in

Chapter 13. We provide conclusions of this work and a discussion on future directions of study in Chapter 14.

CHAPTER II

BACKGROUND

Modern digital sampling technology has enabled the acquisition, transmission, and storage of ever increasing number and variety of physical signals [124, 136]. Despite significant advances in data sampling and processing systems, in many cases, current systems are severely constrained and provide incomplete samples of the signal. The constraints can be due to limited sampling rates of sensors, the time required for complete sampling, the number and the cost of desired sensors, or the power consumption in battery-operated devices. Once the signal is fully sampled, conventional data compression schemes indeed lessen the burden on transmission and storage requirements. Compression schemes are usually lossy as they retain a small number of transform coefficients and throw away the rest without causing a significant loss in the perceptual quality of signal, which raises an important question: since we throw away a large portion of the acquired data with little impact on the signal quality, is it possible to recover the desired signal by acquiring reduced amount of data in the first place? Compressive sensing theory suggests that under certain conditions it is indeed possible [35, 37, 38, 55].

2.1 Sampling and compression

Classical sampling theory, largely based on the pioneering work of Whittaker, Nyquist, and Shannon, views a signal as a member of a vector space in which the signal can be represented as a linear combination of the basis elements of the ambient vector space [103, 123, 145]. The coefficients in the linear combination constitute the samples of the signal with respect to the given basis representation. For instance, bandlimited, continuous-time signals belong to a vector space that admits uniformly-shifted

sinc functions as its basis, and the uniformly-spaced discrete samples, at or above the Nyquist rate (i.e., twice the signal bandwidth), constitute coefficients for the sinc basis functions [94]. The continuous-time signal can be reconstructed from discrete samples by passing them through an ideal low-pass filter, which is equivalent to performing a linear sinc interpolation. More general cases with less trivial acquisition and reconstruction components can be found in [94, 136]. Nevertheless, to reconstruct an arbitrary signal in a vector space, we require all the coefficients for the representative basis.

Compression also plays a fundamental role in modern signal processing and also relies on sparse representations of signals in different bases. The fundamental principle behind any compression scheme is that most natural signals contain redundant information, and they can be represented using a small number of coefficients in a representation basis. For instance, JPEG compression standards represent images using a small number of discrete cosine or wavelet coefficients [130]; MPEG and H.264 standards represent video sequences using a reference frames and small residuals of motion compensation [115]. Efficient (sparse) signal representation, therefore, plays a central role in compression: a small subset of significant coefficients represents the entire signal with very little or no loss in the signal quality.

Over the last two decades, a number of orthogonal bases and tight frames have been designed for sparse representation of various signals of interest [31, 36, 50, 53, 141]. In addition to compression, sparse representation also plays a fundamental role in signal denoising, estimation, analysis, and sampling [58, 89, 94, 104]. However, a sparse representation model is non-linear as the significant coefficients change from signal to signal. Therefore, we cannot directly sample the significant coefficients without a prior knowledge of their locations.

2.2 Compressive sensing

Compressive sensing (CS) theory showed how compression can be combined with sampling. In contrast with the conventional method of compression after sampling the entire signal, CS provides a framework in which a structured, sparse signal can be reliably reconstructed from a small number of linear, non-adaptive measurements [32, 35, 37, 38, 55, 57]. The number of measurements that is required for the signal reconstruction depends on the design of measurements and the sparse structure of the signal, but the number can be much lower than the Shannon-Nyquist sampling limit [61, 133, 142]. However, the reconstruction process is non-linear and requires solving an underdetermined system of equations. CS assumes a sparse or low-dimensional model on the signals during reconstruction, which restricts the signal to a small subset of the vector space; for instance, a *union of subspaces* inside the ambient vector space. A low-dimensional signal model limits the *degrees of freedom* in a signal, hence makes it possible to recover the signal from a small number of measurements [16, 24, 30, 87].

2.2.1 System model

Consider the following linear observation model:

$$y = \Phi\bar{x} + e, \tag{2.1}$$

where Φ denotes an $M \times N$ measurement matrix, $\bar{x}$ denotes the unknown signal of interest, y denotes a set of measurements, and e denotes noise in the system. The measurements in (2.1) can be viewed as generalized sampling, where the rows in Φ represent the sampling functions. For example, shifted sinc functions generate conventional Nyquist samples of $\bar{x}$, or sinusoids with different frequencies yield Fourier coefficients. In the CS framework, $M \ll N$ and the system in (2.1) is highly underdetermined. Since Φ has an $(N - M)$-dimensional null-space, it is not possible to recover an arbitrary true signal $\bar{x}$ from the infinitely many possible solutions of the

system (2.1). However, the situation changes altogether if $\bar{x}$ is sparse, or if it has sparse representation in some basis; for instance, $\bar{x} = \Psi\bar{\alpha}$, where $\bar{\alpha}$ is a sparse vector and Ψ is the representation basis.

2.2.2 Sparsity and restricted isometry

Suppose $\Psi = I$ and $\bar{x}$ contains only S nonzero elements at locations indexed by the support set Γ. We can estimate $\bar{x}$ as

$$\widehat{x} = \begin{cases} (\Phi_\Gamma^T \Phi_\Gamma)^{-1} \Phi_\Gamma^T y & \text{on } \Gamma \\ 0 & \text{otherwise,} \end{cases} \tag{2.2}$$

whenever Φ_Γ is full-rank. Φ_Γ denotes a matrix constructed from S columns in Φ at locations Γ. Thus, a necessary condition to recover an S-sparse signal (even with known support) is that $M \geq S$. Furthermore, in order to recover an arbitrary S-sparse signal, Φ_Γ needs to be full rank for all possible supports Γ. The well-known restricted isometry property (RIP) is commonly used to analyze recovery guarantees for such a case when the support of signal is unknown. A matrix Φ satisfies RIP of order $2S$ if there exists a constant $\delta_{2S} < 1$ such that

$$(1 - \delta_{2S})\|x\|_2^2 \leq \|\Phi x\|_2^2 \leq (1 + \delta_{2S})\|x\|_2^2, \tag{2.3}$$

for all $2S$-sparse vectors x [38]. A small RIP constant implies that every submatrix Φ_Γ is "near-orthogonal" or "near-isometric"—as the name suggests. If the RIP holds for a given matrix Φ with a small constant δ_{2S}, any S-sparse signal x can be stably reconstructed from the underdetermined system in (2.1) [33, 39]. For instance, matrices with i.i.d. random Gaussian or Bernoulli entries are known to satisfy the RIP when $M = O(S \log N)$ [15, 35, 39]. The additional $\log N$ factor, in some sense, is the cost we pay for not knowing the support of the signal in advance.

2.3 Sparse signal recovery

In the absence of noise, a natural way to find the sparsest vector x that satisfies measurements in (2.1) is by solving the following optimization problem:

$$\underset{x}{\text{minimize}} \ \|x\|_0 \quad \text{subject to} \quad \Phi x = y, \tag{2.4}$$

where $\|x\|_0$ is the so-called ℓ_0 quasi-norm, which provides the number of nonzero elements in x. Unfortunately, (2.4) is computationally intractable and known to be NP-hard [100]. An effective class of recovery problems relaxes the ℓ_0 norm with the ℓ_1 norm. The well-known basis pursuit problem [46, 121] takes the following form:

$$\underset{x}{\text{minimize}} \ \|x\|_1 \quad \text{subject to} \quad \Phi x = y, \tag{2.5}$$

where $\|x\|_1 = \sum_{i=1}^{N} |x_i|$ denotes the ℓ_1 norm of x. The optimization problem in (2.5) can be recast as a linear program and solved using a number of efficient solvers [26, 41, 102]. It is well known that if Φ satisfies RIP with a small constant, any S-sparse signal can be exactly recovered by solving (2.5) [37, 38].

In practical settings, when the measurements in (2.1) usually contain noise, the equality constraints can be replaced with a data-fidelity term. Two commonly used convex programs for this purpose are 1) least absolute shrinkage and selection operator (LASSO) [131] or basis pursuit denoising (BPDN) [46], and 2) the Dantzig Selector (DS) [34]. The LASSO/BPDN solves

$$\underset{x}{\text{minimize}} \ \tau\|x\|_1 + \frac{1}{2}\|\Phi x - y\|_2^2, \tag{2.6}$$

and the DS solves

$$\underset{x}{\text{minimize}} \ \|x\|_1 \quad \text{subject to} \quad \|\Phi^T(\Phi x - y)\|_\infty \leq \tau, \tag{2.7}$$

where $\tau > 0$ is a threshold parameter, which can be selected to control the tradeoff between the sparsity of the solution and its fidelity to the measurements. In recent

years, a number of efficient schemes have been developed for solving these problems [19, 20, 41, 59, 62, 137, 144, 150]. A number of performance guarantees are associated with both (2.6) and (2.7) [3, 33, 34, 38, 56, 98]. Furthermore, a small RIP constant provides robustness guarantees for the recovery of signals that are not exactly sparse but can be well-approximated by a sparse signal [32, 33].

The optimization problems in (2.5), (2.6), and (2.7) assume that x is sparse. However, we can easily incorporate the general case when x itself is not sparse but it has a sparse representation in a basis Ψ; for example, $\Psi^T x$ is sparse. The only difference is that instead of $\|x\|_1$, we would use $\|\Psi^T x\|_1$ in the optimization problems [40, 60].

In addition to the above mentioned ℓ_1 problems, a number of greedy and model-based algorithms are also available for the sparse signal recovery [16, 25, 101, 109, 134].

2.3.1 Homotopy for LASSO

The dynamic ℓ_1 updating schemes we present in Part 1 of the thesis are based on homotopy continuation. To familiarize the reader with homotopy principles, below we present the well-known LASSO/LARS homotopy algorithm that solves the problem in (2.6) [59, 106]. Similar homotopy algorithms also exist for (2.7) [10, 73].

The LASSO/LARS homotopy algorithm solves (2.6) for a desired value of τ by tracing the entire solution path for a range of decreasing values of τ (i.e., any point on the so-called homotopy path is a solution of (2.6) for a certain value of τ) [59, 106]. Starting with a large value of τ, LASSO homotopy shrinks τ toward its final value in a sequence of computationally inexpensive steps. The fundamental insight is that as τ changes, the solution of (2.6) follows a piecewise-linear path in which the length and the direction of each segment is completely determined by the support and the sign sequence of the solution on that segment. This fact can be derived by analyzing the optimality conditions for (2.6), as given below in (2.8) [54, 64]. The support of

the solution changes only at certain critical values of τ, when either a new nonzero element enters the support or an existing nonzero element shrinks to zero. These critical values of τ are easy to calculate at any point along the homotopy path. For every homotopy step, we jump from one critical value of τ to the next while updating the support of the solution, until τ has been lowered to its desired value.

In every homotopy step, the update direction and the step-size for moving to a smaller critical value of τ can be easily calculated using certain optimality conditions, which can be derived using the subdifferential of the objective in (2.6) [26, 54, 64]. At any given value of τ, the solution x^* for (2.6) must satisfy the following optimality conditions:

$$\Phi_\Gamma^T(\Phi x^* - y) = -\tau z \tag{2.8a}$$

$$\|\Phi_{\Gamma^c}^T(\Phi x^* - y)\|_\infty \leq \tau, \tag{2.8b}$$

where Γ denotes the support[1] of x^*, z denotes the sign sequence of x^* on Γ, and Φ_Γ denotes a matrix with columns of Φ at indices in the set Γ.

Let us denote the objective function in (2.6) as $f(x)$, which is convex but not differentiable everywhere. A necessary condition for a vector x^* to be a minimizer of $f(x)$ is that the *subdifferential* of f at x^* must contain the zero vector, which is denoted as $0 \in \partial f(x^*)$ [116]. If f is convex and differentiable, then its subdifferential at x is same as the gradient. The subdifferential of a convex function at a point x, where it is non-differentiable, is defined as the set of all *subgradients* of the function at that point. A vector $g \in \mathbb{R}^N$ is a subgradient of $f : \mathbb{R}^N \to R$ at $x \in \mathbb{R}^N$ if for all $u \in \mathbb{R}^N$,

$$f(u) \geq f(x) + g^T(u - x), \tag{2.9}$$

which means that the affine function (of u) $f(x) + g^T(u - x)$ remains below the graph

[1] We use the terms support and active set interchangeably for the index set of nonzero coefficients.

of f for any u [21, 26]. We calculate the subdifferential of f as

$$\partial f(x) = \tau \partial \|x\|_1 + \Phi^T(\Phi x - y), \tag{2.10}$$

where $\partial\|x\|_1$ denotes the subdifferential of the ℓ_1 norm that can be described as

$$\partial\|x\|_1 = \left\{ g \in \mathbb{R}^N \;\middle|\; \begin{array}{ll} g_i = +1, & x_i > 0 \\ g_i = -1, & x_i < 0 \\ g_i \in [-1, 1], & x_i = 0 \end{array} \right\}. \tag{2.11}$$

This implies that $\partial\|x\|_1$ is uniquely defined for the nonzero entries in x as the sign sequence, while it can have any value in $[-1, 1]$ for the zero entries in x. Using the subdifferential $g = \partial\|x^*\|_1$ in (2.10), we can describe the optimality condition, $0 \in \partial f(x^*)$, for a vector x^* as

$$\tau g + \Phi^T(\Phi x^* - y) = 0, \quad \|g\|_\infty \leq 1, g^T x^* = \|x^*\|_1. \tag{2.12}$$

Thus, a vector x^* with support Γ and sign sequence z, yields the optimality conditions described in (2.8).

The optimality conditions in (2.8) can be viewed as N constraints that the solution x^* needs to satisfy with equality on the active set Γ and inequality elsewhere. (The only exception is at the critical values of τ when the support changes and the constraint on the incoming or the outgoing index holds with equality.) As we reduce τ to $\tau - \delta$, for a small value of δ, the solution moves in a direction ∂x, which to maintain optimality must obey

$$\Phi_\Gamma^T(\Phi x^* - y) + \delta \Phi_\Gamma^T \Phi \partial x = -(\tau - \delta) z \tag{2.13a}$$

$$\| \underbrace{\Phi^T(\Phi x^* - y)}_{p} + \delta \underbrace{\Phi^T \Phi \partial x}_{d} \|_\infty \leq (\tau - \delta). \tag{2.13b}$$

The update direction that keeps the solution optimal as we change δ can be written as

$$\partial x = \begin{cases} (\Phi_\Gamma^T \Phi_\Gamma)^{-1} z & \text{on } \Gamma \\ 0 & \text{otherwise.} \end{cases} \tag{2.14}$$

We can move in direction ∂x until either one of the constraints in (2.13b) is violated, indicating that we must add an element to the support Γ, or one of the nonzero elements in x^* shrinks to zero, indicating that we must remove an element from Γ. The smallest step-size that causes one of these changes in the support can be easily computed as $\delta^* = \min(\delta^+, \delta^-)$, where

$$\delta^+ = \min_{i \in \Gamma^c} \left(\frac{\tau - p_i}{1 + d_i}, \frac{-\tau - p_i}{-1 + d_i} \right)_+ \tag{2.15a}$$

and

$$\delta^- = \min_{i \in \Gamma} \left(\frac{-x_i^*}{\partial x_i} \right)_+, \tag{2.15b}$$

and $\min(\cdot)_+$ means that the minimum is taken over only positive arguments. δ^+ is the smallest step-size that causes an inactive constraint to become active at index γ^+, indicating that γ^+ should enter the support, and δ^- is the smallest step-size that shrinks an existing element at index γ^- to zero. The new critical value of τ becomes $\tau - \delta^*$, the new signal estimate x^* becomes $x^* + \delta^* \partial x$, and the support and the sign sequence change accordingly. At every homotopy step, we compute the update direction and the step-size that cause one-element change in the support. We repeat this procedure until τ has been lowered to its desired value.

The main computational cost of every homotopy step comes from computing ∂x by solving an $S \times S$ system of equations in (2.14) (where S denotes the size of the support Γ) and from computing the vector d in (2.8b), which is used to compute the step-size δ in (2.15). Since we know the values of d on Γ by construction and ∂x is nonzero only on Γ, the cost for computing d is same as one application of an $M \times N$ matrix. Moreover, since Γ changes by a single element at every homotopy step, instead of solving the linear system in (2.14) from scratch, we can efficiently compute ∂x using a rank-one update at every step [23, 66].

2.3.2 Homotopy for ℓ_1 updating

The homotopy method described above solves (2.6); the homotopy procedure starts with a zero vector and builds the solution by reducing τ while adding or removing

an element in the support at every homotopy step. In Part 1 of this thesis, we present a suite of homotopy algorithms that dynamically update solutions for ℓ_1-norm minimization problems similar to (2.6); the homotopy procedure for that starts with a nonzero (warm-start) vector, believed to be close to the desired solution, and updates the solution in a sequence of similar homotopy steps. For instance, consider the following optimization problem for which the solution at $\epsilon = 0$ is known in advance:

$$\text{minimize } \tau\|x\|_1 + \frac{1}{2}\|\Phi x - (1-\epsilon)y - \epsilon\tilde{y}\|_2^2. \tag{2.16}$$

We fix τ and build the homotopy using ϵ. Note that by changing ϵ from 0 to 1 in (2.16), we gradually replace y with $\tilde{y}$. In its present form, (2.16) is equivalent to (2.6) at $\epsilon = 0$, and it can be viewed as a method for updating the solution of (2.6) if the sparse signal changes slightly and modified measurements are received. We demonstrate in the next few chapters that as ϵ changes, the solution of (2.16) also follows a piecewise linear path that can be easily traced using homotopy steps similar to those described in Section 2.3.1. We discuss details of the homotopy algorithm for (2.16) and a few similar problems along with their applications in Chapters 3–11.

PART I

Dynamic ℓ_1 updating

CHAPTER III

DYNAMIC SIGNAL RECOVERY USING ℓ_1 HOMOTOPY

Most of the existing sparse recovery methods are *static* in nature in which we usually assume that the unknown signal is a fixed vector for which a fixed set of linear measurements are available. To reconstruct the signal, we assume that the signal has a sparse representation in a known signal transformation and solve an ℓ_1-norm minimization problem, which encourages the solution to be sparse while maintaining fidelity toward the measurements [34, 46]. In this chapter, we discuss *dynamic* updating of ℓ_1-norm minimization programs for sparse signal recovery problems in which the unknown signal, linear measurements, the representation basis, or some other system parameter may change over time. Our primary objective is to avoid solving a new ℓ_1 problem from scratch any time a change occurs in the system; instead, we want to utilize any available information about the signal dynamics to expedite the recovery process. In this regard, we propose a general homotopy algorithm that quickly updates the solution of an ℓ_1 problem as the system changes. Our proposed homotopy algorithm accepts a warm-start vector, expected to be near the desired solution, and updates the solution in a sequence of computationally inexpensive steps. The solution path of our homotopy algorithm is piecewise linear, which can be traced in a sequence of simple steps [59, 106, 131].

3.1 Problem formulation and motivation

Consider the following time-varying linear system:

$$y_t = \Phi_t x_t + e_t, \tag{3.1}$$

where x_t is an unknown signal of interest, y_t is a vector that contains measurements of x_t, Φ_t is a measurement matrix, and e_t is noise in the measurements. Suppose we represent x_t as $\Psi_t \alpha_t$ and the equivalent system for (3.1) as

$$y_t = \Phi_t \Psi_t \alpha_t + e_t, \tag{3.2}$$

where Ψ_t denotes a sparse representation matrix and α_t denotes a sparse vector of transform coefficients. We want to solve the following weighted ℓ_1-norm minimization problem for a sparse estimate of α_t:

$$\underset{\alpha_t}{\text{minimize}} \ \|W_t \alpha_t\|_1 + \frac{1}{2}\|\Phi_t \Psi_t \alpha_t - y_t\|_2^2. \tag{3.3}$$

The ℓ_1 term promotes sparsity in the estimated coefficients; W_t is a diagonal matrix of positive weights that can be adapted to promote a certain sparse structure in the solution [42, 152]; and the ℓ_2 term ensures that the solution remains close to the measurements. The optimization problem in (3.3) is convex and can be solved using a variety of solvers [18, 19, 26, 147, 148].

In dynamic ℓ_1 updating, we are interested in quickly updating the solution of the ℓ_1 problem in (3.3) as the system parameters $(y_t, \Phi_t, \Psi_t, W_t)$ change, independently or simultaneously. For instance, suppose we have solved (3.3) for a given system and one of the following two situations arises: 1) We add a new set of measurements to the system, which implies addition of rows to Φ_t and y_t. 2) The underlying signal changes slightly and we receive a new set of measurements in the same system, which implies changes in y_t, while Φ_t remains the same. In both these cases, we want to update the solution without solving the problem in (3.3) from scratch.

A more elaborate example in which all the system parameters change is as follows. Suppose x_t changes according to the following linear dynamic model:

$$x_{t+1} = F_t x_t + f_t, \tag{3.4}$$

where F_t is a prediction matrix that couples x_t and x_{t+1} and f_t is the error in the prediction, which we assume has a bounded ℓ_2 norm. We want to iteratively estimate

$x_t = \Psi_t \alpha_t$ for increasing t by jointly solving the current system of measurements in (3.1) and the dynamic model in (3.4) using the following modified form of (3.3):

$$\underset{\alpha_t}{\text{minimize}} \; \|W_t \alpha_t\|_1 + \frac{1}{2}\|\Phi_t \Psi_t \alpha_t - y_t\|_2^2 + \frac{\lambda_t}{2}\|\widehat{x}_t - \Psi_t \alpha_t\|_2^2, \tag{3.5}$$

where $\lambda_t > 0$ denotes a regularization parameter and $\widehat{x}_t$ denotes a predicted value of x_t based on an estimate of x_{t-1} from the previous iteration (e.g., $\widehat{x}_t = F_{t-1}\widehat{x}_{t-1}$). The problem in (3.5) can be viewed as a "one-step" ℓ_1-regularized Kalman filter as it uses information about the previous signal estimate and the linear dynamics to iteratively update the signal estimate [44, 45, 47, 78]. Note that before solving the optimization problem, an estimate of α_t can be predicted from $\widehat{x}_t$ and used as a warm-start to expedite the solution of (3.5). We can also select the W_t using the available estimate of the α_t (in the same spirit as iterative reweighting [42]). Although the system parameters $(y_t, \Phi_t, \Psi_t, W_t)$ change at every iteration, as t changes, instead of solving (3.5) from scratch, we want to utilize information from previous iterations to expedite the recovery process.

Classical signal processing methods such as recursive least squares (RLS) and the Kalman filter solve similar dynamic updating problems in the least-squares (LS) framework. An attractive feature of the LS methods is that their solutions admit closed form representations and recursive updates can be computed using a low-rank update [70, 77, 78, 126]. However, in their standard form, the LS methods are oblivious to the sparse structure in the signal. In dynamic ℓ_1 updating, our motivation is to develop a methodology similar to the RLS and Kalman updates for updating the solutions of the ℓ_1 problems. Using the homotopy updates, we can sequentially add and remove multiple measurements in the system, update the weights in the ℓ_1 term, and update the solution in a sequence of a small number of computationally inexpensive homotopy steps. Since the homotopy algorithm solves an ℓ_1 problem, its updating scheme is not as simple as that in the RLS and the Kalman filter; but it has the same recursive spirit and updates the solution by reducing the problem into

a series of low-rank updates.

3.2 ℓ_1-homotopy: A unified homotopy algorithm

We present a versatile homotopy algorithm that uses a known (warm-start) vector as a starting point and builds a homotopy towards the desired solution of the ℓ_1 problem in (3.3). The algorithm is not restricted to a specific ℓ_1 updating problem, and it can be adopted in different scenarios with a simple initialization step. For instance, tracking changes in a time-varying signal, adding or removing sequential measurements, updating the weights, or making arbitrary changes in the system matrix. Detailed discussion and experiments for different dynamic ℓ_1 updating problems are discussed in the subsequent chapters of Part 1, where we either use the homotopy algorithm presented in this section or algorithms that are designed using similar homotopy principles.

Suppose $\mathbf{y}$ is a vector that obeys the following linear model: $\mathbf{y} = \mathbf{\Phi}\bar{\mathbf{x}} + \mathbf{e}$, where $\bar{\mathbf{x}}$ is a sparse, unknown signal of interest, $\mathbf{\Phi}$ is an $M \times N$ system matrix, and $\mathbf{e}$ is a noise vector. We want to solve the following ℓ_1-norm minimization problem to recover $\bar{\mathbf{x}}$:

$$\underset{\mathbf{x}}{\text{minimize}} \ \|\mathbf{W}\mathbf{x}\|_1 + \frac{1}{2}\|\mathbf{\Phi}\mathbf{x} - \mathbf{y}\|_2^2, \tag{3.6}$$

where $\mathbf{W}$ is a diagonal matrix that contains positive weights $\mathbf{w}$ on its diagonal. Instead of solving (3.6) from scratch, we want to expedite the process by using some prior knowledge about the solution of (3.6). We assume that we have a sparse vector, $\widehat{\mathbf{x}}$, that is close to the original solution of (3.6) and that has support[1] $\widehat{\Gamma}$ and sign sequence $\widehat{\mathbf{z}}$. Our proposed homotopy algorithm can be initialized with an arbitrary vector $\widehat{\mathbf{x}}$, given the corresponding matrix $\mathbf{\Phi}_{\widehat{\Gamma}}^T \mathbf{\Phi}_{\widehat{\Gamma}}$ is invertible; however, the update will be quick if $\widehat{\mathbf{x}}$ is close to the final solution.

Homotopy methods provide a general framework to solve an optimization program by continuously transforming it into a related problem for which the solution is either

[1] We use the terms support and active set interchangeably for the index set of nonzero coefficients.

available or easy to compute. Starting from an available solution, a series of simple problems are solved along the so-called *homotopy path* towards the final solution of the original problem [59, 106, 138]. The progression along the homotopy path is controlled by the *homotopy parameter*, which usually varies between 0 and 1, corresponding to the two end points of the homotopy path.

We build the homotopy formulation for (3.6), using $\epsilon \in [0, 1]$ as the homotopy parameter, as follows. We treat the given warm-start vector $\widehat{\mathbf{x}}$ as a starting point and solve the following optimization problem:

$$\underset{\mathbf{x}}{\text{minimize}} \ \|\mathbf{W}\mathbf{x}\|_1 + \frac{1}{2}\|\mathbf{\Phi}\mathbf{x} - \mathbf{y}\|_2^2 + (1-\epsilon)\mathbf{u}^T\mathbf{x} \tag{3.7}$$

by changing ϵ from 0 to 1. We define $\mathbf{u}$ as

$$\mathbf{u} \stackrel{\text{def}}{=} -\mathbf{W}\,\widehat{\mathbf{z}} - \mathbf{\Phi}^T(\mathbf{\Phi}\widehat{\mathbf{x}} - \mathbf{y}), \tag{3.8}$$

where $\widehat{\mathbf{z}}$ can be any vector that is defined as $\text{sign}(\widehat{\mathbf{x}})$ on $\widehat{\Gamma}$ and strictly smaller than one elsewhere. Using the definition of $\mathbf{u}$ in (3.8) and the conditions in (3.10) below, we can establish that $\widehat{\mathbf{x}}$ is the optimal solution of (3.7) at $\epsilon = 0$. As ϵ changes from 0 to 1, the optimization problem in (3.7) gradually transforms into the one in (3.6), and the solution of (3.7) follows a piece-wise linear homotopy path from $\widehat{\mathbf{x}}$ toward the solution of (3.6). To demonstrate these facts and derive the homotopy algorithm, we analyze the optimality conditions for (3.7) below.

The optimality conditions for (3.7) can be derived by setting the subdifferential of its objective function to zero [21, 26]. We can describe the conditions that a vector $\mathbf{x}^*$ needs to satisfy to be an optimal solution as

$$\mathbf{W}\mathbf{g} + \mathbf{\Phi}^T(\mathbf{\Phi}\mathbf{x}^* - \mathbf{y}) + (1-\epsilon)\mathbf{u} = 0, \ \ \|\mathbf{g}\|_\infty \leq 1, \ \ \mathbf{g}^T\mathbf{x}^* = \|\mathbf{x}^*\|_1, \tag{3.9}$$

where $\mathbf{g} = \partial\|\mathbf{x}^*\|_1$ denotes the subdifferential of the ℓ_1 norm of $\mathbf{x}^*$ (see our discussion in Section 2.3.1 for further details) [54, 116]. This implies that for any given value of

$\epsilon \in [0,\ 1]$, the solution $\mathbf{x}^*$ for (3.7) must satisfy the following optimality conditions:

$$\phi_i^T(\mathbf{\Phi}\mathbf{x}^* - \mathbf{y}) + (1-\epsilon)\mathbf{u}_i = -\mathbf{w}_i\mathbf{z}_i \qquad \text{for all } i \in \Gamma \tag{3.10a}$$

$$|\phi_i^T(\mathbf{\Phi}\mathbf{x}^* - \mathbf{y}) + (1-\epsilon)\mathbf{u}_i| \leq \mathbf{w}_i \qquad \text{for all } i \in \Gamma^c, \tag{3.10b}$$

where ϕ_i denotes i^{th} column of $\mathbf{\Phi}$, Γ is the support of $\mathbf{x}^*$, and $\mathbf{z}$ is its sign sequence. The optimality conditions in (3.10) can be viewed as N constraints on $\phi_i^T(\mathbf{\Phi}\mathbf{x}-\mathbf{y})+(1-\epsilon)\mathbf{u}_i$ that the solution $\mathbf{x}^*$ needs to satisfy with equality (in terms of the magnitude and the sign) on the active set Γ and strict inequality (in terms of the magnitude) elsewhere. The only exception is at the critical values of ϵ where the support changes and the constraint on the incoming or the outgoing index holds with equality. Equivalently, the locations of the active constraints in (3.10) determine the support of $\mathbf{x}^*$, Γ, and their signs determine the signs of $\mathbf{x}^*$, $\mathbf{z}$, which in our formulation are opposite to the signs of the active constraints. Note that, the definition of $\mathbf{u}$ in (3.8) ensures that $\widehat{\mathbf{x}}$ satisfies the optimality conditions in (3.10) at $\epsilon = 0$; hence, it is a valid initial solution. It is also evident from (3.10a) that at any value of ϵ the solution $\mathbf{x}^*$ is completely described by the support Γ and the sign sequence $\mathbf{z}$ (assuming that $(\mathbf{\Phi}_\Gamma^T\mathbf{\Phi}_\Gamma)^{-1}$ exists). The support changes only at certain critical values of ϵ, when either a new element enters the support or an existing nonzero element shrinks to zero. These critical values of ϵ are easy to calculate at any point along the homotopy path, and the entire path (parameterized by ϵ) can be traced in a sequence of computationally inexpensive homotopy steps.

For every homotopy step we jump from one critical value of ϵ to the next while updating the support of the solution, until ϵ is equal to 1. As we increase ϵ by a small value δ, the solution moves in a direction $\partial\mathbf{x}$, which to maintain optimality must obey

$$\phi_i^T(\mathbf{\Phi}\mathbf{x}^* - \mathbf{y}) + (1-\epsilon)\mathbf{u}_i + \delta(\phi_i^T\mathbf{\Phi}\partial\mathbf{x} - \mathbf{u}_i) = -\mathbf{w}_i\mathbf{z}_i \qquad \text{for all } i \in \Gamma \tag{3.11a}$$

$$|\underbrace{\phi_i^T(\mathbf{\Phi}\mathbf{x}^* - \mathbf{y}) + (1-\epsilon)\mathbf{u}_i}_{\mathbf{p}_i} + \delta\underbrace{(\phi_i^T\mathbf{\Phi}\partial\mathbf{x} - \mathbf{u}_i)}_{\mathbf{d}_i}| \leq \mathbf{w}_i \qquad \text{for all } i \in \Gamma^c. \tag{3.11b}$$

The update direction that keeps the solution optimal as we change δ can be written as

$$\partial \mathbf{x} = \begin{cases} (\mathbf{\Phi}_\Gamma^T \mathbf{\Phi}_\Gamma)^{-1} \mathbf{u}_\Gamma & \text{on } \Gamma \\ 0 & \text{otherwise.} \end{cases} \tag{3.12}$$

We can move in direction $\partial \mathbf{x}$ until either one of the constraints in (3.11b) is violated, indicating that we must add an element to the support Γ, or one of the nonzero elements in $\mathbf{x}^*$ shrinks to zero, indicating that we must remove an element from Γ. The smallest step-size that causes one of these changes in the support can be easily computed as $\delta^* = \min(\delta^+, \delta^-)$, where[2]

$$\delta^+ = \min_{i \in \Gamma^c} \left(\frac{\mathbf{w}_i - \mathbf{p}_i}{\mathbf{d}_i}, \frac{-\mathbf{w}_i - \mathbf{p}_i}{\mathbf{d}_i} \right)_+ \tag{3.13a}$$

and

$$\delta^- = \min_{i \in \Gamma} \left(\frac{-\mathbf{x}_i^*}{\partial \mathbf{x}_i} \right)_+, \tag{3.13b}$$

and $\min(\cdot)_+$ means that the minimum is taken over only positive arguments. δ^+ is the smallest step-size that causes an inactive constraint to become active at index γ^+, indicating that γ^+ should enter the support and $\mathbf{z}_{\gamma^+}$ should be opposite to the sign of the active constraint at γ^+, and δ^- is the smallest step-size that shrinks an existing element at index γ^- to zero, indicating that γ^- should leave the support. The new critical value of ϵ becomes $\epsilon + \delta^*$ and the new signal estimate $\mathbf{x}^*$ becomes $\mathbf{x}^* + \delta^* \partial \mathbf{x}$; the support and the sign sequence are updated accordingly. If γ^+ is added to the support, at the next iteration we check whether the value of $\partial \mathbf{x}_{\gamma^+}$ has the same sign as $\mathbf{z}_{\gamma^+}$; if the signs mismatch, we immediately remove γ^+ from the support and recompute the update direction $\partial \mathbf{x}$.

At every step along the homotopy path, we compute the update direction, the step-size, and the consequent one-element change in the support. We repeat this

[2] To include the positivity constraint in the optimization problem (3.6), we initialize the homotopy with a non-negative (feasible) warm-start vector and define $\delta^+ = \min_{i \in \Gamma^c} \left(\frac{-\mathbf{w}_i - \mathbf{p}_i}{\mathbf{d}_i} \right)_+$. See Chapter 9 for details.

Algorithm 1 ℓ_1-HOMOTOPY

Input: $\mathbf{\Phi}$, $\mathbf{y}$, $\mathbf{W}$, $\widehat{\mathbf{x}}$, and $\mathbf{u}$ (optional: inverse or decomposition factors of $\mathbf{\Phi}_{\widehat{\Gamma}}^T \mathbf{\Phi}_{\widehat{\Gamma}}$)
Output: $\mathbf{x}^*$

Initialize: $\epsilon = 0$, $\mathbf{x}^* \leftarrow \widehat{\mathbf{x}}$
Repeat:
    Compute $\partial\mathbf{x}$ in (3.12) ▷ Update direction
    Compute $\mathbf{p}$ and $\mathbf{d}$ in (3.11b)
    Compute $\delta^* = \min(\delta^+, \delta^-)$ in (3.13) ▷ Step size
    if $\epsilon + \delta^* > 1$ **then**
        $\delta^* \leftarrow 1 - \epsilon$ ▷ Last iteration
        $\mathbf{x}^* \leftarrow \mathbf{x}^* + \delta^* \partial\mathbf{x}$ ▷ Final solution
        `break`
    end if
    $\mathbf{x}^* \leftarrow \mathbf{x}^* + \delta^* \partial\mathbf{x}$ ▷ Update the solution
    $\epsilon \leftarrow \epsilon + \delta^*$ ▷ Update the homotopy parameter
    if $\delta^* = \delta^-$ **then**
        $\Gamma \leftarrow \Gamma \backslash \gamma^-$ ▷ Remove an element from the support
    else
        $\Gamma \leftarrow \Gamma \cup \gamma^+$ ▷ Add a new element to the support
    end if
until $\epsilon = 1$

procedure until ϵ is equal to 1. The pseudocode outlining the homotopy procedure is presented in Algorithm 1.

The main computational cost of every homotopy step comes from computing $\partial\mathbf{x}$ by solving an $S \times S$ system of equations in (3.12) (where S denotes the size of Γ) and from computing the vector $\mathbf{d}$ in (3.10b) that is used to compute the step-size δ in (3.13). Since we know the values of $\mathbf{d}$ on Γ by construction and $\partial\mathbf{x}$ is nonzero only on Γ, the cost for computing $\mathbf{d}$ is same as one application of an $M \times N$ matrix. Moreover, since Γ changes by a single element at every homotopy step, instead of solving the linear system in (3.12) from scratch, we can efficiently compute $\partial\mathbf{x}$ using a rank-one update at every step:

- ▷ *Update matrix inverse:* We can derive a rank-one updating scheme using the matrix inversion lemma and explicitly update the inverse matrix $(\mathbf{\Phi}_\Gamma^T \mathbf{\Phi}_\Gamma)^{-1}$, which has an equivalent cost of performing one matrix-vector product with an

$M \times S$ and an $S \times S$ matrix each and adding a rank-one matrix to $(\mathbf{\Phi}_\Gamma^T \mathbf{\Phi}_\Gamma)^{-1}$. The update direction $\partial \mathbf{x}$ can be recursively computed with a vector addition. The total cost for rank-one update is approximately $MS + 2S^2$ flops.

- *Update matrix factorization:* Updating the inverse of matrix often suffers from numerical stability issues, especially when S becomes closer to M (i.e, the number of columns in $\mathbf{\Phi}_\Gamma$ becomes closer to the number of rows). In general, a more stable approach is to update the Cholesky factorization of $\mathbf{\Phi}_\Gamma^T \mathbf{\Phi}_\Gamma$ (or the QR factorization of $\mathbf{\Phi}_\Gamma$) as the support changes [66, Chapter 12], [23, Chapter 3]. The computational cost for updating Cholesky factors and $\partial \mathbf{x}$ involves nearly $MS + 3S^2$ flops.

As such, the computational cost of a homotopy step is close to the cost of one application of each $\mathbf{\Phi}$ and $\mathbf{\Phi}^T$ (that is, close to $MN + MS + 3S^2 + O(N)$ flops, assuming S elements in the support). If the inverse or factors of $\mathbf{\Phi}_{\widehat{\Gamma}}^T \mathbf{\Phi}_{\widehat{\Gamma}}$ are not readily available during initialization, then updating or computing that would incur an additional one-time cost.

3.3 Discussion

The homotopy algorithm presented in Algorithm 1 extends and unifies previous work by us and others on similar homotopy problems [6, 8, 9, 12, 14, 65, 149]: The algorithm we presented above is not restricted to a particular updating problem, and with a simple initialization (i.e., $\mathbf{u}$), the same algorithm can be used to update the solution for arbitrary changes in the ℓ_1-problem of the form (3.3). For instance, changes in the measurements (y_t) as the signal (x_t) changes with a fixed measurement matrix [8, 14], which is presented in Chapter 4; adding or removing single or multiple measurements [6, 9, 14, 65], which is presented in Chapter 5; changes in the weights (W_t) [12], which is presented in Chapter 6; or arbitrary changes in the measurement

matrix (Φ_t) or the representation matrix (Ψ_t) [9, 149], which is presented in Chapters 7 and 8. Unlike previous approaches, Algorithm 1 does not impose any restriction on the warm-start vector to be a solution of an ℓ_1 problem; as it can accommodate an arbitrary vector to initialize the homotopy update. Of course, the update will be quicker when the initialization (warm-start) vector is close the final solution.

To demonstrate that Algorithm 1 can be easily used to update (3.3) with arbitrary changes, we discuss two examples below. Additional examples can be found in the next few chapters.

1. Suppose we have solved (3.3) (for the system $y_t = \Phi_t \Psi_t \alpha_t + e_t$), and we have a sparse estimate for α_t in the form of $\widehat{\alpha}_t$. Then we receive a new set of measurements $y_{t+1} = \Phi_{t+1} \Psi_t \alpha_{t+1} + e_{t+1}$, for which we expect that α_{t+1} remains close to α_t. Therefore, we provide $\widehat{\mathbf{x}} \leftarrow \widehat{\alpha}_t$ as a warm-start vector in Algorithm 1 along with system matrix $\mathbf{\Phi} \leftarrow \Phi_{t+1}\Psi_{t+1}$, measurement vector $\mathbf{y} \leftarrow y_{t+1}$, (possibly updated) weights $\mathbf{W} \leftarrow W_{t+1}$, and $\mathbf{u}$ as defined in (3.8). We also need to provide the inverse or decomposition factors of the Gram matrix $\mathbf{\Phi}_{\widehat{\Gamma}}^T \mathbf{\Phi}_{\widehat{\Gamma}}$ for the initialization, which can either be easily updated using the existing inverse or factors of $\Phi_t \Psi_t$ (if the two systems are related to each other), or it needs to be computed from scratch once.

2. Suppose we want to solve (3.5) for α_t. We want to use $\widehat{\alpha}_t = \Psi_t^T \widehat{x}_t$ as a warm-start vector in Algorithm 1. We provide warm-start vector $\widehat{\mathbf{x}} \leftarrow \widehat{\alpha}_t$, system matrix $\mathbf{\Phi} \leftarrow \begin{bmatrix} \Phi_t \Psi_t \\ \sqrt{\lambda_t} \Psi_t \end{bmatrix}$, measurement vector $\mathbf{y} \leftarrow \begin{bmatrix} y_t \\ \sqrt{\lambda_t} \widehat{x}_t \end{bmatrix}$, weights $\mathbf{W} \leftarrow W_t$, and $\mathbf{u}$ as defined in (3.8). Similarly, we have to provide the inverse or decomposition factors of the Gram matrix $\mathbf{\Phi}_{\widehat{\Gamma}}^T \mathbf{\Phi}_{\widehat{\Gamma}}$, which can be computed using a low-rank update for this example.

CHAPTER IV

TIME-VARYING SPARSE SIGNALS

In this chapter we consider the problem of estimating a time-varying sparse signal from a series of linear measurements. We assume that the signal changes only slightly between consecutive measurements so that the reconstructions will be closely related. This type of problem can arise in various situations where we have to estimate closely related sparse signals. For example, in real-time magnetic resonance imaging we want to reconstruct a series of closely related frames from samples in the frequency domain [88]. Another application is channel equalization in communications, where we continuously estimate a time varying (and often times sparse) channel impulse response [51, 83]. Other applications may involve video surveillance or object tracking where we want to update the estimate at regular intervals. The algorithm and experiments presented in this chapter have appeared in [14].

4.1 Problem formulation

Suppose we are given a set of measurements $y_t = \Phi x_t + e_t$, where the measurement matrix is fixed over time and x_t is a time-varying sparse vector. We solve the following ℓ_1 problem to compute a sparse estimate for x_t:

$$\underset{x}{\text{minimize}}\ \tau\|x\|_1 + \frac{1}{2}\|\Phi x - y_t\|_2^2, \tag{4.1}$$

where $\tau > 0$ is a regularization parameter. The problem in (4.1) is equivalent to the one in (3.3) in that all the weights are set to τ. Let us denote the estimate of x_t as $\widehat{x}_t$. Now assume that x_t changes slightly to x_{t+1} (e.g., support remains almost the same), and we receive the new set of measurements: $y_{t+1} = \Phi x_{t+1} + e_{t+1}$. Our task

is to compute the solution of the following updated ℓ_1 problem:

$$\underset{x}{\text{minimize}}\ \tau\|x\|_1 + \frac{1}{2}\|\Phi x - y_{t+1}\|_2^2. \tag{4.2}$$

Instead of solving the new ℓ_1 problem from scratch, we use the available estimate from previous iteration, $\widehat{x}_t$, in a homotopy algorithm to quickly compute the solution for (4.2). Although we can formulate the homotopy for (4.2) using Algorithm 1 in Section 3.2, here we state the homotopy formulation that we used in our original work [8, 14] on this problem.

4.1.1 Homotopy formulation

Let us write the homotopy formulation for (4.2) as follows.

$$\text{minimize}\ \tau\|x\|_1 + \frac{1}{2}\|\Phi x - (1-\epsilon)y_t - \epsilon y_{t+1}\|_2^2, \tag{4.3}$$

where ϵ is the homotopy parameter. As we increase ϵ from 0 to 1, we gradually include the new measurements into the system and remove the old ones; the problem in (4.3) transforms from (4.1) to (4.2); and its solution moves from $\widehat{x}_t$ to the desired solution, say $\widehat{x}_{t+1}$. The homotopy algorithm similarly breaks down the solution update, in a systematic and efficient way, into a small number of linear steps. Each step consists of a rank-one update and a small number of matrix-vector multiplications.

Following the similar optimality criteria described in Section 3.2, the solution x^* for (4.3) at a given value of ϵ must obey the following optimality conditions:

$$\Phi_\Gamma^T(\Phi x^* - (1-\epsilon)y_t - \epsilon y_{t+1}) = -\tau z \tag{4.4a}$$

$$\|\Phi_{\Gamma^c}^T(\Phi x^* - (1-\epsilon)y_t - \epsilon y_{t+1})\|_\infty \leq \tau, \tag{4.4b}$$

where Γ denotes the support of x^* and z denotes the sign sequence of x^* on Γ. We notice from (4.4a) that the solution to (4.3) follows a piecewise linear path as ϵ varies; the critical points in this path occur when an element is either added to or removed from the solution x^*.

For every homotopy step we jump from one critical value of ϵ to the next while updating the support and the sign sequence of the solution, until ϵ is equal to 1. As we increase ϵ to $\epsilon + \delta$ for an infinitesimal $\delta > 0$, the solution moves in the direction ∂x, which to maintain optimality must obey

$$\Phi_\Gamma^T(\Phi x^* - (1-\epsilon)y_t - \epsilon y_{t+1}) + \delta\Phi_\Gamma^T\Phi\partial x - \delta\Phi_\Gamma^T(y_{t+1} - y_t) = -\tau z. \tag{4.5}$$

Thus, the update direction can be written as

$$\partial x = \begin{cases} (\Phi_\Gamma^T\Phi_\Gamma)^{-1}\Phi_\Gamma^T(y_{t+1} - y_t), & \text{on } \Gamma, \\ 0, & \text{otherwise.} \end{cases} \tag{4.6}$$

We move along the update direction ∂x, by increasing the step size δ, until one of the two things happens: one of the entries in $x^* + \delta\partial x$ shrinks to zero or one of the constraints in (4.4b) becomes active (equal to τ). We update the support and sign at such critical point, and calculate new update direction. We repeat this process until $\epsilon = 1$.

The main computational cost at every homotopy step comes from solving a $|\Gamma| \times |\Gamma|$ system of equations to compute the direction in (4.6), and few matrix-vector multiplications to compute the step size. Since the support changes by a single element at every homotopy step, the update direction can be computed using rank-one update methods. Therefore, the computational cost of each step is equivalent to a few matrix-vector multiplications.

4.2 Numerical experiments

The algorithm is most effective when the support of the solution does not change too much from instance to instance.

We performed a number of experiments to evaluate the performance of homotopy updating against different solvers. The results for the recovery of different time-varying sparse signals are presented in Table 4.1. In each of our experiments, we

started with a sparse signal $x_t \in \mathbb{R}^N$ and its M measurements according to the model $y_t = \Phi x_t + e_t$. We first solve (4.1) for a given value of τ and denote its solution as $\widehat{x}_t$. Then we perturbed the signal x_t to generate x_{t+1} and generated a new set of M measurements as $y_{t+1} = \Phi x_{t+1} + e_{t+1}$. We solved (4.3) using the homotopy update; the results are presented under **Dynamic** ℓ_1. We compared the dynamic updating scheme with three popular solvers: **LASSO**, which solves the standard LASSO/BPDN homotopy for (4.2) without a warm start [11, 59]; **GPSR-BB**, which solves gradient projection for sparse reconstruction [62], using $\widehat{x}_t$ as a warm-start; and **FPC_AS**, which solves fixed point continuation method with active set selection [144], using $\widehat{x}_t$ as a warm-start.

To gauge how the difference in support affects the speed of the homotopy update, we used a synthetic signal in our first set of experiments. We generated x_t by selecting S locations at random and assigned them ± 1 values with equal probability. We created x_{t+1} by perturbing x_t as follows. We added S_n new entries at random locations with their values selected from i.i.d. $\mathcal{N}(0, 1)$ distribution, where S_n was selected uniformly from $[0, S/20]$; in addition to this, we perturbed all nonzero entries with i.i.d. $\mathcal{N}(0, 0.01)$ distributed numbers. We selected entries in Φ from i.i.d. $\mathcal{N}(0, 1/M)$ distribution, and entries in e from i.i.d. $\mathcal{N}(0, 0.01)$. The experiments were performed on a standard desktop PC for different values of $\tau = \lambda \|\Phi^T y\|_\infty$ with $\lambda \in \{0.5, 0.1, 0.05, 0.01\}$, and we recorded the average number of matrix-vector multiplications for Φ and Φ^T and average computation time over 500 independent trials.

Table 4.1 also contains results for three other experiments with the following descriptions.

Blocks: We recovered a series of 200 piecewise constant signals of length $N = 2048$ from $M = 1024$ measurements. We started with the Blocks signal from WaveLab [29], and every time we perturbed the previous signal by changing the heights of the flat

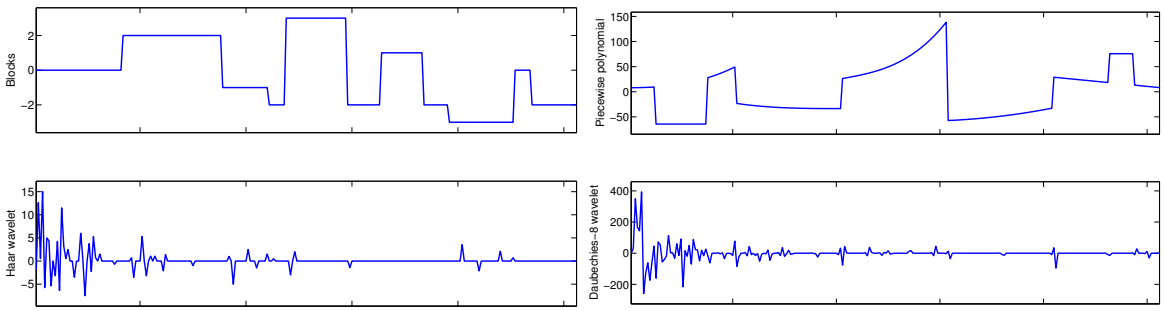


Figure 4.1: Snapshots of Blocks and Piecewise polynomial signals along with their wavelet coefficients.

regions by multiplying them independently with a random number, drawn uniformly between 0.8 and 1.2. We used Haar wavelet transform of the signals to create the x_t. As the signal changes, the signs and locations of the coefficients in x_t change as well. An example of the Blocks signal and its wavelet coefficients is shown in Figure 4.1.

Piecewise polynomial: We used piecewise polynomial (cubic) signal and represented it using the Daubechies-8 wavelet transform. We perturbed the signal by adding a small Gaussian random variable to the polynomial coefficients. An example of the Piecewise polynomial signal and its wavelet coefficients is also shown in Figure 4.1.

Slices of the *House* image: We used the 256 column slices of the House image, shown in Figure 4.2, as our sequence of signals. We used the Haar wavelet transform to compute sparse vectors x_t. As the singularities move slightly from slice to slice, more of the support in the wavelet domain changes, making this a more challenging data set than the previous examples.

We observed that dynamic ℓ_1 updating compares favorably against the three other solvers in all these experiments.

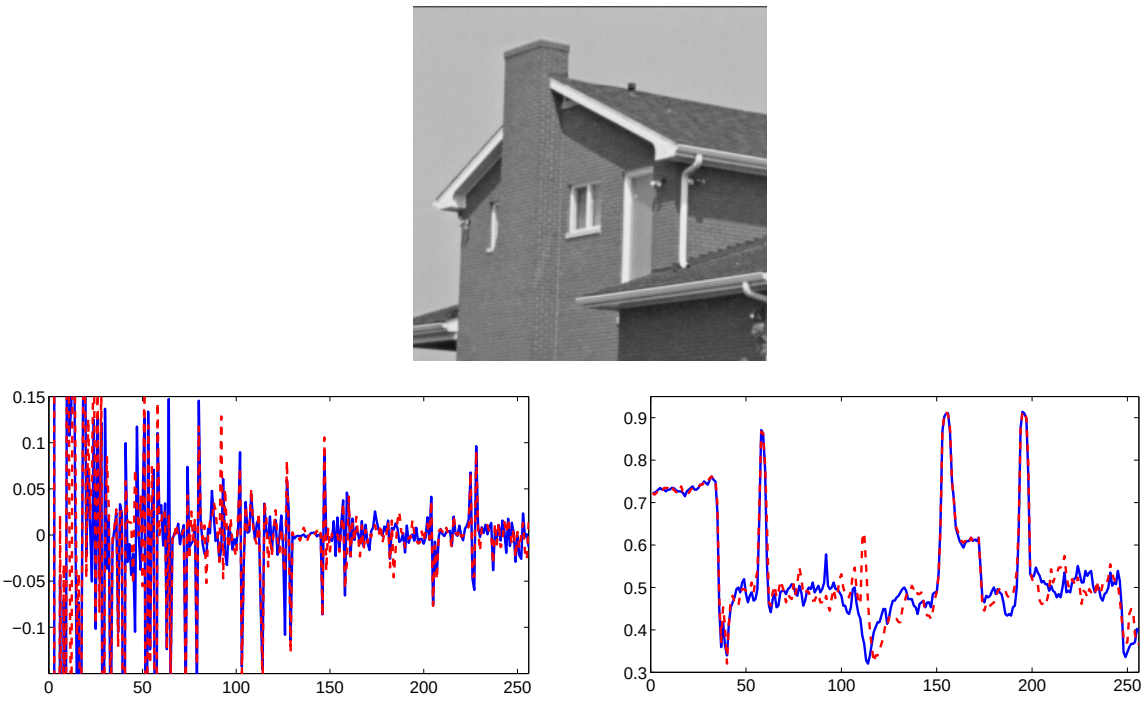


Figure 4.2: An image of house (256x256): Lower images represent two consecutive slices from this image (on the right) and their respective wavelet transform coefficients (on the left). Note the small difference between consecutive slices.

Table 4.1: Comparison of the dynamic ℓ_1 updating for time-varying sparse signals with standard LASSO homotopy, warm-started GPSR-BB, and warm-started FPC_AS. Results are reported as **left:** the average number of products with Φ^T and Φ **; right:** average computation time in seconds. $\tau \stackrel{\text{def}}{=} \lambda \|\Phi^T y\|_\infty$.

Signal	λ	**Dynamic ℓ_1**	**LASSO**	**GPSR-BB**	**FPC_AS**
$N = 1024$,	0.5	11.8 ; 0.03	42.1 ; 0.10	15.3 ; 0.03	31.3 ; 0.06
$M = 512$,	0.1	12.9 ; 0.06	154.5 ; 0.50	54.4 ; 0.01	103.4 ; 0.13
$S = M/5$,	0.05	14.6 ; 0.06	162 ; 0.52	58.2 ; 0.10	102.4 ; 0.14
± 1 spikes	0.01	23.7 ; 0.13	235 ; 0.92	104.5 ; 0.18	148.7 ; 0.18
Blocks	0.01	2.7 ; 0.03	76.8 ; 0.49	17 ; 0.13	53.5 ; 0.20
Pcwpoly	0.01	13.8 ; 0.15	150.2 ; 1.10	26.1 ; 0.21	66.9 ; 0.25
House	0.005	44.7 ; 0.02	76.8 ; 0.03	220.5 ; 0.03	147 ; 0.06

CHAPTER V

SEQUENTIAL MEASUREMENTS OF FIXED SIGNALS

In this chapter we consider the problem of adding sequential measurements of a fixed signal into the system and dynamically updating the solution of the ℓ_1 problem. Adding new measurements to the ℓ_1 problem will change the solution, but instead of solving a new problem from scratch, the homotopy method can quickly update the solution in a small number of steps. The recursive least squares filter performs equivalent tasks in least-squares settings, where instead of solving the entire system of equations every time new measurements are added, the solution can be recursively updated using a low-rank update. Although the homotopy method we use for dynamic ℓ_1 updating does not provide update in one step, it breaks the updating procedure into a sequence of rank-one updates.

5.1 Problem formulation

Suppose we are given a set of measurements $y_t = \Phi_t \bar{x} + e_t$, where the sparse signal $\bar{x}$ remains fixed over time. We solve the following ℓ_1 problem to compute a sparse estimate for $\bar{x}$:

$$\underset{x}{\text{minimize}}\ \tau\|x\|_1 + \frac{1}{2}\|\Phi_t x - y_t\|_2^2, \tag{5.1}$$

where $\tau > 0$ is a regularization parameter. The problem in (5.1) is equivalent to the one in (3.3) in that all the weights are set to τ. Let us denote the estimate of $\bar{x}$ as $\widehat{x}$. We receive a new set of measurements: $y_{t+1} = \Phi_{t+1}\bar{x} + e_{t+1}$. Our task is to compute the solution of the following updated ℓ_1 problem:

$$\underset{x}{\text{minimize}}\ \tau\|x\|_1 + \frac{1}{2}\|\Phi_t x - y_t\|_2^2 + \frac{1}{2}\|\Phi_{t+1} x - y_{t+1}\|_2^2. \tag{5.2}$$

Instead of solving the new ℓ_1 problem from scratch, we use the estimate available

from the previous iteration, $\widehat{x}$, in our homotopy algorithm to quickly compute the solution for (5.2). We write the homotopy problem for (5.2) in the form of (3.7) as

$$\underset{x}{\text{minimize}}\ \tau\|x\|_1 + \frac{1}{2}\|\Phi_t x - y_t\|_2^2 + \frac{1}{2}\|\Phi_{t+1}x - y_{t+1}\|_2^2 + (1-\epsilon)\mathbf{u}^T x. \tag{5.3}$$

We solve (5.3) using Algorithm 1 described in Section 3.2, where we assigned $\mathbf{\Phi} \leftarrow \begin{bmatrix} \Phi_t \\ \Phi_{t+1} \end{bmatrix}$, $\mathbf{y} \leftarrow \begin{bmatrix} y_t \\ y_{t+1} \end{bmatrix}$, $\mathbf{W} \leftarrow \tau$ (equal weights), and $\mathbf{u} \leftarrow -\tau\widehat{z} - \mathbf{\Phi}^T(\mathbf{\Phi}\widehat{x} - \mathbf{y})$ using $\widehat{x}$ as the warm-start vector with sign sequence $\widehat{z}$ and support $\widehat{\Gamma}$.

5.1.1 Previous work

In our previous work [9, 14], we used the following homotopy formulations for this problem:

$$\underset{x}{\text{minimize}}\ \tau\|x\|_1 + \frac{1}{2}(\|\Phi_t x - y_t\|_2^2 + \epsilon\|\Phi_{t+1}x - y_{t+1}\|_2^2). \tag{5.4}$$

where ϵ is the homotopy parameter. A similar version of the homotopy algorithm was independently proposed in [65]. However, the homotopy updates for (5.4) allows one new measurement at a time. A more versatile homotopy scheme that can add multiple measurements simultaneously was presented in [9]:

$$\underset{x}{\text{minimize}}\ \tau\|x\|_1 + \frac{1}{2}(\|\Phi_t x - y_t\|_2^2 + \|\Phi_{t+1}x - (1-\epsilon)\Phi_{t+1}\widehat{x} - \epsilon y_{t+1}\|_2^2), \tag{5.5}$$

which is similar to the homotopy formulation in (4.3). Note that the new term does not affect the objective at $\epsilon = 0$, and $\widehat{x}$ remains a valid solution of (5.5). As ϵ is increased to 1, the solution of (5.5) reaches the desired solution of (5.2). The homotopy algorithm for (5.5) is identical to the homotopy algorithm for dynamic updating of time-varying signals, discussed in Section 4.1. However, none of these two algorithms accommodate arbitrary changes in the system matrices. On the other hand, the homotopy formulation in (3.7) allows arbitrary changes in the system.

5.2 Numerical experiments

Simulation results comparing the computational cost of updating the solution with P additional measurements are presented in Table 5.1. We selected the entries in $M \times N$ matrix Φ_t and $P \times N$ matrix Φ_{t+1} from i.i.d. $\mathcal{N}(0, 1/(M+P))$ distribution, and entries in e_t and e_{t+1} from $\mathcal{N}(0, 0.01)$. Sparse signal $\bar{x}$ was generated by selecting $S = M/5$ locations at random and assigning them ± 1 values with equal probability. We performed 50 experiments with $N = 1024, M = 512$, at different values of $\tau = \lambda \|\mathbf{\Phi}^T \mathbf{y}\|_\infty$, using different values of P. We recorded average number of applications of $\mathbf{\Phi}$ and its adjoint. The results are summarized in Table 5.1 under **Dynamic** ℓ_1, and compared against the standard LASSO/BPDN homotopy **LASSO** without a warm start, **GPSR-BB** with the warm start [62], and **FPC_AS** with the warm start [144].

We observed that dynamic ℓ_1 updating performed consistently better than the three other solvers in all these experiments. The average number of homotopy steps required for the update varies with the number of nonzero entries in the solution. For large values of τ, the solution has a small number of non-zero entries and the update requires 2–7 homotopy steps. For smaller values of τ, the solution has many more non-zero terms and the number of homotopy steps required for the update increases; for example, at $\tau = 0.01 \|\mathbf{\Phi}^T y\|_\infty$ an average 8 homotopy steps were required to add one new measurement.

Table 5.1: Comparison of the dynamic ℓ_1 updating for P sequential measurements with standard LASSO homotopy, warm-started GPSR, and warm-started FPC_AS. Results are reported as **left:** the average number of products with $\mathbf{\Phi}^T$ and $\mathbf{\Phi}$ **; right:** average computation time in seconds. $\tau \stackrel{\text{def}}{=} \lambda \|\mathbf{\Phi}^T \mathbf{y}\|_\infty$.

P	λ	**Dynamic ℓ_1**	**LASSO**	**GPSR-BB**	**FPC_AS**
1	0.5	2.3 ; 0.004	41.9 ; 0.066	11.9 ; 0.012	15.9 ; 0.018
	0.1	4.8 ; 0.010	161.4 ; 0.240	42.6 ; 0.037	50.9 ; 0.039
	0.05	4.6 ; 0.010	164.6 ; 0.238	38.8 ; 0.034	97.2 ; 0.074
	0.01	8.1 ; 0.021	235 ; 0.386	55.5 ; 0.048	78.4 ; 0.059
5	0.5	5.9 ; 0.008	42.0 ; 0.065	14.2 ; 0.013	15.9 ; 0.018
	0.1	9.7 ; 0.018	154.6 ; 0.216	46.4 ; 0.037	47.1 ; 0.034
	0.05	10.9 ; 0.020	163.7 ; 0.231	48 ; 0.039	99 ; 0.071
	0.01	20.7 ; 0.045	230.7 ; 0.358	66.6 ; 0.055	78 ; 0.056
10	0.5	7.5 ; 0.011	44.7 ; 0.074	15 ; 0.015	16.3 ; 0.018
	0.1	15.2 ; 0.028	157.5 ; 0.224	53.1 ; 0.044	47.9 ; 0.036
	0.05	16.4 ; 0.030	165.4 ; 0.237	52.1 ; 0.043	97.9 ; 0.070
	0.01	30.1 ; 0.071	242.5 ; 0.406	75.4 ; 0.0625	81.3 ; 0.061

CHAPTER VI

ITERATIVE AND ADAPTIVE WEIGHTED ℓ_1

To recover a sparse signal from an underdetermined system, we often solve a constrained ℓ_1-norm minimization problem. In many cases, the signal sparsity and the recovery performance can be further improved by replacing the ℓ_1 norm with a "weighted" ℓ_1 norm [42, 81, 152]. Without any prior information about nonzero elements of the signal, the procedure for selecting weights is iterative in nature. Common approaches update the weights at every iteration using the solution of a weighted ℓ_1 problem from the previous iteration. In this chapter, we present two homotopy-based algorithms for efficiently solving reweighted ℓ_1 problems. First, we present an algorithm that quickly updates the solution of a weighted ℓ_1 problem as the weights change. Since the solution changes only slightly with small changes in the weights, we develop a homotopy algorithm that replaces the old weights with the new ones in a small number of computationally inexpensive steps. Second, we propose an algorithm that solves a weighted ℓ_1 problem by adaptively selecting the weights while estimating the signal. This algorithm integrates the reweighting into every step along the homotopy path by changing the weights according to the changes in the solution and its support, allowing us to achieve a high quality signal reconstruction by solving a single homotopy problem. We compare the performance of both algorithms, in terms of reconstruction accuracy and computational complexity, against state-of-the-art solvers and show that our methods have smaller computational cost. In addition, we demonstrate that the adaptive selection of the weights inside the homotopy often yields reconstructions of higher quality. The algorithm and experiments presented in this chapter have been submitted for publication [12].

6.1 Introduction

Consider the following linear system:

$$y = \Phi\bar{x} + e, \tag{6.1}$$

where $\bar{x} \in \mathbb{R}^N$ is an unknown sparse vector that is measured through an $M \times N$ matrix Φ, y is the measurement vector, and e denotes noise. We want to solve the following weighted ℓ_1-norm minimization problem to estimate $\bar{x}$:

$$\underset{x}{\text{minimize}} \ \|Wx\|_1 + \frac{1}{2}\|\Phi x - y\|_2^2, \tag{6.2}$$

where W is a diagonal matrix that contains positive weights $w > 0$ at its diagonal. We can adjust w in (6.2) to selectively penalize different coefficients in the solution. To promote the same sparsity structure in the solution that is present in the original signal, we can select w such that the weights have small values on the nonzero locations of the signal and significantly larger values elsewhere [45, 47, 63]. Since the locations and amplitudes of the nonzero coefficients of the original signal are unknown a priori, the critical task of selecting the weights is performed iteratively. Common approaches for such "iterative reweighting" re-compute weights at every iteration using the solution of (6.2) at the previous iteration. Suppose $\widehat{x}$ denotes the solution of (6.2) for a given set of weights. For the next iteration, we compute the w_i as

$$w_i = \frac{\tau}{|\widehat{x}_i| + \epsilon}, \tag{6.3}$$

for $i = 1, \ldots, N$, using an appropriate choice of positive values for parameters τ and ϵ. We use these updated weights in (6.2) to re-compute the signal estimate, which we then use in (6.3) to update the weights for the next iteration. The major computational cost of every iteration in such a reweighting scheme arises from solving (6.2), for which a number of solvers are available [18–20, 137, 147, 148].

In this chapter, we present two homotopy-based algorithms for efficiently solving reweighted ℓ_1-norm minimization problems. In a typical homotopy algorithm for an ℓ_1

problem, as the homotopy parameter changes, the solution moves along a piecewise-linear path, and each segment on this homotopy path is traced with a computationally inexpensive homotopy step. The major computational cost for every homotopy step involves one full matrix-vector multiplication and one rank-one update of the inverse of a small matrix. The standard LASSO homotopy solves (6.2) when all the weights are set to the same value, say τ [14, 59, 106] (further details on LASSO homotopy are also presented in Section 2.3.1). By comparison, (6.2) has N parameters in the form of w_i, and both the homotopy algorithms we present in this chapter change the w_i in such a way that their respective solutions follow piecewise-linear paths in sequences of inexpensive homotopy steps.

First, we present an algorithm that quickly updates the solution of (6.2) as the weights change in the iterative reweighting framework. Suppose we have the solution of (6.2) for a given set of weights w, and we wish to update the weights to $\tilde{w}$. We develop a homotopy program that updates the solution of (6.2) by replacing the old weights (w) with the new ones ($\tilde{w}$). Since the solution of (6.2) changes only slightly with small changes in the weights, the homotopy procedure uses an existing solution as the starting point and updates the solution in a small number of inexpensive homotopy steps.

Second, we propose a new homotopy algorithm that performs an internal "adaptive reweighting" after every homotopy step. Adaptive reweighting is a natural combination of the standard homotopy and reweighting—instead of solving the ℓ_1 program in (6.2) multiple times with different weights, it adjusts the weights (using the same principles as iterative reweighted ℓ_1) at every homotopy step. Our algorithm yields a solution for a weighted ℓ_1 problem of the form (6.2) for which the final values of w_i are not assigned a priori, but instead are adaptively selected inside the algorithm. In our proposed homotopy algorithm, we follow a solution path for (6.2) by adaptively reducing each w_i, while updating the support of the signal estimate by one element

at every step. In contrast with the standard LASSO homotopy, which assumes all the w_i to have same value, here we update each w_i independently. After every homotopy step, we adjust the weights so that w on the support of the available signal estimate shrinks at a faster rate, toward smaller values (e.g., of the form in (6.3)), while w elsewhere shrink at a slower rate, toward a predefined threshold ($\tau > 0$). In contrast with the iterative reweighting, which solves (6.2) for a fixed set of weights and updates the weights after every reweighting iteration, here we update the weights after every homotopy step. This allows us to recover a high-quality signal by solving a single homotopy problem, instead of solving (6.2) multiple times via iterative reweighting (i.e., updating w after solving (6.2)). We have also observed that such an adaptive reweighting tends to provide better quality of reconstruction compared to the standard method of iterative reweighting. In addition to assigning smaller weights to the active indices, this adaptive reweighting serves another purpose: it encourages active elements to remain nonzero, which in turn reduces the total number of homotopy steps required for solving the entire problem.

Our proposed adaptive reweighting method bears some resemblance to a variable selection method recently presented in [113], which adjusts the level of shrinkage at each step (equivalent to reducing the w_i toward zero) so as to optimize the selection of the next variable. However, the procedure we adopt for the selection of w_i in this work is more flexible, and it offers an explicit control over the values of w_i, which we exploit to embed a reweighted ℓ_1-norm regularization inside the homotopy.

6.2 Iterative reweighting via homotopy

In this section, we present a homotopy algorithm for iterative reweighting that quickly updates the solution of (6.2) as the weights change. Suppose we have solved (6.2) for a given set of weights in W to get an estimate $\widehat{x}$, and now we wish to solve the

following modified problem:

$$\underset{x}{\text{minimize}}\ \|\widetilde{W}x\|_1 + \frac{1}{2}\|\Phi x - y\|_2^2, \tag{6.4}$$

where the $\widetilde{W}$ contains new weights $\tilde{w}$ in its diagonal. To incorporate changes in the weights (i.e., replace the w_i with the $\tilde{w}_i$) and quickly compute the new solution of (6.2), we can use the homotopy formulation and the algorithm described in Section 3.2. The homotopy formulation for that takes the following form:

$$\underset{x}{\text{minimize}}\ \|\widetilde{W}\,x\|_1 + \frac{1}{2}\|\Phi x - y\|_2^2 + (1-\epsilon)u^T x, \tag{6.5}$$

where $u \stackrel{\text{def}}{=} -\widetilde{W}\widehat{z} - \Phi^T(\Phi\widehat{x} - y)$ is as described in (3.8), $\widehat{x}$ denotes the warm-start vector and $\widehat{z}$ denotes its sign sequence. However, in our original work [12], we used the following homotopy formulation for this problem:

$$\underset{x}{\text{minimize}}\ \|((1-\epsilon)W + \epsilon\widetilde{W})\,x\|_1 + \frac{1}{2}\|\Phi x - y\|_2^2, \tag{6.6}$$

where ϵ denotes the homotopy parameter that we change from zero to one to phase in the new weights and phase out the old ones. As we increase ϵ, the solution of (6.6) follows a homotopy path from the solution of (6.2) to that of (6.4). We show below that the path the solution takes is also piecewise linear with respect to ϵ, making every homotopy step computationally inexpensive. A pseudocode outlining the important steps is presented in Algorithm 2.

At any value of ϵ, the solution x^* must obey the following optimality conditions:

$$\phi_i^T(\Phi x^* - y) = -((1-\epsilon)w_i + \epsilon\tilde{w}_i)z_i, \qquad \text{for all } i \in \Gamma, \tag{6.7a}$$

and

$$|\phi_i^T(\Phi x^* - y)| \le (1-\epsilon)w_i + \epsilon\tilde{w}_i, \qquad \text{for all } i \in \Gamma^c, \tag{6.7b}$$

where ϕ_i denotes ith column of Φ. As we increase ϵ to $\epsilon + \delta$, for some small δ, the solution moves in a direction ∂x and the optimality conditions change as

$$\Phi_\Gamma^T(\Phi x^* - y) + \delta\Phi_\Gamma^T\Phi\partial x = -((1-\epsilon)W_\Gamma + \epsilon\widetilde{W}_\Gamma)z_\Gamma + \delta(W_\Gamma - \widetilde{W}_\Gamma)z_\Gamma \tag{6.8a}$$

Algorithm 2 Iterative reweighting via homotopy

Input: Φ, y, $\widehat{x}$, w, and $\tilde{w}$
Output: x^*

Initialize: $\epsilon = 0$, $x^* \leftarrow \widehat{x}$
Repeat:
  Compute ∂x in (6.9) ▷ Update direction
  Compute p, d, q, and s in (6.8b)
  Compute $\delta^* = \min(\delta^+, \delta^-)$ ▷ Step size
  if $\epsilon + \delta^* > 1$ **then**
    $\delta^* \leftarrow 1 - \epsilon$ ▷ Last iteration
    $x^* \leftarrow x^* + \delta^* \partial x$ ▷ Final solution
    `break`
  end if
  $x^* \leftarrow x^* + \delta^* \partial x$ ▷ Update the solution
  $\epsilon \leftarrow \epsilon + \delta^* \partial x$ ▷ Update the homotopy parameter
  if $\delta^* = \delta^-$ **then**
    $\Gamma \leftarrow \Gamma \backslash \gamma^-$ ▷ Remove an element from the support
  else
    $\Gamma \leftarrow \Gamma \cup \gamma^+$ ▷ Add a new element to the support
  end if
until $\epsilon = 1$

$$|\underbrace{\phi_i^T(\Phi x^* - y)}_{p_i}| + \delta\,\underbrace{\phi_i^T \Phi \partial x}_{d_i}| \le \underbrace{(1-\epsilon)w_i + \epsilon\tilde{w}_i}_{q_i} + \delta\,\underbrace{(\tilde{w}_i - w_i)}_{s_i}, \tag{6.8b}$$

where W_Γ and $\widetilde{W}_\Gamma$ denote $|\Gamma| \times |\Gamma|$ diagonal matrices with their diagonal entries being the values of w and $\tilde{w}$ on Γ, respectively. The update direction is specified by the new optimality conditions (6.8a) as

$$\partial x = \begin{cases} (\Phi_\Gamma^T \Phi_\Gamma)^{-1}(W_\Gamma - \widetilde{W}_\Gamma) z_\Gamma & \text{on } \Gamma \\ 0 & \text{on } \Gamma^c. \end{cases} \tag{6.9}$$

As we increase δ, the solution moves in the direction ∂x until either a new element enters the support of the solution (when a constraint in (6.8b) becomes active) or an existing element shrinks to zero. The stepsize that takes the solution to such a critical value of ϵ can be computed as $\delta^* = \min(\delta^+, \delta^-)$, where

$$\delta^+ = \min_{i \in \Gamma^c} \left(\frac{q_i - p_i}{-s_i + d_i}, \frac{-q_i - p_i}{s_i + d_i} \right)_+ \tag{6.10a}$$

$$\delta^- = \min_{i \in \Gamma} \left(\frac{-x_i^*}{\partial x_i} \right)_+ . \tag{6.10b}$$

δ^+ denotes the smallest step-size that causes a constraint in (6.8b) to become active, indicating the entrance of a new element at index γ^+ in the support, whereas δ^- denotes the smallest step-size that shrinks an existing element at index γ^- to zero. The new critical value of ϵ becomes $\epsilon + \delta^*$, the signal estimate x^* becomes $x^* + \delta^* \partial x$, where its support and sign sequence are updated accordingly. At every homotopy step, we jump from one critical value of ϵ to the next while updating the support of the solution, until $\epsilon = 1$.

The main computational cost of every homotopy step comes from solving a $|\Gamma| \times |\Gamma|$ system of equations to compute ∂x in (6.9) and one matrix-vector multiplication to compute the d_i in (6.10). Since Γ changes by a single element at every homotopy step, the update direction can be computed using a rank-one update. As such, the computational cost of each homotopy step is close to one matrix-vector multiplication with Φ and one with Φ^T. We demonstrate with experiments in Section 6.4 that as the w_i change, our proposed homotopy algorithm updates the solution in a small number of homotopy steps, and the total cost for updating the weights is just a small fraction of the cost of solving (6.2) from scratch.

6.3 Adaptive reweighting via homotopy

In this section, we present a homotopy algorithm that solves a weighted ℓ_1-norm minimization problem of the form (6.2) by adaptively selecting the weights w_i. The motivation for this algorithm is that instead of solving (6.2) for a given set of w_i and updating the w_i after every reweighting iteration, we can perform reweighting at every homotopy step by updating the w_i as the signal estimate evolves. Recall that in the standard LASSO homotopy we build the solution by adding or removing one element in the support while shrinking a single homotopy parameter (see Section 2.3.1 for details on LASSO homotopy). By comparison, each w_i in (6.2) can act as a separate

homotopy parameter, and we can attempt to achieve desired values of w_i by adaptively shrinking them at every homotopy step.

In adaptive reweighting, we trace a solution path for (6.2) by adaptively reducing the w_i while updating the solution and its support in a sequence of inexpensive homotopy steps. At every homotopy step, we start with a solution of (6.2) for certain values of w_i. We encourage the algorithm to focus on the set of active indices in the solution (i.e., the support of the solution) and reduce the w_i so that they decrease at a faster rate and achieve smaller values on the active set than the w_i on the inactive set. Suppose, using certain criterion, we select the $\tilde{w}_i$ as the desired values of the weights. As we change the w_i toward the $\tilde{w}_i$, the solution moves in a certain direction until either the w_i become equal to the $\tilde{w}_i$ or the support of the solution changes by one element. By taking into account any change in the support, we revise the values of $\tilde{w}_i$ for the next homotopy step. We repeat this procedure until each w_i is reduced below a certain predefined threshold $\tau > 0$.

In summary, we solve a single homotopy problem that builds the solution of a weighted ℓ_1 problem of the form (6.2) by adjusting the w_i according to the changes in the support of the solution. A pseudocode with a high-level description is presented in Algorithm 3. Details regarding the weight selection, the update direction, and the step size and support selection are discussed below.

6.3.1 Weight selection criteria

Suppose we want to shrink the w_i in (6.2) toward a preset threshold τ, and by construction, we want the w_i to have smaller values on the support of the solution (e.g., of the form (6.3)). At every homotopy step, we divide the indices into an active and an inactive set. We can follow a number of heuristics to select the desired values of the weights ($\tilde{w}_i$) so that the w_i reduce at a faster rate on the active set than on the inactive set.

Algorithm 3 Adaptive reweighting via homotopy

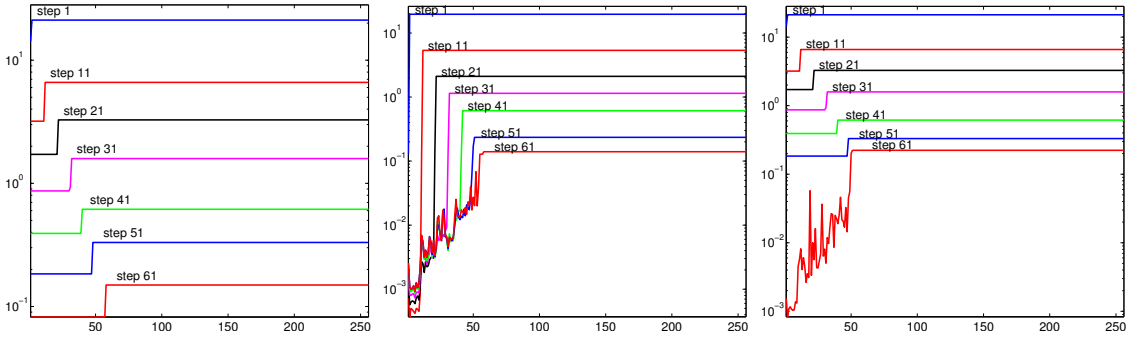
Input: Φ, y and τ
Output: x^*, w

Initialize: $x^* \leftarrow \mathbf{0}$, $w_i \leftarrow \max_i |\phi_i^T y|$ for all i, $\Gamma \leftarrow \arg\max_i |\phi_i^T y|$
Repeat:
  Select $\tilde{w}_i$ ▷ Desired values for the weights
  Compute ∂x in (6.13) ▷ Update direction
  Compute $\delta^* = \min(\delta^+, \delta^-)$ ▷ Step size
  $x^* \leftarrow x^* + \delta^* \partial x$ ▷ Update the solution
  $w_i \leftarrow w_i + \delta^*(\tilde{w}_i - w_i)$ ▷ Update w_i
  if $\delta^* = \delta^-$ **then**
    $\Gamma \leftarrow \Gamma \backslash \gamma^-$ ▷ Remove an element from the support
  else
    $\gamma^+ = \arg\max_{i \in \Gamma^c} |\phi_i^T(\Phi x^* - y)|$ ▷ (*Option 2 only*) Select new element
    $\Gamma \leftarrow \Gamma \cup \gamma^+$ ▷ Add a new element to the support
  end if
  $w_i \leftarrow \max_j |\phi_j^T(\Phi x^* - y)|$ for all $i \in \Gamma^c$ ▷ (*Optional*) Update w_i on the inactive set
until $\max_i (w_i) \leq \tau$

Initialization: We initialize all the weights with a large value (e.g., $w_i = \max_i |\phi_i^T y|$ for all i) for which the solution is a zero vector. The only element in the active set correspond to $\arg\max_i |\phi_i^T y|$, where ϕ_i denotes ith column in Φ.

Weights on the active set: We can select the $\tilde{w}_i$ on the active set in a variety of ways. For instance, we can select each $\tilde{w}_i$ as a fraction of the present value of the corresponding w_i as $\tilde{w}_i \leftarrow w_i/\beta$, for some $\beta > 1$, or as a fraction of the maximum value of the w_i on the active set as $\tilde{w}_i \leftarrow \max_{i \in \Gamma} w_i/\beta$. The former will reduce each w_i on the active set at the same rate, while the latter will reduce each w_i to the same value as well. To introduce reweighting of the form in (6.3), we can change the w_i on the support using the solution from the previous homotopy step as $\tilde{w}_i \leftarrow \min(\tau, \tau/\beta|x_i^*|)$, for some $\beta > 1$.

Weights on the inactive set: We can assign the $\tilde{w}_i$ on the inactive set a single value that is either equal to the maximum value of $\tilde{w}_i$ on the active set or equal to τ, whichever is the larger.



(a) $\tilde{w}_i \leftarrow \dfrac{\max_{i\in\Gamma} w_i}{2}$ **(b)** $\tilde{w}_i \leftarrow \min\left(\tau, \dfrac{\tau}{\beta|x_i^*|}\right)$ **(c)** a hybrid of (a) and (b)

Figure 6.1: Illustrations of variations (on a log-scale) in the w_i on the sets of active and inactive indices at different homotopy steps. Subfigures (a), (b), and (c) correspond to three different choices for the $\tilde{w}_i$. Left part of each plot (with the lower values of w_i) corresponds to the active set of indices in the order in which they entered the support and the right part (with larger values of the w_i) corresponds to the inactive set of the solution at every step.

In Figure 6.1, we present three examples to illustrate the evolution of the w_i on the active and the inactive set at various homotopy steps. These examples were constructed with different choices of $\tilde{w}_i$ during the recovery a Blocks signal of length 256 from 85 noisy Gaussian measurements according to the experimental setup described in Section 6.4. We plotted the w_i at different homotopy steps in such a way that the left part of each plot corresponds to the active set and the right part to the inactive set of the solution. As the homotopy progresses, the support size increases and all the w_i decrease, but the w_i on the active set become distinctly smaller than the rest. In Figure 6.1a we selected $\tilde{w}_i \leftarrow \max_{i\in\Gamma} w_i/2$ at every step; in Figure 6.1b we selected $\tilde{w}_i \leftarrow \min\left(\tau, \tau/\beta|x_i^*|\right)$ at every step, with certain values of τ and β; and in Figure 6.1c we selected $\tilde{w}_i \leftarrow \max_{i\in\Gamma} w_i/2$ for first few homotopy steps and then we selected $\tilde{w}_i \leftarrow \tau/\beta|x_i^*|$. In our experiments in Section 6.4, we selected weights according to the scheme illustrated in Figure 6.1b.

6.3.2 Update direction

To compute the update direction in which the solution moves as we change the weights w_i toward the $\tilde{w}_i$, we use the same methodology that we used for (6.6) in Section 6.2.

For any given values of the w_i, a solution x^* for (6.2) satisfies the following optimality conditions:

$$\phi_i^T(\Phi x^* - y) = -w_i z_i, \quad \text{for all } i \in \Gamma, \tag{6.11a}$$

and

$$|\phi_i^T(\Phi x^* - y)| \leq w_i, \quad \text{for all } i \in \Gamma^c, \tag{6.11b}$$

where Γ denotes the support of x^* and z denotes the sign sequence of x^* on Γ. If we change w_i toward $\tilde{w}_i$ along a straight line, $(1-\delta)w_i + \delta\tilde{w}_i$, the solution moves in a direction ∂x, which to maintain optimality must obey

$$\Phi_\Gamma^T(\Phi x^* - y) + \delta\Phi_\Gamma^T\Phi\partial x = -Wz + \delta(W_\Gamma - \widetilde{W}_\Gamma)z, \tag{6.12a}$$

$$|\underbrace{\phi_i^T(\Phi x^* - y)}_{p_i} + \delta\underbrace{\phi_i^T\Phi\partial x}_{d_i}| \leq \underbrace{w_i}_{q_i} + \delta\underbrace{(\tilde{w}_i - w_i)}_{s_i}, \tag{6.12b}$$

where W_Γ and $\widetilde{W}_\Gamma$ denote $|\Gamma| \times |\Gamma|$ diagonal matrices constructed with the respective values of old (w_i) and new ($\tilde{w}_i$) weights on Γ. Subtracting (6.11a) from (6.12a) yields the following expression for the update direction ∂x:

$$\partial x = \begin{cases} (\Phi_\Gamma^T\Phi_\Gamma)^{-1}(W_\Gamma - \widetilde{W}_\Gamma)z & \text{on } \Gamma \\ 0 & \text{on } \Gamma^c. \end{cases} \tag{6.13}$$

6.3.3 Step size and support selection

As we increase δ from 0 to 1, x^* moves along the update direction ∂x as $x^* + \delta\partial x$ and the w_i change toward $\tilde{w}_i$ as $w_i + \delta(\tilde{w}_i - w_i)$. At certain value of $\delta \in (0, 1)$, an existing element in x^* may shrink to zero, and we must remove that element from the support. Alternatively, an inactive constraint in (6.12b) may become active, and to maintain the optimality of the solution, we can choose one of the following options: 1) either include the index of the active constraint in the support or 2) artificially increase the value of w_i at that index. The stepsize and the support selection rule for the first option is same as we discussed before—compute $\delta^* = \min(\delta^+, \delta^-)$, using definitions of δ^+ and δ^- in (6.10), and update the support accordingly; a selection

rule for the second option, which we use in our experiments in Section 6.4 as well, is discussed below. The two additional operations for the second option are indicated using dashed boxes in Algorithm 2, where Line 11 is not executed in the first case and Line 14 is optional.

The optimality conditions (6.11b) and (6.12b) suggest that as long as a strict inequality is maintained for an index i in the inactive set, we can change the corresponding weight to an arbitrary value without affecting the solution. Since the w_i are not fixed a priori in this scheme, we have the flexibility to disregard any violation of the inequality constraints and adjust the w_i so that the solution remains optimal. Under this setting, we can compute the optimal stepsize δ^* and identify a change in the support of the signal as follows. Suppose δ^- causes an element at index $\gamma^- \in \Gamma$ in x^* to shrink to zero. If $\delta^- < 1$, we must remove γ^- from the support and set $\delta^* = \delta^-$. If $\delta^- > 1$, which implies that the w_i has changed to the $\tilde{w}_i$ on Γ, we set $\delta^* = \delta^+ = 1$ and search for a new element γ^+ to add to the support. We set $\delta^* = \min(\delta^-, 1)$ and update $x^* \leftarrow x^* + \delta^* \partial x$ and $w_i \leftarrow w_i + \delta^*(\tilde{w}_i - w_i)$. If $\delta^- > 1$, we select the new element γ^+ that corresponds to the inactive constraint with largest magnitude, which can be determined as

$$\gamma^+ = \arg\max_{i \in \Gamma^c} |\phi_i^T(\Phi x^* - y)|, \tag{6.14}$$

and set $w_{\gamma^+} = |\phi_{\gamma^+}^T(\Phi x^* - y)|$.

We repeat the procedure of selecting $\tilde{w}_i$, computing the update direction and the stepsize, and updating the solution and its support at every homotopy step, until a termination criterion is satisfied (e.g. $w_i \leq \tau$ for all i).

The main computational cost at every homotopy step comes from solving a $|\Gamma| \times |\Gamma|$ system of equations in (6.13) for computing ∂x and one matrix-vector multiplication whenever we need to find γ^+ in (6.14). Since Γ changes by a single element at every homotopy step, the update direction can be efficiently computed using a rank-one update. As such, the computational cost of every step is equivalent one matrix-vector

multiplication with Φ and one with Φ^T.

6.4 Numerical experiments

We present some experiments to demonstrate the performance of our proposed algorithms: (1) iterative reweighting via homotopy (Algorithm 2), which we will call IRW-H and (2) adaptive reweighting via homotopy (Algorithm 3), which we will call ARW-H. We evaluate the performances of ARW-H and IRW-H in terms of the computational cost and the reconstruction accuracy. We show that, in comparison with iterative reweighting schemes, solving (6.2) using ARW-H yields significantly higher quality signal reconstruction, at a computational cost that is comparable to solving (6.2) once from scratch. Furthermore, we show that using IRW-H we can quickly update the weights in (6.2) at a small computational expense. To compare ARW-H and IRW-H against existing ℓ_1 solvers, we also present results for the sparse signal recovery using iterative reweighting for three state-of-the-art solvers[1]: YALL1 [148], SpaRSA [147], and SPGL1 [137] in which we used old solutions as a "warm start" at every iteration of reweighting. We show that IRW-H outperforms YALL1, SpaRSA, and SPGL1 in terms of the computational cost for iterative reweighting, while ARW-H yields better overall performance in terms of both the computational cost and the reconstruction accuracy.

6.4.1 Experiment setup

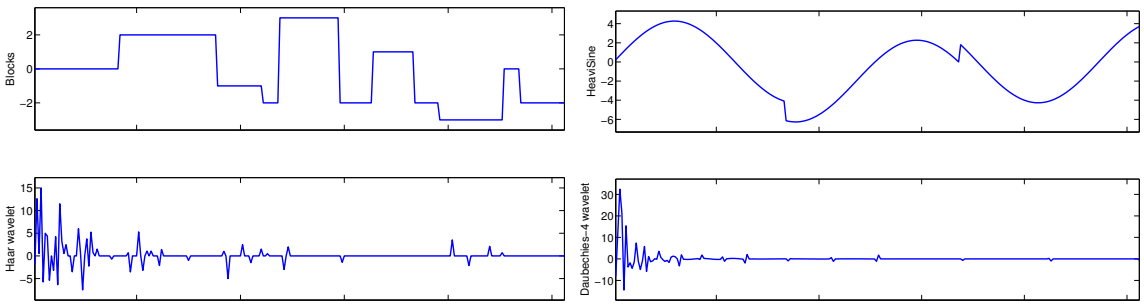
We compared the performances of the algorithms above for the recovery of two types of sparse signals from noisy, random measurements that were simulated according to the model in (6.1). We generated sparse signals by applying wavelet transforms on the modified forms of "Blocks" and "HeaviSine" signals from the Wavelab toolbox [29] as described below.

[1] We selected these solvers for comparison because we found these to be the fastest and sufficiently accurate with a warm start among the commonly used ℓ_1 solvers [1, 62, 68].

i. **Blocks**: We generated a piecewise-constant signal of length N by randomly dividing the interval $[1,\ N]$ into 11 disjoint regions. Setting the first region to zero, we iteratively assigned a random value to every region by adding an integer chosen uniformly in the range of $[-5,\ 5]$ to the value from the previous region. We applied Haar wavelet transform on the piecewise-constant signal to generate a sparse signal $\bar{x}$. An example of such a piecewise-constant signal and its Haar wavelet transform is presented in Figure 6.2a. Because of the piecewise constant structure of these signals, the resulting Haar wavelet transforms will have only a small number of nonzero coefficients that depend on the number of discontinuities and the finest wavelet scale. Since we have fixed the number of discontinuities, the ratio of the number of nonzero elements to the length of the signal becomes smaller as the length of the signal (N) increases.

ii. **HeaviSine**: We generated a sinusoidal signal with nearly two cycles and two jumps at random locations. First, we generated a sinusoidal signal of length N for which we selected the amplitude in the range of $[4,\ 6]$ and the number of cycles in the range $[2,\ 2.5]$ uniformly at random. Then, we divided the signal into three non-overlapping regions and added a different Gaussian random variable to each region. We applied Daubechies 4 wavelet transform on the resulting signal to generate the sparse signal $\bar{x}$. An example of such a sinusoidal signal with jumps and its Daubechies 4 wavelet transform is presented in Figure 6.2b. In this type of signals, most of the wavelet coefficients in $\bar{x}$ will not be exactly zero, but if sorted in the decreasing order of magnitude, the coefficients quickly decay to extremely small values. Hence, this type of signals can be classified as near-sparse or compressible.

In every experiment, we generated an $M \times N$ measurement matrix Φ with its entries drawn independently according to $\mathcal{N}(0, 1/\sqrt{M})$ distribution and added Gaussian noise vector e to generate the measurement vector as $y = \Phi\bar{x} + e$. We selected each



(a) Blocks **(b)** HeaviSine

Figure 6.2: (a) An example of piecewise-constant (blocks) signal and its sparse representation using Haar wavelets. (b) An example of perturbed HeaviSine signal and its sparse representation using Daubechies-4 wavelets.

entry in e as i.i.d. $\mathcal{N}(0, \sigma^2)$, where the variance σ^2 was selected to set the expected SNR with respect to the measurements $\Phi\bar{x}$ at 40 dB. We reconstructed the solution $\widehat{x}$ using all the algorithms according to the procedures described below.

In our experiments, we fixed the parameter $\tau = \sigma\sqrt{\log N}$, where σ denotes the standard deviation of the measurement noise. Although the weights can be tuned according to the signal structure, measurement matrix, and noise level, we did not make such an attempt in our comparison. Instead, we adopted a general rule for selecting weights that provided good overall performance for all the solvers, in all of our experiments. We set up the algorithms in the following manner.

i. **ARW-H** : We solved a weighted ℓ_1-norm minimization problem of the form (6.2) following the procedure outlined in Algorithm 3, in which the exact values for the w_i are not known a priori as they are selected adaptively. In line 3 of Algorithm 3, we selected $\tilde{w}_i \leftarrow \min\left(\tau, \frac{\tau}{\beta|x_i^*|}\right)$ using $\beta \leftarrow M\frac{\|x^*\|_2^2}{\|x^*\|_1^2}$ at every step, where x^* denotes the solution from the previous homotopy step. We selected this value of β because it helps in shrinking the w_i to smaller values when M is large and to larger values when the solution is dense. (We are using the ratio of the ℓ_1 to the ℓ_2 norm as a proxy for the support size here.) The main computational cost at every step of ARW-H involves one matrix-vector multiplication

for identifying a change in the support and a rank-one update for computing the update direction. We used the matrix inversion lemma-based scheme to perform the rank-one updates. MATLAB code is available in ℓ_1-homotopy package at http://users.ece.gatech.edu/∼sasif/homotopy.

ii. **IRW-H**: We solved (6.2) via iterative reweighting in which we updated the solution at every reweighting iteration according to the procedure outlined in Algorithm 2. For the first iteration, we used standard LASSO homotopy algorithm [11] to solve (6.2) with $w_i = \tau$ for all i. Afterwards, at every reweighting iteration, we updated the w_i as

$$w_i \leftarrow \frac{\tau}{\beta|\widehat{x}_i| + \epsilon}, \tag{6.15}$$

where $\widehat{x}$ denotes the solution from previous reweighting iteration and $\beta \geq 1$ and $\epsilon > 0$ denote two parameters that can be used to tune the weights according to the problem. In our experiments, we fixed $\epsilon = 1$ and updated $\beta \leftarrow M\frac{\|\widehat{x}\|_2^2}{\|\widehat{x}\|_1^2}$ at every reweighting iteration. The main computational cost at every step of IRW-H also involves one matrix-vector multiplication and a rank-one update of a small matrix. We used matrix inversion lemma-based scheme to perform rank-one updates.

iii. **YALL1**: YALL1 is a first-order algorithm that uses an alternating direction minimization method for solving various ℓ_1 problems, see [148] for further details. We iteratively solved (6.2) using weighted-ℓ_1/ℓ_2 solver in the YALL1 package. For the initial solution, we solved (6.2) with $w = \tau$ using YALL1. At every subsequent reweighting iteration, we used previous YALL1 solution to renew the weights according to (6.15) and solved (6.2) by providing the old solution as a warm-start to YALL1. We fixed the tolerance parameter to 10^{-4} in all the experiments. The main computational cost of every step in the YALL1 solver comes from applications of Φ and Φ^T. We used MATLAB package for YALL1

available at http://yall1.blogs.rice.edu/.

iv. **SpaRSA** : SpaRSA is also a first-order method that uses a fast variant of iterative shrinkage and thresholding for solving various ℓ_1-regularized problems, see [147] for further details. Similar to IRW-H and YALL1, we iteratively solved (6.2) using SpaRSA, while updating weights using the old solution in (6.15) and using the old solution as a warm-start at every reweighting iteration. We used the SpaRSA code with default adaptive continuation procedure in the Safeguard mode using the duality gap-based termination criterion for which we fixed the tolerance parameter to 10^{-4} and modified the code to accommodate weights in the evaluation. The main computational cost for every step in the SpaRSA solver also involves applications of Φ and Φ^T. We used MATLAB package for SpaRSA available at http://lx.it.pt/∼mtf/SpaRSA/.

v. **SPGL1**: SPGL1 solves an equivalent constrained form of (6.2) by employing a root-finding algorithm [137]. We solved the following problem using SPGL1:

$$\underset{x}{\text{minimize}} \sum_{i=1}^{N} w_i|x_i| \text{ subject to } \|\Phi x - y\|_2 \leq \lambda, \tag{6.16}$$

in which we used $\lambda = \sigma\sqrt{M}$. For the initial solution, we solved (6.16) using $w_i = 1$ for all i. At every subsequent reweighting iteration, we used the previous SPGL1 solution to renew the weights according to (6.15) (using $\tau = 1$) and solved (6.16) using the old solution as a warm start. We solved SPGL1 using default parameters with optimality tolerance set at 10^{-4}. The computational cost of every step in SPGL1 is also dominated by matrix-vector multiplications. We used MATLAB package for SPGL1 available at http://www.cs.ubc.ca/labs/scl/spgl1.

To summarize, ARW-H solves (6.2) by adaptively selecting the values of w_i, while IRW-H, YALL1, and SpaRSA iteratively solve (6.2) and SPGL1 iteratively solves (6.16), using updated values of w_i at every reweighting iteration.

We used MATLAB implementations of all the algorithms and performed all the experiments on a standard desktop computer. We used a single computational thread for all the experiments, which involved recovery of a sparse signal from a given set of measurements using all the candidate algorithms. In every experiment, we recorded three quantities for each algorithm: 1) the quality of reconstructed signal in terms of the signal-to-error ratio in dB, defined as

$$\text{SER} = 20 \log_{10} \frac{\|\bar{x}\|_2}{\|\bar{x} - \widehat{x}\|_2},$$

where $\bar{x}$ and $\widehat{x}$ denote the original and the reconstructed signal, respectively, 2) the number of matrix-vector products with Φ and Φ^T, and 3) the execution time in MATLAB.

6.4.2 Results

We compared performances of ARW-H, IRW-H, YALL1, SpaRSA, and SPGL1 for the recovery of randomly perturbed Blocks and HeaviSine signals from random, noisy measurements. We performed 100 independent trials for each of the following combinations of N and M: $N = [256,\ 512,\ 1024]$ and $M = [N/2,\ N/2.5,\ N/3,\ N/3.5,\ N/4]$. In each experiment, we recovered a solution $\widehat{x}$ from simulated noisy, random measurements using all the algorithms, according to the procedures described above, and recorded the corresponding SER, the number of matrix-vector products, and MATLAB runtime. The results, averaged over all the trials, for each combination of M and N are presented in Figures 6.3, 6.5, 6.7 (for Blocks signals) and Figures 6.4, 6.6, 6.8 (for HeaviSine signals).

Comparison of SERs for the solutions of all the algorithms at different values of N and M is presented in Figure 6.3 (for Blocks signals) and Figure 6.4 (for HeaviSine signals). Three plots in the first row depict SERs for the solutions after first iteration of all the algorithms. Since ARW-H solves a weighted ℓ_1-norm formulation (as in (6.2)) via adaptive reweighting, its performance is superior to all the other algorithms,

which solve unweighted ℓ_1-norm problems in their first iteration. Since SPGL1 solves the ℓ_1 problem in (6.16), its performance is slightly different compared to IRW-H, YALL1, and SpaRSA, all of which solve (6.2) and should provide identical solutions if they converge properly. The plots in the second row present SERs for the solutions after five reweighting iterations of all the algorithms except ARW-H, which was solved only once. As we can see that the solutions of ARW-H display the best SERs in all these experiments. Although SERs for the solutions of IRW-H, YALL1, SpaRSA, and SPGL1 improve with iterative reweighting, in some cases there is a significant gap between their SERs and that of ARW-H.

Comparison of the computational cost of all the algorithms in terms of the number of matrix-vector multiplications is presented in Figure 6.5 (for Blocks signals) and Figure 6.6 (for HeaviSine signals). We counted an application of each Φ and Φ^T as one count of AtA. For the homotopy algorithms, we approximated the cost of one step as one application of $\Phi^T\Phi$. Three plots in the first row present the count of $\Phi^T\Phi$ applications that each algorithm used for computing the initial solution. Second row depicts the count of $\Phi^T\Phi$ applications summed over five reweighting iterations in each of the algorithms. Since we solved ARW-H just once, the count for ARW-H is zero and does not appear in the second row. Third row presents total count of $\Phi^T\Phi$ applications, which is the sum of the counts in the first and the second row. We can see in the second row that, compared to YALL1, SPGL1, and SpaRSA, IRW-H used a distinctly smaller number of matrix-vector products for updating the solution as the weights change in iterative reweighting. The final count in the third row shows that ARW-H consumed the least number of total $\Phi^T\Phi$ applications in all the cases.

Comparison of MATLAB runtime for all the algorithms is presented in Figure 6.7 (for Blocks signals) and Figure 6.8 (for HeaviSine signals). The first row presents runtime that each algorithm utilized for computing the initial solution, the second row presents execution time for five reweighting iterations, and the third row presents

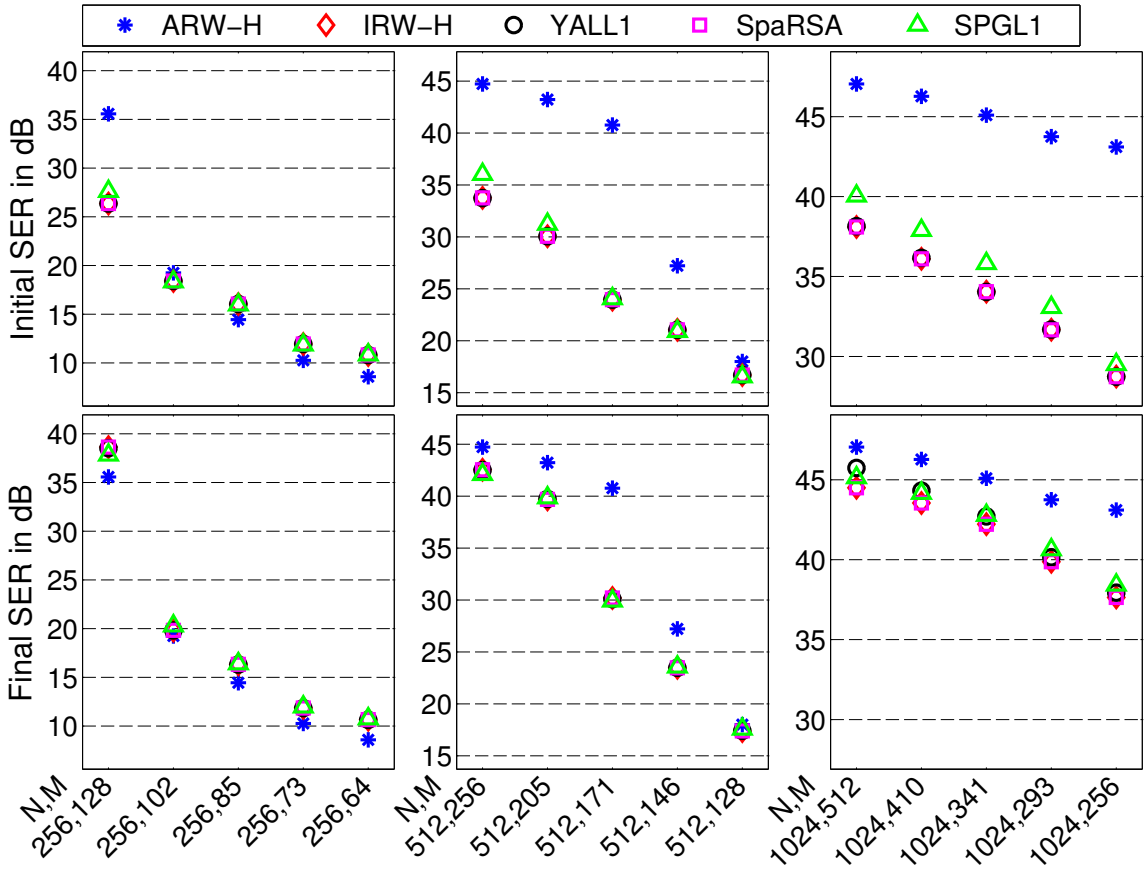


Figure 6.3: Comparison of SER for the recovery of sparse signals that were constructed by taking Haar wavelet transform of randomly perturbed "Blocks" signals and measured with $M \times N$ Gaussian matrices in the presence of Gaussian measurement noise at 40 dB SNR. ARW-H solves adaptive-reweighted ℓ_1 problem once, while other methods solve unweighted ℓ_1 problem in their first iteration and perform five reweighted iterations afterwards. **(First row)** SER for the solution after first iteration. **(Second row)** SER for solutions after five reweighting iterations (SER for ARW-H is copied from the top row)

total time consumed by each of the recovery algorithms. As we can see in the second row that, compared to YALL1, SPGL1, and SpaRSA, IRW-H consumed distinctly lesser time for updating solutions in iterative reweighting. In the third row, we see small difference in the total runtime for IRW-H, SpaRSA, and YALL1, where IRW-H and SpaRSA display comparable performance. Nevertheless, in all the experiments, the total runtime for ARW-H is the smallest among all the algorithms.

A brief summary of the results for our experiments is as follows. We observed that

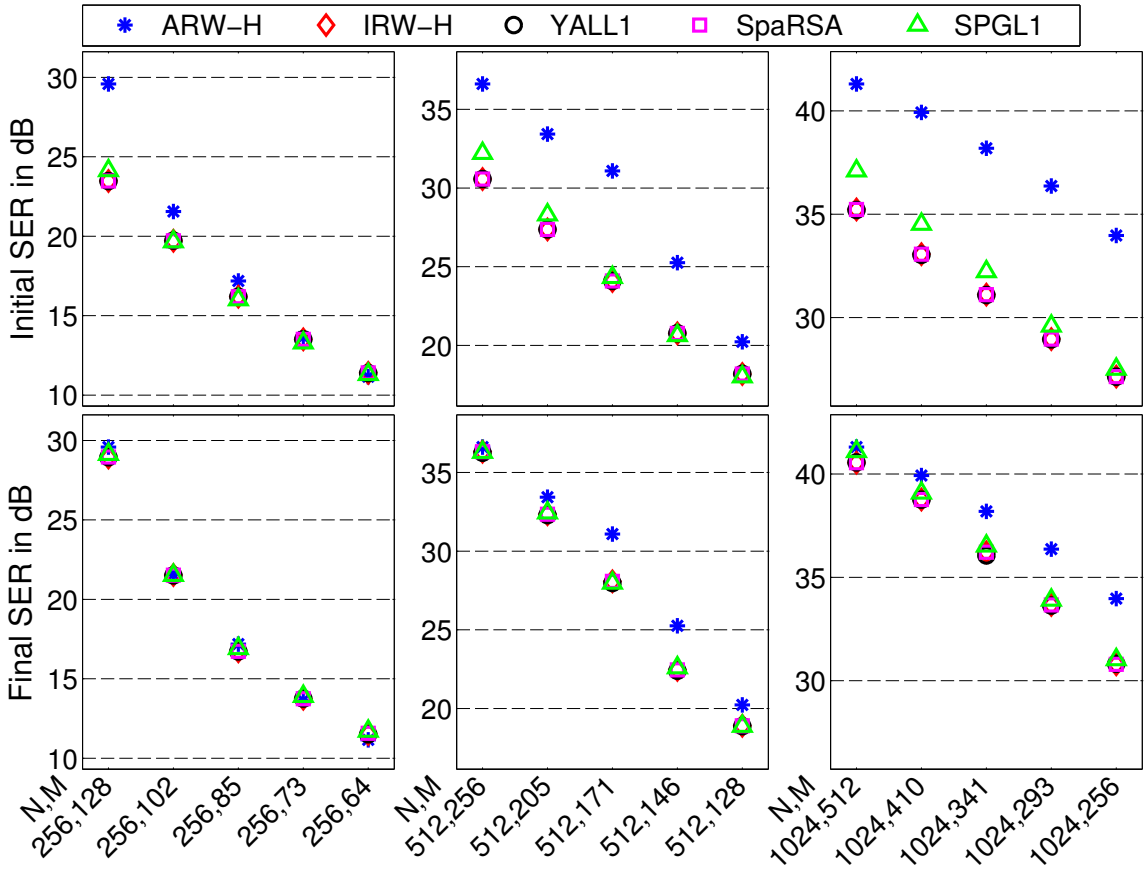


Figure 6.4: Comparison of SER for the recovery of near-sparse signals that were constructed by taking Daubechies 4 wavelet transform of randomly perturbed HeaviSine signals and measured with $M \times N$ Gaussian matrices in the presence of Gaussian noise at 40 dB SNR. ARW-H solves adaptive-reweighted ℓ_1 problem once, while other methods solve unweighted ℓ_1 problem in their first iteration and perform five reweighted iterations afterwards. **(First row)** SER for the solution after first iteration. **(Second row)** SER for solutions after five reweighting iterations (SER for ARW-H is copied from the top row)

the adaptive reweighting scheme (ARW-H) recovered signals with better SERs compared to the iterative reweighting schemes (IRW-H, SpaRSA, YALL1, and SPGL1), and it does so by solving a single homotopy problem at the expense of a small amount of computational cost and time. Among the iterative reweighting schemes, IRW-H quickly updated the solutions during iterative reweighting at the expense of marginal computational cost and time, which are distinctly smaller than the respective costs and times for SpaRSA, YALL1, and SPGL1; although SpaRSA and YALL1 with warm-start provided competitive results for longer signals.

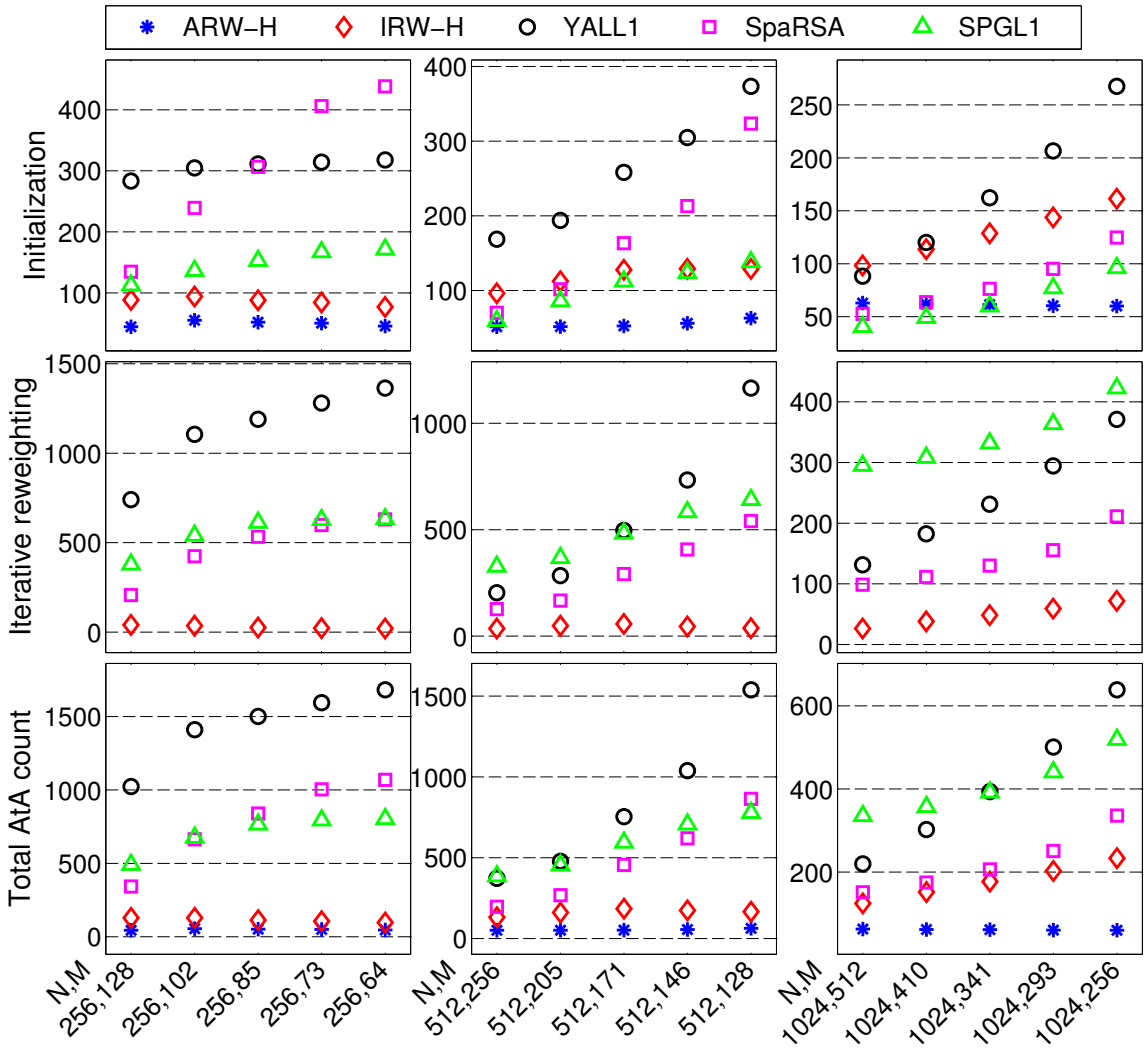


Figure 6.5: Comparison of the number of matrix-vector products for the recovery of Blocks signals. ARW-H solves adaptive-reweighted ℓ_1 problem once, while other methods solve unweighted ℓ_1 problem in their first iteration and perform five reweighted iterations afterwards. **(First row)** Count for the first iteration only. **(Second row)** Count for all the reweighting iterations (ARW-H does not appear because its count is zero). **(Third row)** Count for all the iterations.

6.5 Discussion

We presented two homotopy algorithms that can efficiently solve reweighted ℓ_1 problems. In Section 6.2, we presented an algorithm for updating the solution of (6.2) as the w_i change. We demonstrated with experiments that, in reweighting iterations, our proposed algorithm quickly updates the solution at a small computational expense. In Section 6.3, we presented a homotopy algorithm that adaptively selects weights inside

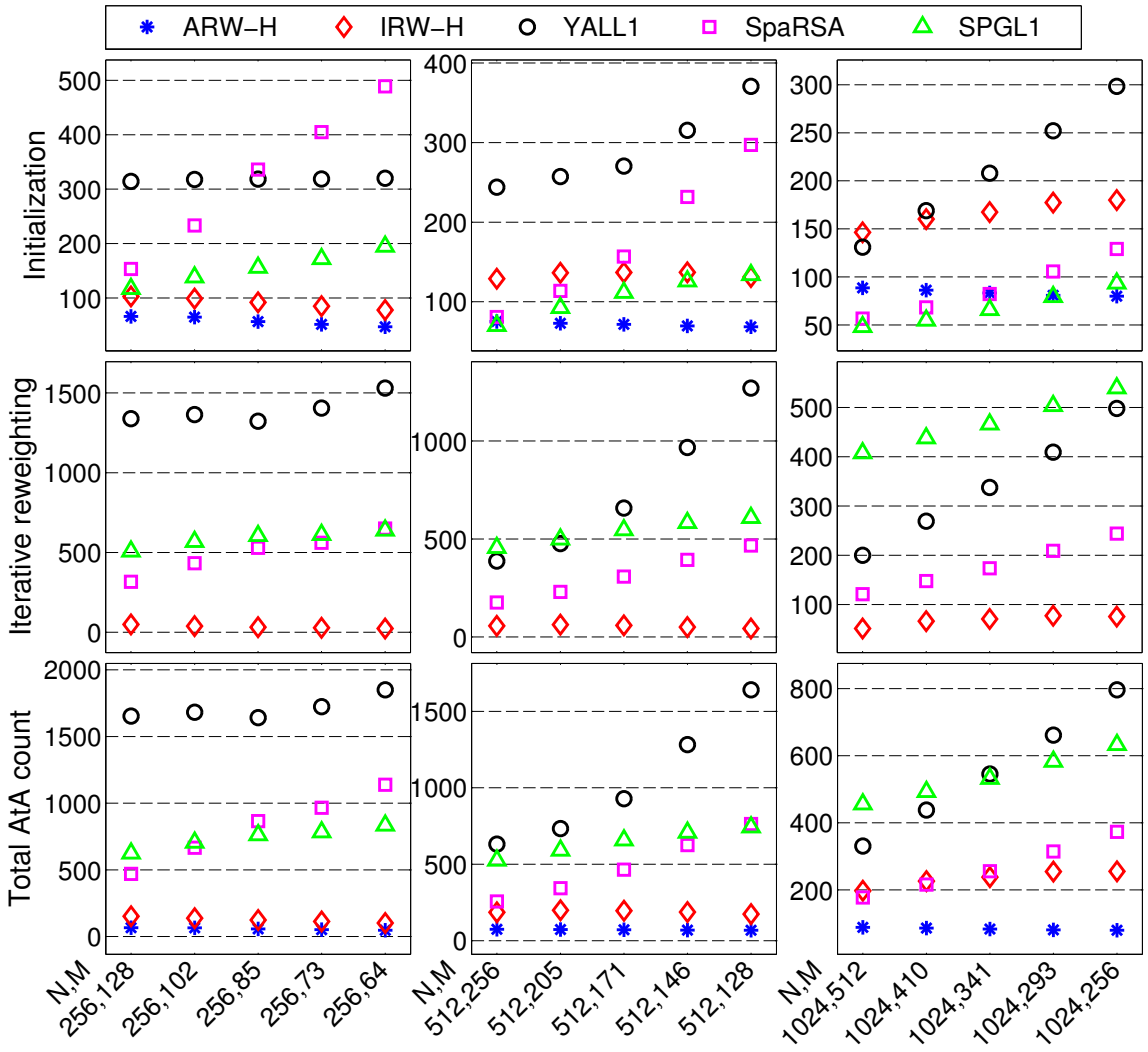


Figure 6.6: Comparison of the number of matrix-vector products for the recovery of HeaviSine signals. ARW-H solves adaptive-reweighted ℓ_1 problem once, while other methods solve unweighted ℓ_1 problem in their first iteration and perform five reweighted iterations afterwards. **(First row)** Count for the first iteration only. **(Second row)** Count for all the reweighting iterations (ARW-H does not appear because its count is zero). **(Third row)** Count for all the iterations.

a single homotopy program. We demonstrated with experiments that our proposed adaptive reweighting method outperforms iterative reweighting methods in terms of the reconstruction quality and the computational cost.

We would like to mention that an adaptive reweighting scheme, similar to the

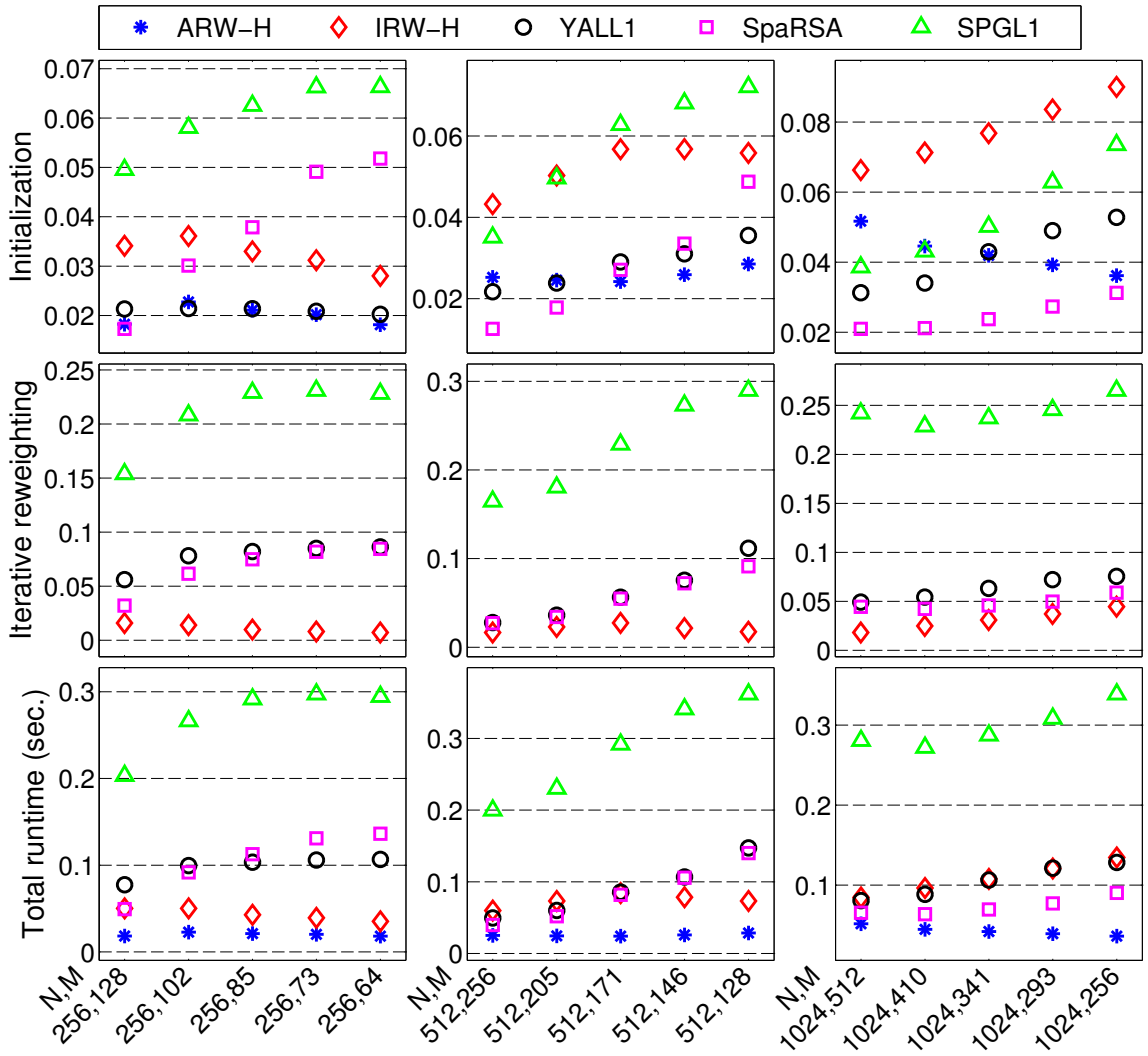


Figure 6.7: Comparison of MATLAB runtime for the recovery of Blocks signals. ARW-H solves adaptive-reweighted ℓ_1 problem once, while other methods solve unweighted ℓ_1 problem in their first iteration and perform five reweighted iterations afterwards. **(First row)** Time for the first iteration. **(Second row)** Time for all the reweighted iterations. **(Third row)** Total runtime.

one we presented for homotopy, can also be embedded inside iterative shrinkage-thresholding algorithms [18, 22, 52, 68, 147]. A recent paper [96] has also employed a similar principle of reweighting inside SPGL1. Standard iterative shrinkage-thresholding algorithms solve the program in (6.2) by solving the following shrinkage problem at every inner iteration:

$$\underset{x}{\text{minimize}} \; \frac{L}{2}\|x-u\|_2^2 + \sum w_i|x_i|, \tag{6.17}$$

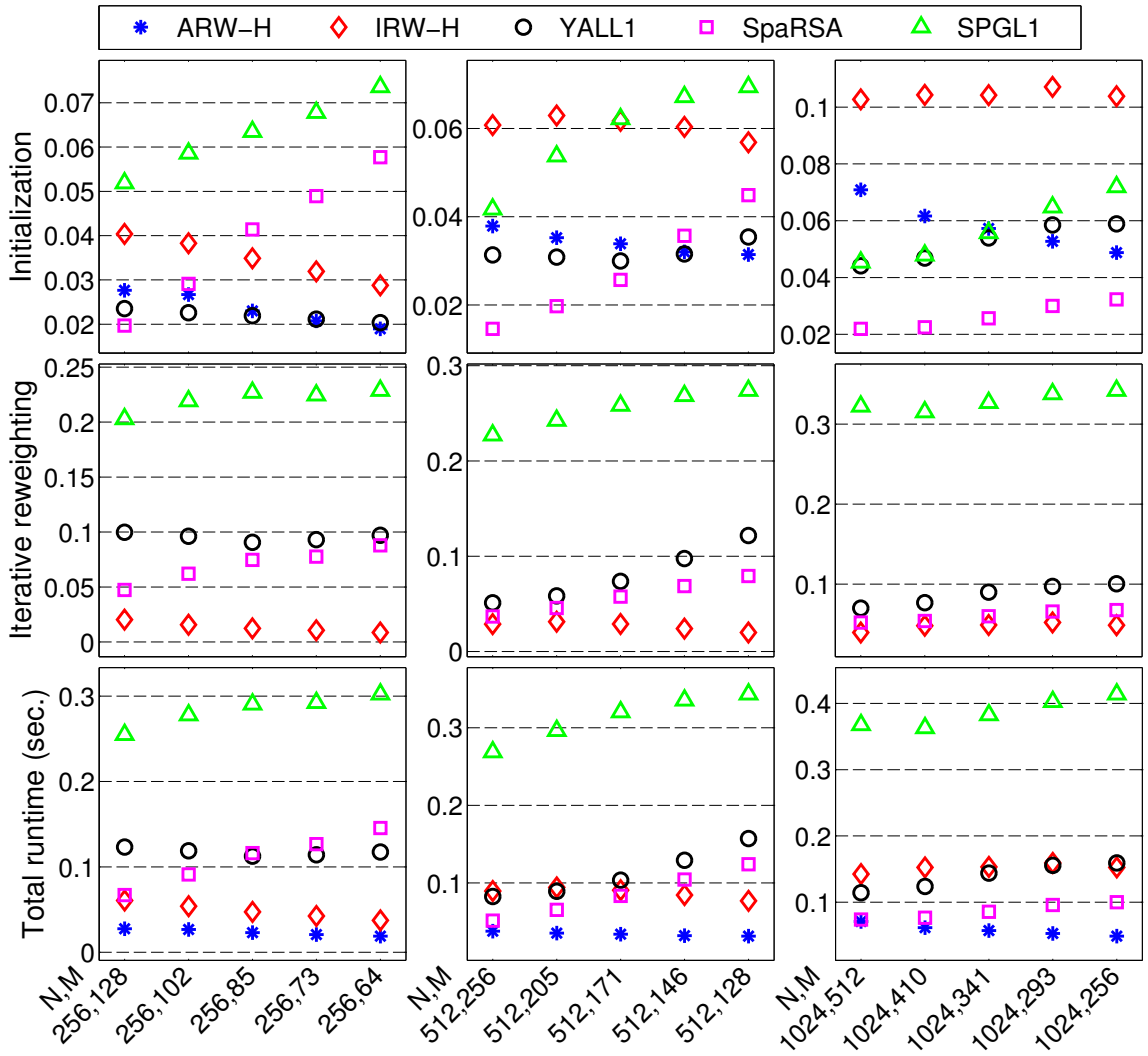


Figure 6.8: Comparison of MATLAB runtime for the recovery of HeaviSine signals. **(First row)** Time for the first iteration. ARW-H solves adaptive-reweighted ℓ_1 problem once, while other methods solve unweighted ℓ_1 problem in their first iteration and perform five reweighted iterations afterwards. **(Second row)** Time for all the reweighted iterations. **(Third row)** Total runtime.

where $u = x^{k-1} - \frac{1}{L}\Phi^T(\Phi x^{k-1} - y)$ denotes a vector that is generated using a solution x^{k-1} from a previous iteration and L determines the stepsize. The solution of (6.17) involves a simple soft-thresholding of the entries in u with respect to the w_i/L (i.e., $x_i = \text{soft}(u_i, w_i/L)$, where $\text{soft}(u, \alpha) \equiv \text{sign}(u)\max\{|u| - \alpha, 0\}$ defines the soft-thresholding/shrinkage function). To embed adaptive reweighting inside such shrinkage algorithms, instead of using a fixed set of w_i for soft-thresholding in (6.17)

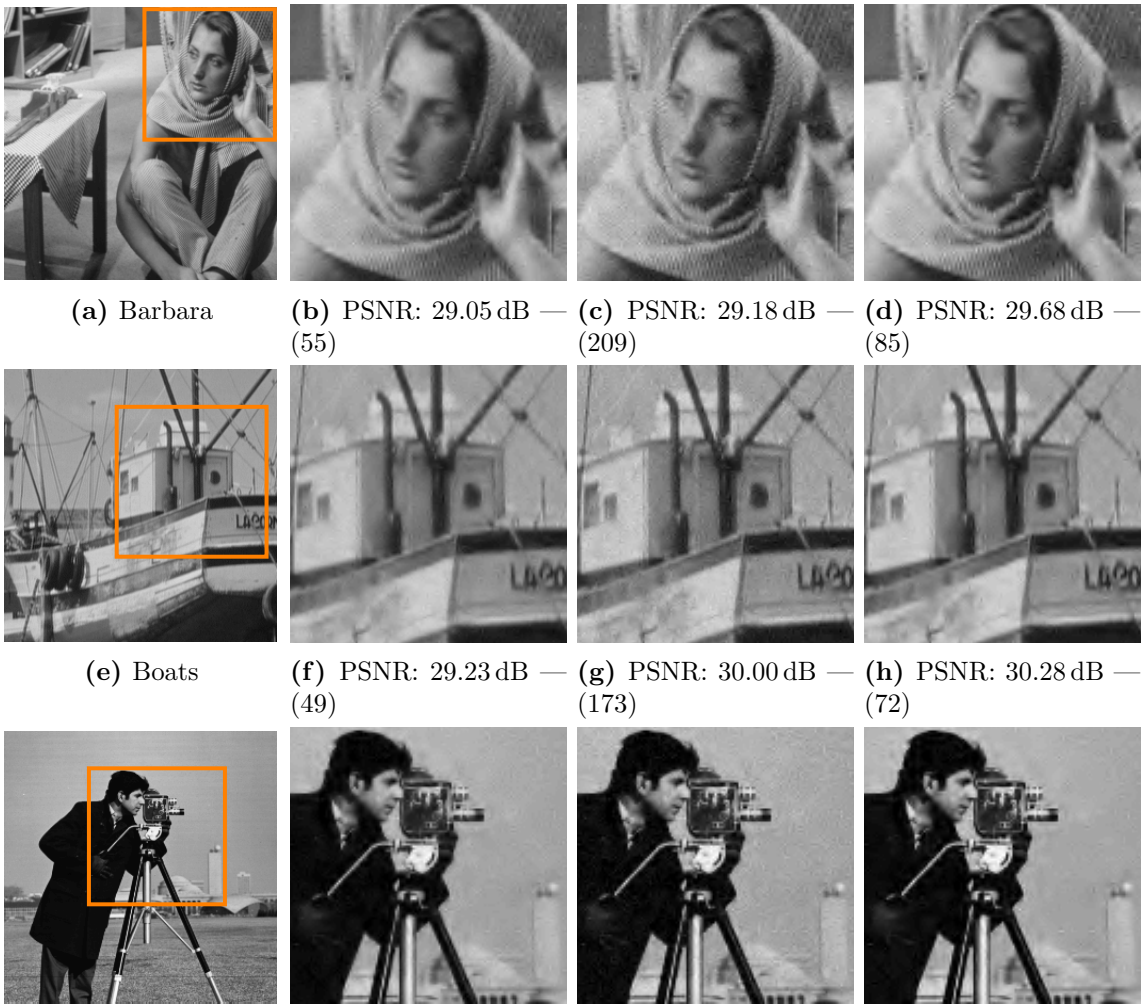
at every iteration, we can adaptively select the w_i according to the changes in the solution.

To examine our suggestion, we added an adaptive reweighting scheme in the source code of SpaRSA. SpaRSA offers a feature for adaptive continuation in which it starts the shrinkage parameter τ with a large value and decreases it towards the desired value after every few iterations. We added an extra line in the code that uses the available solution $\widehat{x}$ to update each w_i as $\min(\tau, \tau/\beta|\widehat{x}_i|)$ whenever τ changes. We gauged the performance of this modified method by using it for the recovery of gray-scale images from compressive measurements. The problem formulation following the model in (6.1) is as follows. We generated a sparse signal $\bar{x}$ of length N by applying a Daubechies 9/7 biorthogonal wavelet transform [48, 94] with odd-symmetric extension on an image, selected Φ as a subsampled noiselet transform [49], and added Gaussian noise e in the measurements by selecting each entry in e as i.i.d. $\mathcal{N}(0, \sigma^2)$.

In Figure 6.9 we present results averaged over 10 experiments that we performed for the recovery of three 256×256 images from $M = 30,000$ noiselet measurements in the presence of noise at 40 dB SNR using SpaRSA in three different ways. The first column presents the original 256×256 images. The second column presents small portions of the reconstructed images. We reconstructed the images by solving the standard ℓ_1 problem in (3.3) using $\tau = \sigma\sqrt{\log N}$. The peak signal-to-noise ratio (PSNR) for the entire reconstructed image is presented in each caption along with the total count for the number of applications of $\Phi^T\Phi$ (in parentheses) averaged over 10 experiments. The third column presents portions of the reconstructed images after three reweighting iterations using the warm-start and weight selection procedure employed in the experiments in Section 6.4. The last column presents portions of the reconstructed images by solving the modified version of SpaRSA in which we initialized the SpaRSA code by setting all the weights to a same value, and after every continuation iteration we modified the values of weights according to the available

solution. We observe that SpaRSA with this adaptive reweighting modification yields better performance in terms of PSNR over the other two methods, while its computational cost is significantly smaller than the cost of iterative reweighting (in column 2).

We have presented these results as a proof-of-concept, and we anticipate that adding a simple adaptive reweighting scheme within existing iterative shrinkage algorithms can potentially enhance their performance in many scenarios without any additional cost; however, a detailed study in this regard is beyond the scope of this chapter.



(a) Barbara **(b)** PSNR: 29.05 dB — (55) **(c)** PSNR: 29.18 dB — (209) **(d)** PSNR: 29.68 dB — (85)

(e) Boats **(f)** PSNR: 29.23 dB — (49) **(g)** PSNR: 30.00 dB — (173) **(h)** PSNR: 30.28 dB — (72)

(i) Cameraman **(j)** PSNR: 29.82 dB — (52) **(k)** PSNR: 30.27 dB — (193) **(l)** PSNR: 30.77 dB — (78)

Figure 6.9: Results for the recovery of 256×256 images from $M = 30,000$ noiselet measurements in the presence of Gaussian noise at 40dB SNR, using Daubechies 9/7 biorthogonal wavelet transform with odd-symmetric extensions as the sparse representation. **(Column 1)** Original images. **(Column 2)** Portions of the images (inside the orange box) reconstructed by solving (3.3) using SpaRSA. **(Column 3)** Reconstruction after three reweighting iterations. **(Column 4)** Adaptive reweighting by updating the w_i after every continuation step in SpaRSA. The caption under each subimage shows the PSNR over the entire reconstructed image and a count for the number of applications of $\Phi^T\Phi$ (in parentheses) averaged over 10 experiments.

CHAPTER VII

SPARSE RECOVERY FROM STREAMING SYSTEMS

In this chapter we discuss the problem of estimating a sparse, time-varying signal from streaming measurements. Most of the existing sparse recovery methods assume a *static* system in which the unknown signal is a finite-length vector for which a fixed set of linear measurements and a representation basis are available and an ℓ_1-norm minimization program is solved for the signal reconstruction. However, the same representation and reconstruction framework is not readily applicable in a *streaming* system in which the unknown signal varies over time and has no clear beginning and end. Instead of measuring the entire signal or processing the entire set of measurements at once, these tasks are performed sequentially over short, shifting time intervals. A streaming framework for the reconstruction is particularly desired when dividing a streaming signal into disjoint, finite-length blocks and processing each block independently is either infeasible or inefficient.

We consider a streaming system in which measurements of a time-varying signal are recorded over short, possibly overlapping, intervals. We iteratively process a small number of measurements over a sliding, active interval and solve a weighted ℓ_1-norm minimization problem for estimating sparse coefficients. For the sparse representation of the time-varying signals, instead of using block transforms, we use lapped orthogonal transforms. Since we estimate overlapping portions of the streaming signal while adding and removing measurements, instead of solving a new ℓ_1 program from scratch at every iteration, we use an available signal estimate as the starting point (warm-start) in a homotopy formulation and update the solution in a small number of computationally inexpensive homotopy steps. We demonstrate the performance of

our proposed streaming recovery algorithm for various time-varying signals.

7.1 Introduction

We consider the following time-varying linear observation model for a discrete-time signal $x[n]$:

$$y_t = \Phi_t x_t + e_t, \tag{7.1}$$

where x_t is a vector that represents $x[n]$ over an interval of time, y_t is a vector that contains measurements of x_t, Φ_t is a measurement matrix, and e_t is noise in the measurements. We use the subscript t to indicate that the system in (7.1) represents a small part of an infinite-dimensional streaming system, in which for any t, x_t precedes x_{t+1} in $x[n]$ and the two may overlap. If we treat the x_t independent from the rest of the streaming signal ($x[n]$), we can solve (7.1) as a stand-alone system for every t as follows. Suppose we can represent each x_t as $\Psi_t \alpha_t$, where Ψ_t denotes a representation matrix (e.g., a discrete cosine or a wavelet transform) for which α_t is a sparse vector of transform coefficients. We write the equivalent system for (7.1) as

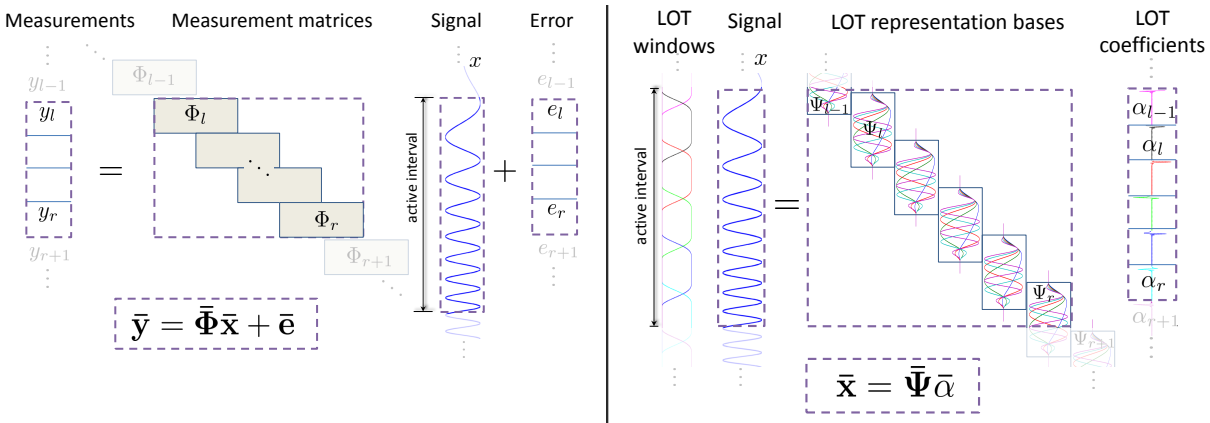
$$y_t = \Phi_t \Psi_t \alpha_t + e_t, \tag{7.2}$$

and solve the following weighted ℓ_1-norm minimization problem for a sparse estimate of α_t:

$$\underset{\alpha_t}{\text{minimize}} \ \|W_t \alpha_t\|_1 + \frac{1}{2}\|\Phi_t \Psi_t \alpha_t - y_t\|_2^2. \tag{7.3}$$

The ℓ_1 term promotes sparsity in the estimated coefficients; W_t is a diagonal matrix of positive weights that can be adapted to promote a certain sparse structure in the solution [42, 152]; and the ℓ_2 term ensures that the solution remains close to the measurements.

The method described above represents and reconstructs the signal blocks (x_t) independently, which is natural if both the measurement system in (7.1) and the representation system in (7.2) are block-diagonal; that is, the x_t are non-overlapping



(a) Overlapping measurements of a signal.

(b) LOT-based representation of a signal.

Figure 7.1: Illustration of an overlapping measurement system **(a)** and a lapped orthogonal transform (LOT)-based representation system **(b)**. Boxed regions represent the system over the active interval. Subscripts l and r indicate the left and the right border of the active interval.

in (7.1) and each x_t is represented as a sparse vector using a block transform in (7.2). However, estimating the x_t independently is not optimal if the streaming system for (7.1) or (7.2) is not block diagonal, which can happen if Φ_t, Ψ_t, or both of them overlap across the x_t. An illustration of such an overlapping measurement and representation system is presented in Figure 7.1. Figure 7.1a depicts a measurement system in which the Φ_t overlap (the x_t, which are not labeled in the figure, are the overlapping portions of $x[n]$ that constitute the y_t). Figure 7.1b depicts a representation of $x[n]$ using lapped orthogonal transform (LOT) bases [94, 95] in which the Ψ_t overlap (and multiple $\Psi_t\alpha_t$ may add up to constitute a portion of $x[n]$).

In this chapter we present ℓ_1-norm minimization based algorithms for the sparse recovery of smooth, time-varying signals from streaming measurements in (7.1) using sparse representation bases with compact but overlapping supports. We assume that the sets of measurements are sequentially recorded over short, shifting (possibly overlapping) intervals of the streaming signal. Instead of estimating each block (x_t) independently, we iteratively estimate the signal ($x[n]$) over small, sliding intervals, which allows us to link together the blocks that share information. At every iteration,

we build a system model that describes the measurements and the sparse coefficients of the streaming signal over an active interval (one such example is depicted in Figure 7.1). We estimate the sparse coefficients for the signal over the active interval by solving a weighted ℓ_1-norm minimization problem. Before solving the optimization problem, an estimate of the signal over the active interval is either available from the previous iteration or it can be predicted. We use the available signal estimate to aide the recovery process in two ways: We update the W_t using available estimates of the α_t (in the same spirit as iterative reweighting [42]), and we use the available estimates of the α_t as a starting point to expedite the solution of the ℓ_1 problem. In particular, we focus on the ℓ_1 homotopy for dynamic updating of the recovery problem as the underlying signal and the measurements change.

The chapter is organized as follows. We discuss signal representation in Section 7.2 and the recovery framework for the streaming system in Section 7.3. We present experimental results to demonstrate the performance of our algorithms, in terms of the quality of reconstructed signals and the computational cost of the recovery algorithm, in Section 7.4.

7.2 Signal representation in compactly supported bases

We will represent a discrete-time signal $x[n]$ as

$$x[n] = \sum_{p \in \mathbb{Z}} \sum_{0 \leq k < l_p} \alpha_{p,k} \psi_{p,k}[n], \tag{7.4}$$

where the set of functions $\psi_{p,k}$ forms an orthogonal basis of $\ell_2(\mathbb{Z})$ and the $\alpha_{p,k} = \langle \psi_{p,k}, x \rangle$ denote the corresponding basis coefficients that we expect to be sparse or compressible. For a fixed $p \in \mathbb{Z}$, $\{\psi_{p,k}\}_k$ denotes a set of orthogonal basis vectors that have a compact support over an interval I_p. The supports of the $\psi_{p,k}$ and the $\psi_{p',k}$ (i.e., I_p and $I_{p'}$) may overlap if $p \neq p'$. An example of such a signal representation using lapped orthogonal bases is depicted in Figure 7.1b, where a Ψ_p denotes the

basis functions in $\{\psi_{p,k}\}_k$ supported on I_p, an α_p denotes the respective $\{\alpha_{p,k}\}_k$, and the overlapping windows denote the intervals I_p.

A lapped orthogonal transform (LOT) decomposes a signal into orthogonal components with compact, overlapping supports [95]. Orthogonality between the components in the overlapping regions is maintained due to projections with opposite (i.e., even and odd) symmetries. LOT basis functions can be designed using modified cosine-IV basis functions that are multiplied by smooth, overlapping windows. The advantage of using the LOT instead of a simple block-based discrete cosine or Fourier transform is that block-based transforms use rectangular windows to divide a signal into disjoint blocks and that can introduce artificial discontinuities at the boundaries of the blocks and ruin the sparsity [94].

A discrete LOT basis can be designed as follows. Divide the support of the signal into consecutive, overlapping intervals $I_p = [a_p - \eta_p, a_{p+1} + \eta_{p+1}]$, where $\{a_p\}_{p\in\mathbb{Z}}$ is a sequence of half integers (i.e., $a_p + 1/2 \in \mathbb{Z}$) and $\{\eta_p\}_{p\in\mathbb{Z}}$ is a sequence of transition width parameters such that $l_p \stackrel{\text{def}}{=} a_{p+1} - a_p \geq \eta_p + \eta_{p+1}$. The LOT basis function, $\psi_{p,k}$ in (7.4), for every p, k is defined as

$$\psi_{p,k}[n] = g_p[n]\sqrt{\frac{2}{l_p}}\cos\left[\pi\left(k+\frac{1}{2}\right)\frac{n-a_p}{l_p}\right], \tag{7.5}$$

which is a translated and dilated cosine-IV basis function, multiplied by a smooth window g_p that is supported on I_p. For a careful choice of g_p, coupled with the even and the odd symmetries of cosine-IV basis functions with respect to a_p and a_{p+1}, respectively, the set of functions $\psi_{p,k}$ forms an orthonormal basis of $\ell_2(\mathbb{Z})$ (see [94, Sec. 8.4] for further details).

To compute the LOT coefficients of $x[n]$ over an arbitrary interval Π, we assume a partition of Π into appropriate LOT subintervals I_p. Figure 7.2a depicts an example with such a partition of a time interval using LOT windows and the decomposition of a linear chirp signal into overlapping components using LOT bases and their respective coefficients. Since the set of functions $\{\psi_{p,k}\}_k$ defines an orthogonal basis for a LOT

subspace on respective I_p, the corresponding LOT projection of $x[n]$ can be written as

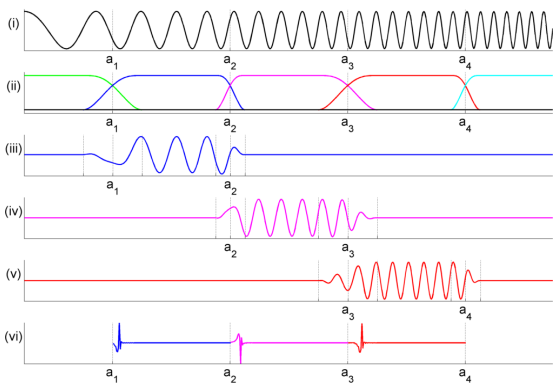
$$\widetilde{x}_p[n] = \sum_{k=0}^{l_p-1} \underbrace{\langle x, \psi_{p,k} \rangle}_{\alpha_{p,k}} \psi_{p,k}[n], \tag{7.6}$$

where $\widetilde{x}_p[n]$ is supported on I_p. We can represent the restriction of $\widetilde{x}_p[n]$ on I_p as $\Psi_p \alpha_p$, where Ψ_p is a synthesis matrix whose k^{th} column consists of $\psi_{p,k}[n]$ restricted to I_p and α_p is an l_p-length vector of LOT coefficients that consists of $\{\alpha_{p,k}\}_k$ for $0 \leq k < l_p$. Note that the $\widetilde{x}_p[n]$ are the overlapping, orthogonal components of $x[n]$, and to synthesize $x[n]$ over Π, we have to add all the $\widetilde{x}_p[n]$ that overlap Π. Referring to Figure 7.1b, $\bar{\mathbf{x}}$ denotes $x[n]$ over the active interval Π, $\bar{\alpha}$ denotes a vector that contains all the α_p that contribute to $\bar{\mathbf{x}}$, stacked on top of one another, $\bar{\mathbf{\Psi}}$ contains the corresponding Ψ_p (in part or full) at appropriate columns and rows, and the columns of Ψ_p and Ψ_{p+1} overlap in $2\eta_{p+1}$ rows.

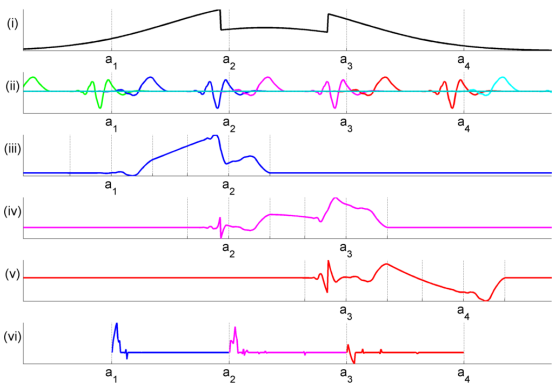
Another example of an orthogonal basis that can be naturally separated into overlapping, compact intervals is the wavelet transform. A wavelet transform decomposes a signal into orthogonal components with compact, overlapping supports at different resolutions in time and frequency [94, 141]. The scaling and wavelet functions used for this purpose overlap one another while maintaining orthogonality. Although commonly used filter-bank implementations assume that the finite-length signals are symmetrically or periodically extended during convolution, which yields a block-based wavelet transform, we can write wavelet bases in terms of shifted, dilated wavelet and scaling functions that overlap across adjacent blocks. Figure 7.2b depicts an example of the decomposition of a piece-wise smooth signal into overlapping components using wavelet bases and their respective coefficients.

7.3 Sparse recovery from overlapping systems

In a streaming system, we iteratively estimate sparse coefficients of the signal over an active, sliding interval. We describe a system for the measurements and the sparse



(a) LOT projections and coefficients. (i) A discrete-time linear chirp signal ($x[n]$). (ii) LOT windows over different intervals (I_p); distance between dotted lines around a_p represent η_p. (iii–v) LOT projections $\tilde{x}[n]$ over respective intervals. (vi) Sparse coefficients ($\alpha_{p,k}$).



(b) Wavelet projections and coefficients. (i) A piecewise smooth signal ($x[n]$). (ii) A subset of scaling and wavelet functions at the coarsest scale. Dotted lines denoteI_p. (iii–v) Wavelet projections $\tilde{x}[n]$ over respective intervals. (vi) Sparse coefficients ($\alpha_{p,k}$).

Figure 7.2: Signal decomposition in **(a)** LOT and **(b)** wavelet bases.

representation of the signal over the active interval and solve a weighted ℓ_1-norm minimization problem for estimating the sparse coefficients. At every iteration of the streaming recovery process, we shift the active interval by removing a few oldest measurements and adding a few new ones in the system. Estimates of the sparse coefficients and the signal portion that leave the active interval are committed to the output. The length of the active interval determines the delay, memory, and computational complexity of the system.

7.3.1 System model

Consider the linear system in (7.1): $y_t = \Phi_t x_t + e_t$, where x_t denotes a portion of $x[n]$ over a short interval and the consecutive x_t may also overlap. We denote $x[n]$ over the active interval Π as $\bar{\mathbf{x}}$ and assume that $\bar{\mathbf{x}}$ consists of a small number of x_t. We describe the equivalent system for $\bar{\mathbf{x}}$ in the following compact form:

$$\bar{\mathbf{y}} = \bar{\mathbf{\Phi}}\bar{\mathbf{x}} + \bar{\mathbf{e}}, \tag{7.7}$$

where $\bar{\mathbf{y}}$ denotes a vector that contains y_t for the x_t that belong to $\bar{\mathbf{x}}$, $\bar{\mathbf{\Phi}}$ denotes a matrix that contains the corresponding Φ_t, and $\bar{\mathbf{e}}$ denotes the noise vector. At every iteration of the streaming recovery algorithm, we shift Π by removing the oldest y_t in the system and adding a new one and update the system in (7.7) accordingly. An example of such a measurement system in depicted in Figure 7.1a, where the active system is represented in a boxed region. To represent the signal $\bar{\mathbf{x}}$ using the model in (7.4), we use the following compact form:

$$\bar{\mathbf{x}} = \bar{\mathbf{\Psi}}\bar{\alpha}, \tag{7.8}$$

where $\bar{\alpha}$ contains the $\alpha_{p,k}$ that synthesize $\bar{\mathbf{x}}$ and the synthesis matrix $\bar{\mathbf{\Psi}}$ contains the corresponding $\psi_{p,k}$ restricted to Π as its columns. An example of such a representation system in depicted in Figure 7.1b. Using lapped orthogonal bases for the signal representation in (7.8), we describe the system in (7.7) for the active interval Π in the following equivalent form:

$$\bar{\mathbf{y}} = \bar{\mathbf{\Phi}}\bar{\mathbf{\Psi}}\bar{\alpha} + \bar{\mathbf{e}}. \tag{7.9}$$

Note that even when $\bar{\mathbf{\Phi}}$ is a block diagonal matrix, the system in (7.9) cannot be separated into independent blocks if $\bar{\mathbf{\Psi}}$ has overlapping columns.

One important consideration in our system is the design of $\bar{\mathbf{\Psi}}$ with respect to the decomposition of Π into overlapping intervals I_p. Our motivation is to have as few unknown coefficients in $\bar{\alpha}$ as possible. Note that if an interval I_p overlaps with Π (partially or fully), we have to include its corresponding coefficient vector α_p of length l_p into $\bar{\alpha}$. Since we can divide the interior of Π in an arbitrary fashion, the special consideration is only for the I_p that partially overlap with Π on its left and right borders.

On the right end of Π, we align the right-most interval, say I_r, such that it partially overlaps with Π but the interval after that, say I_{r+1}, lies completely outside Π. In such a case $\bar{\alpha}$ would contain α_r but not α_{r+1}. Such a relationship between the active

interval (Π) and the subintervals (I_p) is depicted in Figure 7.1b, where we adjusted the right-most interval such that the overlapping region on its right side lies outside the active interval Π.

On the left end of Π, we align the left-most interval, say I_l, such that it is fully included in Π. However, in such a setting I_{l-1} will partially overlap with Π and the corresponding coefficient vector α_{l-1} of length l_{l-1} will be included in $\bar{\alpha}$. Suppose we have committed the estimate of α_{l-1} to the output, and we want to remove it from the system in (7.9). If the system in (7.9) were block-diagonal, we could simply update the system by removing α_{l-1} from $\bar{\alpha}$ and the corresponding rows from $\bar{\mathbf{y}}$ and $\bar{\mathbf{\Phi}}\bar{\mathbf{\Psi}}$. But if the system in (7.9) has overlapping rows, where the rows are coupled with more than one set of variables, instead of removing the rows, we remove the columns. Thus, removing α_{l-1} is equivalent to removing the first l_{l-1} coefficients from the vector $\bar{\alpha}$, removing the first l_{l-1} columns from the matrix $\bar{\mathbf{\Phi}}\bar{\mathbf{\Psi}}$ in (7.9), and modifying the measurement vector $\bar{\mathbf{y}}$ accordingly. To do this we divide $\bar{\mathbf{x}}$ into two components as

$$\bar{\mathbf{x}} = \bar{\mathbf{\Psi}}\bar{\alpha} = \begin{bmatrix} \breve{\mathbf{\Psi}} & \tilde{\mathbf{\Psi}} \end{bmatrix} \begin{bmatrix} \breve{\alpha} \\ \tilde{\alpha} \end{bmatrix} = \breve{\mathbf{\Psi}}\breve{\alpha} + \tilde{\mathbf{\Psi}}\tilde{\alpha}, \tag{7.10}$$

where we divided $\bar{\mathbf{\Psi}}$ into two matrices $\breve{\mathbf{\Psi}}$ and $\tilde{\mathbf{\Psi}}$ and $\bar{\alpha}$ into the corresponding vectors $\breve{\alpha}$ and $\tilde{\alpha}$. An example of such a decomposition is depicted in Figure 7.3b. To remove α_{l-1} from the system in (7.9), we can modify $\bar{\mathbf{y}}$ as follows. Since we only have an estimate of α_{l-1}, which we denote as $\widehat{\alpha}_{l-1}$, we remove its expected contribution from the system by modifying $\bar{\mathbf{y}}$ as

$$\tilde{\mathbf{y}} \stackrel{\text{def}}{=} \bar{\mathbf{y}} - \bar{\mathbf{\Phi}}\breve{\mathbf{\Psi}}\breve{\alpha}, \tag{7.11}$$

where we use $\breve{\mathbf{\Psi}}$ to denote the first l_{l-1} columns in $\bar{\mathbf{\Psi}}$, which contains a part of Ψ_{l-1}, and $\breve{\alpha}$ to denote $\widehat{\alpha}_{l-1}$. We write the resultant, modified form of the system in (7.9) as

$$\tilde{\mathbf{y}} = \bar{\mathbf{\Phi}}\tilde{\mathbf{\Psi}}\tilde{\alpha} + \tilde{\mathbf{e}}, \tag{7.12}$$

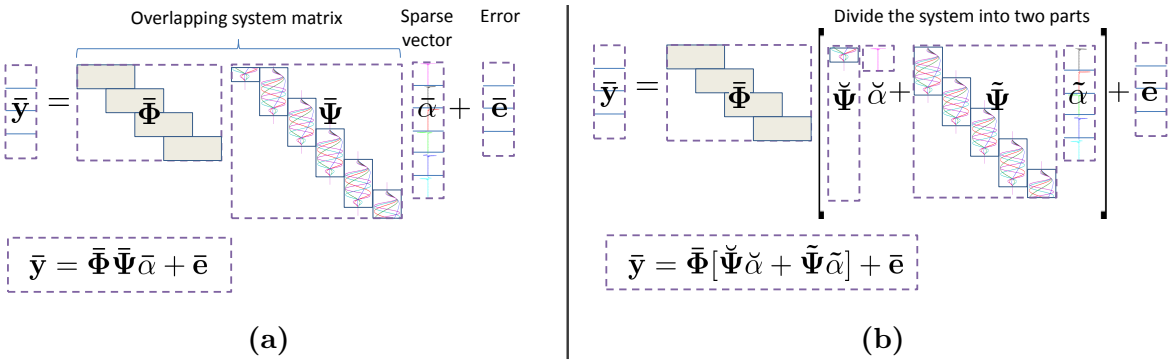


Figure 7.3: Illustration of the system used for the signal reconstruction. **(a)** System over the active interval. **(b)** System divided into two parts so that $\breve{\alpha}$ can be removed.

where $\tilde{\mathbf{e}}$ denotes the combined error in the system and $\tilde{\alpha}$ denotes the unknown vector of coefficients that we estimate by solving a weighted ℓ_1-norm minimization problem.

7.3.2 Recovery problem

To estimate $\tilde{\alpha}$ from the system in (7.12), we solve the following optimization problem:

$$\underset{\alpha}{\text{minimize}} \; \|\mathbf{W}\alpha\|_1 + \frac{1}{2}\|\bar{\mathbf{\Phi}}\tilde{\mathbf{\Psi}}\alpha - \tilde{\mathbf{y}}\|_2^2, \tag{7.13}$$

where $\mathbf{W}$ is a diagonal matrix that consists of positive weights. We select the weights using prior knowledge about the estimate of $\tilde{\alpha}$ from the previous streaming iteration. Let us denote $\widehat{\alpha}$ as our prior estimate of $\tilde{\alpha}$. Since there is a significant overlap between the active intervals at the present and the previous iterations, we expect $\widehat{\alpha}$ to be very close to the solution of (7.13). We compute i^{th} diagonal entry in $\mathbf{W}$ as

$$\mathbf{w}_i \leftarrow \frac{\tau}{\beta|\widehat{\alpha}_i| + 1}, \tag{7.14}$$

where $\tau > 0$ and $\beta >> 1$ are two parameters that can be used to tune the weights according to the problem. Instead of solving (7.13) from scratch, we can speed up the recovery process by providing $\widehat{\alpha}$ as a warm-start (initialization) vector to an appropriate solver [19, 62, 144, 147].

In the dynamic ℓ_1 updating framework, described in Chapter 3, we use the homotopy algorithm in Algorithm 1 to dynamically update the solution of (7.13). Similar

to the homotopy formulation in (3.7), we use the available estimate $\widehat{\alpha}$ as a warm start and solve (7.13) at every streaming iteration using the following homotopy program:

$$\underset{\alpha}{\text{minimize}} \; \|\mathbf{W}\alpha\|_1 + \frac{1}{2}\|\bar{\mathbf{\Phi}}\tilde{\mathbf{\Psi}}\alpha - \tilde{\mathbf{y}}\|_2^2 + (1-\epsilon)\mathbf{u}^T\alpha, \tag{7.15}$$

by changing ϵ from 0 to 1. We provide the following parameters to Algorithm 1: the warm-start vector $\widehat{\alpha}$, the system matrix $\mathbf{\Phi} \leftarrow \bar{\mathbf{\Phi}}\tilde{\mathbf{\Psi}}$, and the measurement vector $\mathbf{y} \leftarrow \tilde{\mathbf{y}}$. We define $\mathbf{u}$ as

$$\mathbf{u} \stackrel{\text{def}}{=} -\mathbf{W}\,\widehat{\mathbf{z}} - (\bar{\mathbf{\Phi}}\tilde{\mathbf{\Psi}})^T(\bar{\mathbf{\Phi}}\tilde{\mathbf{\Psi}}\widehat{\alpha} - \tilde{\mathbf{y}}), \tag{7.16}$$

where $\widehat{\mathbf{z}}$ can be any vector that is defined as $\text{sign}(\widehat{\alpha})$ on the support (nonzero indices) of $\widehat{\alpha}$ and strictly smaller than one elsewhere.

We compute $\widehat{\alpha}$ using the signal estimate from the previous streaming iteration and the available set of measurements. Since we have an estimate of $\bar{\mathbf{x}}$ for the part of Π that is common between the current and the previous iteration, our main task is to predict the signal values that are new to the system. Let us denote the available signal estimate for $\bar{\mathbf{x}}$ as $\widehat{\mathbf{x}}$. We can assign values to the new locations in $\widehat{\mathbf{x}}$ using zero padding, periodic extension, or symmetric extension, and compute $\widehat{\alpha}$ from $\widehat{\mathbf{x}} = \bar{\mathbf{\Psi}}\widehat{\alpha}$. In our experiments, first we update $\widehat{\mathbf{x}}$ by symmetric signal extension onto new locations and identify a candidate support for the new coefficients in $\widehat{\alpha}$; then we calculate magnitudes of the new coefficients by solving a least-squares problem, restricted to the chosen support, using the corresponding measurements in (7.9); and finally we truncate extremely small values of the least-squares solution and update $\widehat{\alpha}$.

7.4 Numerical experiments

We present experiments for the recovery of smooth, time-varying signals from streaming, compressive measurements in (7.1), where we use LOT bases for sparse signal representation. We evaluate the performance of our proposed recovery algorithm for two signals at different compression factors. We compare the performance of

ℓ_1-homotopy algorithm against two state-of-the-art ℓ_1 solvers and demonstrate that ℓ_1-homotopy requires significantly lesser computational operations and time.

7.4.1 Experiment setup

In these experiments, we used the following two discrete-time signals, $x[n]$, from the Wavelab toolbox [29], that have sparse representations in LOT bases: 1) `LinChirp`, which is a critically sampled sinusoidal chirp signal and its frequency increases linearly from zero to one-half of the sampling frequency. 2) `MishMash`, which is a summation of a quadratic and a linear chirp with increasing frequencies and a sinusoidal signal. For both the signals, we generated 2^{15} samples and prepended them with $N = 256$ zeros. Snapshots of `LinChirp` and `MishMash` and their LOT coefficients are presented in Figure 7.4a and Figure 7.5a, respectively. We estimated sparse LOT coefficients of these signals from streaming, compressive measurements using the system model and the recovery procedure outlined in Section 7.3.

We selected the parameters for compressive measurements and the signal representation as follows.

Compressive measurements: To simulate streaming, compressive measurements of a given time-varying signal, $x[n]$, at a compression rate R, we followed the model in (7.1): $y_t = \Phi_t x_t + e_t$. We used non-overlapping x_t of length N to generate a set of $M = N/R$ measurements in y_t. We generated entries in Φ_t independently at random as $\pm 1/\sqrt{M}$ with equal probability. We added Gaussian noise in the measurements by selecting every entry in e_t according to $\mathcal{N}(0, \sigma^2)$ distribution. We selected the variance σ^2 such that the expected SNR with respect to the measurements $\Phi_t x_t$ becomes 35 dB.

Signal representation: To represent $x[n]$ using LOT bases, according to (7.4), we selected the overlapping intervals, I_p, of the same length $N + 2\eta_p = 2N$, where we fixed $\eta_p = N/2$, $a_p = pN + 1/2$, and $l_p = N$ for all $p \in \mathbb{Z}$. We divided $x[n]$ into overlapping

intervals, I_p, and computed LOT coefficients, α_p, corresponding to every I_p.

At every streaming iteration, we built the system in (7.12) for $P = 5$ consecutive x_t in $\bar{\mathbf{x}}$. We updated the system in (7.7), from the previous iteration, by shifting the active interval, removing old measurements, and adding new measurements. We computed $\tilde{\mathbf{y}}$ in (7.11), committed a portion of $\widehat{\alpha}$ to the output. The combined system in (7.12), corresponding to the unknown vector $\bar{\mathbf{x}}$ of length PN, thus, consists of a measurement vector $\tilde{\mathbf{y}}$ of length PM, a block diagonal $PM \times PN$ measurement matrix $\bar{\mathbf{\Phi}}$, a $PN \times PN$ LOT representation matrix $\tilde{\mathbf{\Psi}}$ in which adjacent pairs of columns overlap in N rows, the unknown LOT coefficient vector $\tilde{\alpha}$ of length PN, and a noise vector $\tilde{\mathbf{e}}$. An example of such a system in depicted in Figure 7.3. We predicted the new coefficients in $\widehat{\alpha}$, updated the weights $\mathbf{W}$, and solved (7.13) using $\widehat{\alpha}$ as a warm-start. We updated the weights according to (7.14) using $\beta = M\frac{\|\widehat{\alpha}\|_2^2}{\|\widehat{\alpha}\|_1^2}$ and $\tau = \max\{10^{-2}\|\mathbf{\Phi}^T\mathbf{y}\|_\infty, \sigma\sqrt{\log(PN)}\}$, where $\mathbf{\Phi}$ and $\mathbf{y}$ denote the system matrix and the measurement vector in (7.13), respectively, and σ denotes the standard deviation of the measurement noise. For the first streaming iteration, we initialized α as zero and solved (7.13) as an iterative reweighted ℓ_1 problem, starting with $\mathbf{W} = \tau$, using five reweighting iterations [12, 42].

We solved (7.13) using our proposed ℓ_1-homotopy algorithm and two state-of-the-art ℓ_1 solvers: YALL1 [148] and SpaRSA [147], with identical initialization (warm-start) and weight selection schemes. Further description of these algorithms is as follows.

i. ℓ_1-*homotopy*: We solved (7.15) following the procedure outlined in Algorithm 1. The main computational cost at every step of ℓ_1-homotopy involves one matrix-vector multiplication for identifying a change in the support and a rank-one update for computing the update direction. We used the matrix inversion lemma-based scheme to perform the rank-one updates. MATLAB code for the ℓ_1-homotopy package is available at http://users.ece.gatech.edu/∼sasif/homotopy.

ii. *YALL1*: YALL1 is a first-order algorithm that uses an alternating direction minimization method for solving various ℓ_1 problems, see [148] for further details. We solved (7.13) using weighted-ℓ_1/ℓ_2 solver in YALL1 package by selecting the initialization vector and weights according to the procedure described in Section 7.3.2. At every streaming iteration, we used the previous YALL1 solution to predict the initialization vector and the weights according to (7.14). We fixed the tolerance parameter to 10^{-4} in all the experiments. The main computational cost of every step in the YALL1 solver comes from applications of the system matrix in (7.13) and its adjoint. MATLAB package for YALL1 is available at http://yall1.blogs.rice.edu/.

iii. *SpaRSA*: SpaRSA is also a first-order method that uses a fast variant of iterative shrinkage and thresholding for solving various ℓ_1-regularized problems, see [147] for further details. Similar to YALL1, we solved (7.13) using SpaRSA at every streaming iteration by selecting the initialization vector and the weights from the solution of the previous iteration. We used the SpaRSA code with the default adaptive continuation procedure in the Safeguard mode using the duality gap-based termination criterion for which we fixed the tolerance parameter to 10^{-4} and modified the code to accommodate weights in the evaluation. The main computational cost for every step in the SpaRSA solver also involves applications of the system matrix in (7.13) and its adjoint. MATLAB package for SpaRSA is available at http://lx.it.pt/∼mtf/SpaRSA/.

To summarize, ℓ_1-homotopy solves the homotopy formulation of (7.13), given in (7.15), while YALL1 and SpaRSA solve (7.13) using a warm-start vector for the initialization.

We used MATLAB implementations of all the algorithms and performed all the experiments on a standard laptop computer. We used a single computational thread for all the experiments, which involved the recovery of a sparse signal from a given set

of streaming measurements using all the candidate algorithms. In every experiment, we recorded three quantities for each algorithm: 1) the quality of the reconstructed signal in terms of the signal-to-error ratio (SER) in dB, defined as

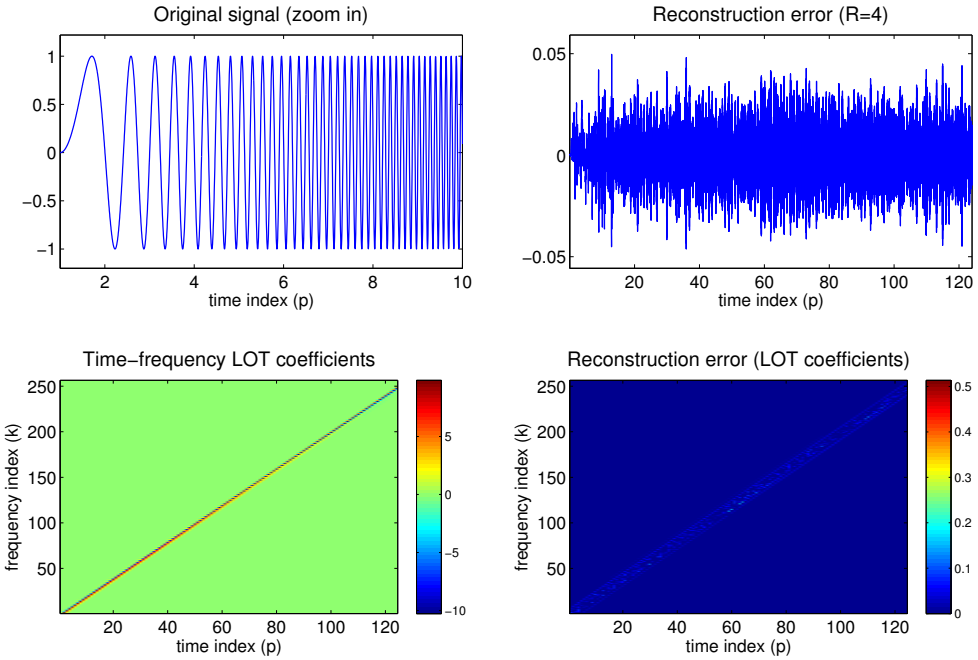
$$\text{SER} = -10 \log_{10} \frac{\|x - \widehat{x}\|_2^2}{\|x\|_2^2},$$

where x and $\widehat{x}$ denote the original and the reconstructed streaming signal, respectively, 2) the number of matrix-vector products with the system matrix in (7.13) and its adjoint, and 3) the execution time in MATLAB.

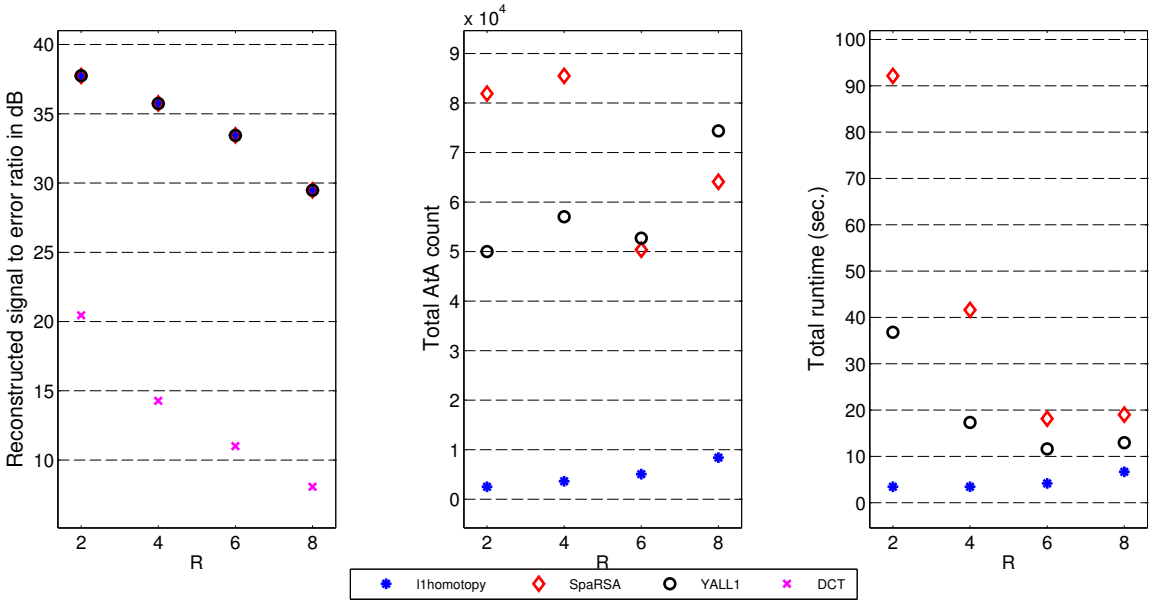
7.4.2 Results

We compared performances of the ℓ_1-homotopy, YALL1, and SpaRSA for the recovery of `LinChirp` and `MishMash` signals from streaming, compressive measurements. We performed 5 independent trials for the recovery of the streaming signal from random, streaming measurements at different values of the compression factor R. The results, averaged over all the trials are presented in Figures 7.4–7.5.

Figure 7.4 presents results for experiments with `LinChirp` signal. Figure 7.4a presents a snapshot of the `LinChirp` signal, its LOT coefficients, and the reconstruction error at $R = 4$. Three plots in Figure 7.4b present results for the three solvers: ℓ_1-homotopy ($*$), SpaRSA ($\diamond$), and YALL1 ($\circ$). The left plot in Figure 7.4b compares the SER for the three solvers. Since all of them solve the same convex program, SERs for the reconstructed signals are almost identical. To gauge the advantage of the LOT-based reconstruction over a block transform-based reconstruction, we repeated the same experiments by replacing the LOT bases with the DCT bases for the signal representation (results shown as $\times$). We can see a significant degradation (more than 20 dB loss in the SER) in the results for the DCT-based representation as compared to the results for the LOT-based representation. The middle plot in Figure 7.4b compares the computational cost of all the algorithms in terms of the total number of matrix-vector multiplications used in the signal reconstruction. We

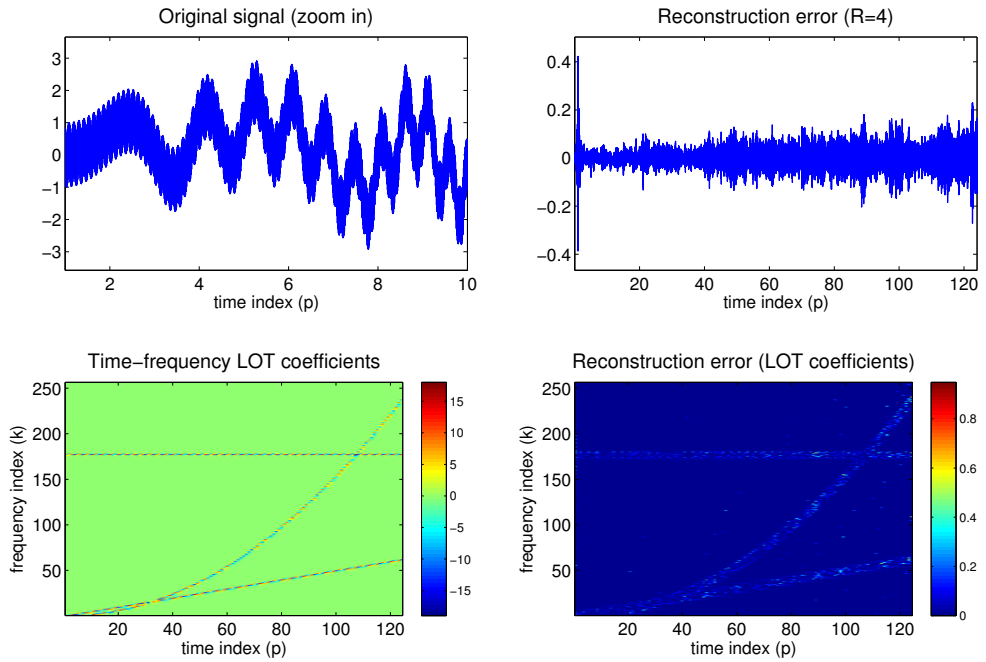


(a) Snapshot of `LinChirp` signal, LOT coefficients, and errors in the reconstruction. **Top left:** Signal $x[n]$ (zoomed in over first 2560 samples). **Bottom left:** LOT coefficients α_p. **Top right:** Error in the reconstructed signal at $R = 4$. **Bottom right:** Error in the reconstructed LOT coefficients

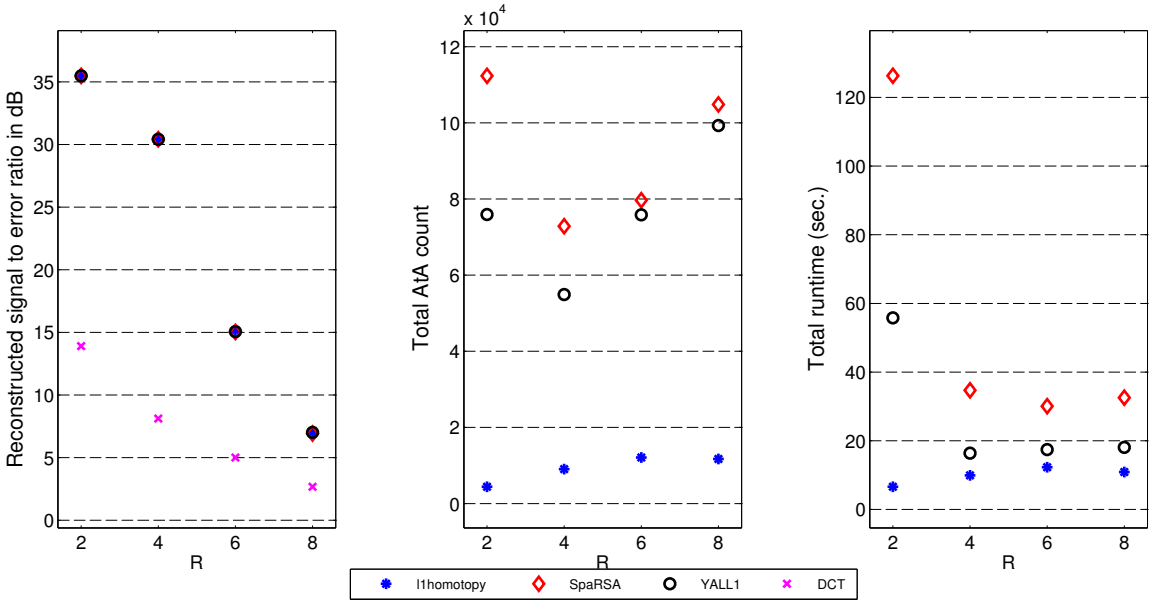


(b) Results for the recovery of `LinChirp` signal from random, compressive measurements in the presence of noise at 35dB SNR. **Left:** SER at different R. **Middle:** Approximate count of matrix-vector multiplications. **Right:** Matlab execution time in seconds.

Figure 7.4: Experiments on the `LinChirp` signal reconstruction from streaming, compressed measurements using LOT representation.



(a) Snapshot of `MishMash` signal, LOT coefficients, and errors in the reconstruction. **Top left:** Signal $x[n]$ (zoomed in over first 2560 samples). **Bottom left:** LOT coefficients α_p. **Top right:** Error in the reconstructed signal at $R = 4$. **Bottom right:** Error in the reconstructed LOT coefficients



(b) Results for the recovery of `MishMash` signal from random, compressive measurements in the presence of noise at 35dB SNR. **Left:** SER at different R. **Middle:** Approximate count of matrix-vector multiplications. **Right:** Matlab execution time in seconds.

Figure 7.5: Experiments on the `MishMash` signal reconstruction from streaming, compressed measurements using LOT representation.

counted an application of the system matrix in (7.13) and its adjoint in as one count in AtA. For the homotopy algorithms, we approximated the cost of one step as one application of AtA. We can see that, out of the three solvers, ℓ_1-homotopy required the least number of AtA count in all the cases. The right plot in Figure 7.4b compares the MATLAB execution time for each solver. We can see that, compared to YALL1 and SpaRSA, ℓ_1-homotopy consumed distinctly lesser time for the reconstruction.

Figures 7.5 presents similar results for experiments with `MishMash` signal. Figure 7.5a presents a snapshot of the `MishMash` signal, its LOT coefficients, and reconstruction error at $R = 4$. Three plots in Figure 7.5b compare performance of the three solvers. In these plots we see similar results that the reconstruction error for (7.13) using all the solvers is almost identical, but ℓ_1-homotopy performs significantly better in terms of the computational cost and execution time.

A brief summary of the results for our experiments is as follows. We observed that the signals reconstructed using the LOT-based representation had significantly better quality compared to those reconstructed using the DCT-based signal representation. The computational cost and execution time for ℓ_1-homotopy is significantly smaller than that for SpaRSA and YALL1.

CHAPTER VIII

SPARSE RECOVERY FROM DYNAMICAL SYSTEMS

In this chapter we discuss the problem of estimating a sparse, time-varying signal from streaming measurements when the signal varies according to a linear dynamic model. We describe the signal using a discrete-time linear dynamical system in which measurements of the signal are recorded at regular time intervals. Given the linear dynamical system and the sparse representation basis of the signal, we sequentially estimate the signal over a sliding interval by solving an optimization problem that balances fidelity to the measurements, the linear dynamic model (both measured by the standard ℓ_2 norm), and the sparsity of signal in the representation basis (measured using the ℓ_1 norm). Since we iteratively estimate overlapping portions of the signal while adding and removing measurements, instead of solving a new optimization program at every iteration, we use an available signal estimate as a starting point in a homotopy formulation and update the solution in a small number of computationally inexpensive homotopy steps. We demonstrate the performance of our proposed recovery algorithm for various time-varying signals.

8.1 Introduction

We consider the following linear dynamical system for a discrete-time signal $x[n]$:

$$y_t = \Phi_t x_t + e_t, \tag{8.1a}$$

$$x_{t+1} = F_t x_t + f_t, \tag{8.1b}$$

where x_t is a vector that denotes the state of $x[n]$ at time index t, y_t is a vector that contains measurements of x_t, Φ_t is a measurement matrix, and e_t is the noise in the measurements; F_t is a prediction matrix that couples consecutive, non-overlapping

x_t and x_{t+1} and f_t is the error in the prediction, which we assume has a bounded ℓ_2 norm.

The Kalman filter is a classical signal estimation method that solves the system in (8.1) in the least-squares framework [78, 126]. Given a sequence of measurements $y_1, \ldots, y_P$, and starting from an initial estimate $\widehat{x}_0$ of the signal at time $t = 0$, we can estimate $x_1, \ldots, x_P$ by solving the following optimization program

$$\underset{x_1,\ldots,x_P}{\text{minimize}} \sum_{p=1}^{P} \|\Phi_p x_p - y_p\|_2^2 + \lambda_p \|F_{p-1} x_{p-1} - x_p\|_2^2. \tag{8.2}$$

The λ_p above are regularization parameters that can be tuned based on our relative confidence in the measurement and prediction errors—if the error vectors e_t and w_t are independent and identically distributed Gaussian noise vectors, then optimal choices of these parameters are based on the variances of the entries in the noise vectors.

The Kalman filter owes its status as one of the pillars of statistical signal processing to the fact that the solution to (8.2) can be computed recursively. Say we have solved (8.2) after P time steps, and denote the estimate of x_P at this point as $\widehat{x}_{P|P}$. Then given a new set of measurements y_{P+1}, we can efficiently update the estimate $\widehat{x}_{P+1|P+1}$ of the new current state given only the previous estimate $\widehat{x}_{P|P}$ and the signal covariance matrix [70, 79]. Computational cost for the solution update is dominated by a low-rank update of the covariance matrix. Estimates for previous values of x_t for $t < P + 1$ can be updated by working backwards ("smoothing"), with each step requiring a similar low-rank computation. However, the standard Kalman filter is oblivious to a sparse structure in the signal.

To leverage the sparse structure of the signal in its estimation, we add an ℓ_1-norm regularization term to the Kalman filter optimization program in (8.2) as follows. Suppose x_t can be represented as $x_t = \Psi_t \alpha_t$, where α_t is a sparse vector and Ψ_t is a sparse representation basis. We write the modified optimization problem as

$$\underset{\alpha_1,\ldots,\alpha_P}{\text{minimize}} \sum_{p=1}^{P} \|W_p \alpha_p\|_1 + \frac{1}{2}\|\Phi_p \Psi_p \alpha_p - y_p\|_2^2 + \frac{\lambda_p}{2}\|F_{p-1}\Psi_{p-1}\alpha_{p-1} - \Psi_p \alpha_p\|_2^2, \tag{8.3}$$

where the ℓ_1 term promotes sparsity in the estimated coefficients; W_p is a diagonal matrix of positive weights that can be adapted to promote a certain sparse structure in the solution [42, 152]; and the ℓ_2 terms ensure fidelity of the solution to the system in (8.1). Our recovery algorithm maintains the optimal solution over a sliding active interval of time instances, and yields the jointly optimal solution given the measurements inside the active interval and the estimate from the *last frame out*, which in this case is the initial estimate $\widehat{x}_0$.

Recently, several methods have been proposed to incorporate signal dynamics into the sparse signal estimation framework [2, 4, 5, 43, 44, 139, 151]. The method in [139] identifies the support of the signal by solving an ℓ_1 problem and modifies the Kalman filter to estimate the signal on the identified support; [43] embeds additional steps within the original Kalman filter algorithm for promoting a sparse solution; [151] uses a belief propagation algorithm to identify the support and update the signal estimate; [44] compares different types of sparse dynamics by solving an ℓ_1 problem for one signal block; [4] assumes that the prediction error is sparse and jointly estimates multiple signal blocks using a homotopy algorithm; and [2] solves a group-sparse ℓ_1 problem for a highly restrictive signal model in which the locations of nonzero components of the signal do not change.

In this chapter, we consider a general dynamic model in (8.1) and solve the problem in (8.3) over a sliding interval. Our emphasis is on an efficient updating scheme for moving from one solution to the next as new measurements are added and old ones are removed. Before solving the optimization problem, an estimate of the signal over the active interval is either available from the previous iteration or it can be predicted. We use the available signal estimate to aide the recovery process in two ways: We update the W_t using available estimates of the α_t (in the same spirit as iterative reweighting [42]), and we use the available estimates of the α_t as a starting point to expedite the solution of the ℓ_1 problem.

We present the sparse recovery problem in Section 8.2 and experimental results to demonstrate the performance of our algorithms, in terms of the quality of reconstructed signals and the computational cost of the recovery algorithm, in Section 8.3.

8.2 Sparse recovery with dynamic model

The recovery procedure for (8.3) is similar to the one described in Section 7.3, with the exception that here we have appended the linear dynamic model in (8.1b) to the system and updated the optimization problem accordingly. At every streaming iteration, we estimate sparse coefficients of the signal over an active, sliding interval. We solve the weighted ℓ_1-norm minimization problem in (8.3) to estimate the sparse coefficients for the signal over the active interval. After every iteration of the streaming recovery process, we shift the active interval by removing a few oldest measurements and adding a few new ones in the system. Estimates of the sparse coefficients and the signal portion that leave the active interval are committed to the output.

8.2.1 System model

Consider the linear dynamic system in (8.1), which describes linear measurements of the x_t and the dependencies between the consecutive x_t. At every streaming iteration, we denote $x[n]$ over the active interval Π as $\bar{\mathbf{x}}$ and assume that $\bar{\mathbf{x}}$ consists of a small number of x_t. We describe the system of measurements for $\bar{\mathbf{x}}$ in the following compact form:

$$\bar{\mathbf{y}} = \bar{\mathbf{\Phi}}\bar{\mathbf{x}} + \bar{\mathbf{e}}, \tag{8.4}$$

where $\bar{\mathbf{y}}$ denotes a vector that contains y_t for the x_t that belong to $\bar{\mathbf{x}}$, $\bar{\mathbf{\Phi}}$ denotes a matrix that contains the corresponding Φ_t, and $\bar{\mathbf{e}}$ denotes the noise vector. At every iteration of the streaming recovery algorithm, we shift Π by removing the oldest y_t in the system and adding a new one and update the system in (8.4) accordingly.

We describe a combined system of prediction equations for the x_t that belong to $\bar{\mathbf{x}}$ as follows. Suppose $\bar{\mathbf{x}}$ contains $x_l, \ldots, x_r$, and an estimate of x_{l-1}, which was removed

from $\bar{\mathbf{x}}$ and committed to the output, is given as $\widehat{x}_{l-1}$. We rearrange the equations in (8.1b) for $t = l$ as $-F_{l-1}\widehat{x}_{l-1} = -x_l + f_{l-1}$ and for the rest of t as $0 = F_t x_t - x_{t+1} + f_t$. We stack these equations on top of one another to write the following compact form:

$$\bar{\mathbf{q}} = \bar{\mathbf{F}}\bar{\mathbf{x}} + \bar{\mathbf{f}}. \tag{8.5}$$

$\bar{\mathbf{F}}$ denotes a banded matrix that consists of negative identity matrices in the main diagonal and $F_l, \ldots, F_r$ below the diagonal; $\bar{\mathbf{f}}$ denotes the combined prediction error; $\bar{\mathbf{q}}$ denotes a vector that contains $-F_{l-1}\widehat{x}_{l-1}$ followed by zeros.

Combining the systems in (8.4) and (8.5) with the sparse representation ($\bar{\mathbf{x}} = \bar{\mathbf{\Psi}}\bar{\alpha}$), we write the modified system over the active interval Π as

$$\begin{bmatrix} \bar{\mathbf{y}} \\ \bar{\mathbf{q}} \end{bmatrix} = \begin{bmatrix} \bar{\mathbf{\Phi}} \\ \bar{\mathbf{F}} \end{bmatrix} \bar{\mathbf{\Psi}}\bar{\alpha} + \begin{bmatrix} \bar{\mathbf{e}} \\ \bar{\mathbf{f}} \end{bmatrix}. \tag{8.6}$$

As we discussed in Section 7.3.1 that using (7.10) we can remove those components of $\bar{\alpha}$ from the system that are committed to the output. Following the same procedure, if we want to remove a vector α_{l-1} that belongs to $\bar{\alpha}$ from the system, we decompose $\bar{\mathbf{\Psi}}\bar{\alpha}$ into two components; $\tilde{\mathbf{\Psi}}\tilde{\alpha} + \breve{\mathbf{\Psi}}\breve{\alpha}$, where $\breve{\alpha}$ denotes the vector that we want to remove from the system, and modify the system in (8.6) as

$$\begin{bmatrix} \tilde{\mathbf{y}} \\ \tilde{\mathbf{q}} \end{bmatrix} \stackrel{\text{def}}{=} \begin{bmatrix} \bar{\mathbf{y}} \\ \bar{\mathbf{q}} \end{bmatrix} - \begin{bmatrix} \bar{\mathbf{\Phi}} \\ \bar{\mathbf{F}} \end{bmatrix} \breve{\mathbf{\Psi}}\breve{\alpha}, \tag{8.7}$$

where $\breve{\alpha}$ denotes $\widehat{\alpha}_{l-1}$ that is the estimate of α_{l-1} and $\breve{\mathbf{\Psi}}$ denotes the columns in $\bar{\mathbf{\Psi}}$ that correspond to the locations of α_{l-1} in $\bar{\alpha}$. We represent the modified form of the system as

$$\begin{bmatrix} \tilde{\mathbf{y}} \\ \tilde{\mathbf{q}} \end{bmatrix} = \begin{bmatrix} \bar{\mathbf{\Phi}} \\ \bar{\mathbf{F}} \end{bmatrix} \tilde{\mathbf{\Psi}}\tilde{\alpha} + \begin{bmatrix} \tilde{\mathbf{e}} \\ \tilde{\mathbf{f}} \end{bmatrix}. \tag{8.8}$$

8.2.2 Recovery problem

To estimate $\tilde{\alpha}$ from the system in (8.8), we solve the following optimization problem:

$$\underset{\alpha}{\text{minimize}} \, \|\mathbf{W}\alpha\|_1 + \frac{1}{2}\|\bar{\mathbf{\Phi}}\tilde{\mathbf{\Psi}}\alpha - \tilde{\mathbf{y}}\|_2^2 + \frac{\lambda}{2}\|\bar{\mathbf{F}}\tilde{\mathbf{\Psi}}\alpha - \tilde{\mathbf{q}}\|_2^2, \tag{8.9}$$

where $\mathbf{W}$ is a diagonal matrix that consists of positive weights and $\lambda > 0$ is a regularization parameter that controls the effect of the dynamic model on the solution. We select the weights using a prior estimate of $\tilde{\alpha}$, which we denote as $\widehat{\alpha}$. Estimate of a significant portion of $\widehat{\alpha}$ is known from the previous streaming iteration, and only a small portion is new to the system. We predict the incoming portion of the signal, say x_r, using the prediction matrix in (8.1b) and the signal estimate from the previous iteration as $\widehat{x}_{r|r-1} \stackrel{\text{def}}{=} F_{r-1}\widehat{x}_{r-1}$. We update the coefficients in $\widehat{\alpha}$ accordingly and set very small coefficients in $\widehat{\alpha}$ to zero. We computed i^{th} diagonal entry in $\mathbf{W}$ as

$$\mathbf{w}_i \leftarrow \frac{\tau}{\beta|\widehat{\alpha}_i| + 1}, \tag{8.10}$$

where $\tau > 0$ and $\beta >> 1$ are two parameters that can be used to tune the weights according to the problem.

Instead of solving (8.9) from scratch, we use the available estimate $\widehat{\alpha}$ as a warm-start. In the dynamic ℓ_1 updating framework, described in Chapter 3, we use the following homotopy formulation to solve (8.9):

$$\underset{\alpha}{\text{minimize}} \, \|\mathbf{W}\alpha\|_1 + \frac{1}{2}\|\bar{\mathbf{\Phi}}\tilde{\mathbf{\Psi}}\alpha - \tilde{\mathbf{y}}\|_2^2 + \frac{\lambda}{2}\|\bar{\mathbf{F}}\tilde{\mathbf{\Psi}}\alpha - \tilde{\mathbf{q}}\|_2^2 + (1-\epsilon)\mathbf{u}^T\alpha, \tag{8.11}$$

by changing ϵ from 0 to 1 in Algorithm 1, using the warm-start vector $\widehat{\alpha}$, the system matrix $\mathbf{\Phi} \leftarrow \begin{bmatrix} \bar{\mathbf{\Phi}}\tilde{\mathbf{\Psi}} \\ \sqrt{\lambda}\bar{\mathbf{F}}\tilde{\mathbf{\Psi}} \end{bmatrix}$, and the measurement vector $\mathbf{y} \leftarrow \begin{bmatrix} \tilde{\mathbf{y}} \\ \sqrt{\lambda}\tilde{\mathbf{q}} \end{bmatrix}$. We define $\mathbf{u}$ as

$$\mathbf{u} \stackrel{\text{def}}{=} -\mathbf{W}\,\widehat{\mathbf{z}} - (\bar{\mathbf{\Phi}}\tilde{\mathbf{\Psi}})^T(\bar{\mathbf{\Phi}}\tilde{\mathbf{\Psi}}\widehat{\alpha} - \tilde{\mathbf{y}}) - \lambda\,(\bar{\mathbf{F}}\tilde{\mathbf{\Psi}})^T(\bar{\mathbf{F}}\tilde{\mathbf{\Psi}}\widehat{\alpha} - \tilde{\mathbf{q}}), \tag{8.12}$$

where $\widehat{\mathbf{z}}$ is defined as before.

8.3 Numerical experiments

We present experiments for the recovery of time-varying signals using the linear dynamic model in (8.1) and using wavelet transforms for the sparse signal representation. We evaluate the performance of our proposed recovery algorithm for two signals at different compression factors. We compare the performance of the ℓ_1-homotopy algorithm against a state-of-the-art ℓ_1 solver and demonstrate that ℓ_1-homotopy requires significantly lesser computational operations and time.

8.3.1 Experiment setup

In these experiments, we simulated time-varying signal $x[n]$ according to the linear dynamic model defined in (8.1b): $x_{t+1} = F_t x_t + f_t$. We generated a seed signal of length $N = 256$, which we will denote as x_0. Starting with x_0, we generated a sequence of signal instances x_t for $t = 1,\ 2, \ldots$ as follows. For each t we generated x_{t+1} by applying a non-integer, left-circular shift $\epsilon_t \sim \text{uniform}(0.5,\ 1.5)$ to x_t (i.e., $x_{t+1}[n] = x_t[(n + \epsilon_t)_{\text{mod } N}]$, where ϵ_t is drawn uniformly, at random from interval $[0.5,\ 1.5]$). We computed values of x_t over non-integer locations using linear interpolation. For defining the dynamic model, we assumed that the individual shifts (ϵ_t) are unknown and only their average value is known, which is one in our experiments. Therefore, we defined F_t, for all t, as a matrix that applies left-circular shift of one, whereas f_t accounts for the prediction error in the model because of the unaccounted component of the shift ϵ_t. We used the following two signals from the Wavelab toolbox [29] as x_0: 1) `HeaviSine`, which is a summation of a sinusoidal and a rectangular signal and 2) `Piece-Regular`, which is a piecewise smooth signal. `HeaviSine` and `Piece-Regular` signals along with examples of their shifted copies are presented in Figure 8.1a and Figure 8.2a, respectively. We concatenated the x_t for $t = 1,\ 2, \ldots, 128$ to build the time-varying signal $x[n]$ of length 2^{15}. We estimated sparse wavelet coefficients of $x[n]$ from streaming, compressive measurements using the system model and the recovery

procedure outlined in Section 8.2.

We selected the compressive measurements and the signal representation as follows.

Compressive measurements: We simulated streaming, compressive measurements of $x[n]$ according to (8.1a), using the following procedure: For a desired compression rate R, we generated y_t with $M = N/R$ measurements of non-overlapping x_t, generated entries in Φ_t as $\pm 1/\sqrt{M}$ with equal probability, and added Gaussian noise in the measurements such that the expected SNR becomes 35 dB.

Signal representation: We represented $x[n]$ using block-based wavelet bases. We used Daubechies-8 orthogonal wavelets [53] for the signal representation with five levels of decomposition. We divided $x[n]$ into non-overlapping components, x_t, of length N and computed wavelet coefficients, α_t, using circular convolution in the wavelet analysis filter bank. We used five levels of wavelet decomposition.

At every streaming iteration, we built the system in (8.8) for $P = 3$ consecutive x_t in $\bar{\mathbf{x}}$. We updated the system in (8.6), from the previous iteration, by shifting the active interval, removing old measurements, and adding new measurements. We computed $\tilde{\mathbf{y}}$ and $\tilde{\mathbf{q}}$ in (8.7) and committed a portion of $\widehat{\alpha}$ to the output. The combined system in (8.8), corresponding to the unknown vector $\bar{\mathbf{x}}$ of length PN, thus, consists of measurement vectors $\tilde{\mathbf{y}}, \tilde{\mathbf{q}}$ of length PM and PN, respectively, a block diagonal $PM \times PN$ measurement matrix $\bar{\bar{\mathbf{\Phi}}}$, a banded $PN \times PN$ prediction matrix $\bar{\mathbf{F}}$, a block-diagonal $PN \times PN$ representation matrix $\tilde{\bar{\mathbf{\Psi}}}$, the unknown wavelet coefficient vector $\tilde{\alpha}$ of length PN, and error vectors $\tilde{\mathbf{e}}, \tilde{\mathbf{f}}$. We predicted values of the new coefficients in $\widehat{\alpha}$, updated the weights $\mathbf{W}$, and solved (8.9) using $\widehat{\alpha}$ as a warm-start. We selected $\lambda = 1/2$ and updated the weights according to (8.10) using $\beta = M\frac{\|\widehat{\alpha}\|_2^2}{\|\widehat{\alpha}\|_1^2}$ and $\tau = \max\{10^{-2}\|\mathbf{\Phi}^T\mathbf{y}\|_\infty, \sigma\sqrt{\log(PN)}\}$, where $\mathbf{\Phi}$ and $\mathbf{y}$ denote the system matrix and the measurements in (8.9), respectively, and σ denotes the standard deviation of the measurement noise. We truncated the values in $\widehat{\alpha}$ that are smaller than $\tau/\sqrt{\log(PN)}$

to zero. For the first streaming iteration, we initialized $\widehat{x}_{l-1}$ as x_0 and α as zero. We solved (8.9) as an iterative reweighted ℓ_1 problem, starting with $\mathbf{W} = \tau$, using five reweighting iterations [12, 42].

We solved (8.9) using our proposed ℓ_1-homotopy algorithm in Algorithm 1 (which in fact solves (8.11)) and SpaRSA, with identical initialization (warm-start) and weight selection scheme. Since YALL1 only works with under-determined systems, we did not use that in these experiments.

We used MATLAB implementations of all the algorithms and performed all the experiments on a standard laptop computer. In every experiment, we recorded three quantities for each algorithm: 1) the quality of reconstructed signal in terms of the signal-to-error ratio (SER) in dB, defined as

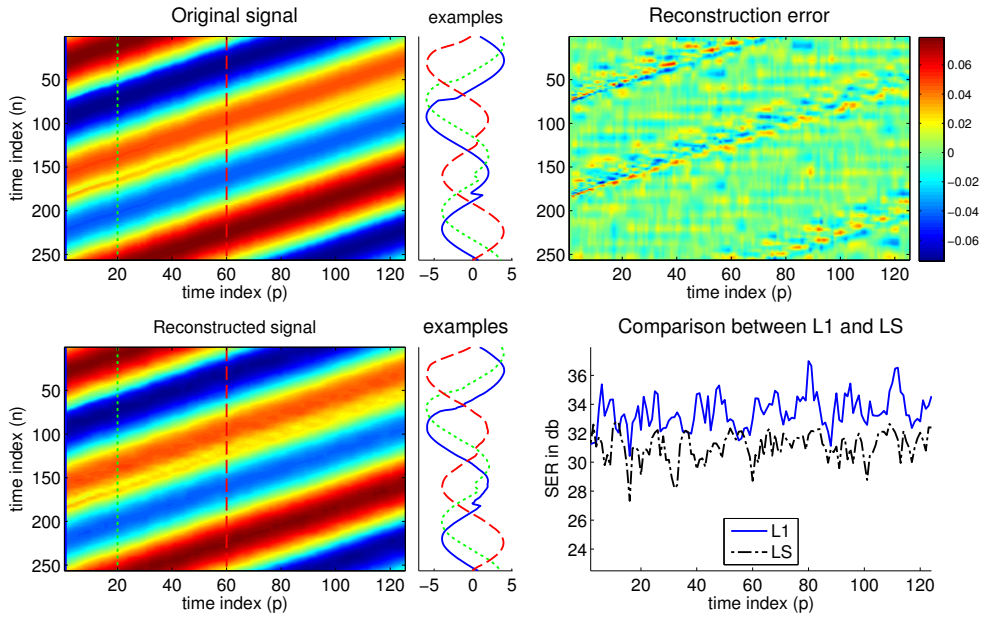
$$\text{SER} = -10 \log_{10} \frac{\|x - \widehat{x}\|_2^2}{\|x\|_2^2},$$

where x and $\widehat{x}$ denote the original and the reconstructed signal, respectively, 2) the number of matrix-vector products with the system matrix in (8.9) and its adjoint, and 3) the execution time in MATLAB.

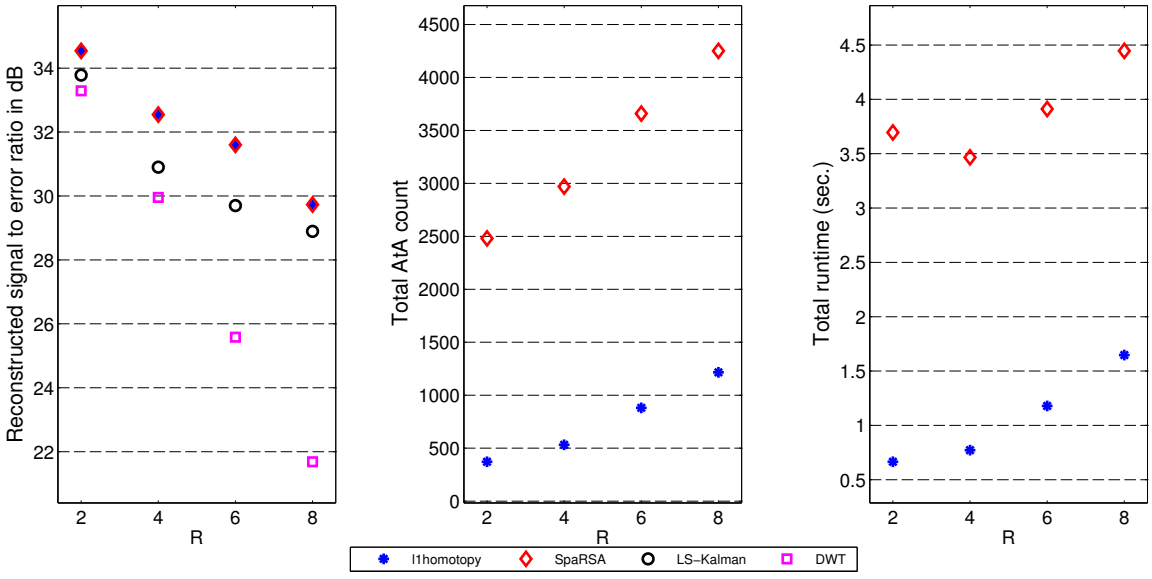
8.3.2 Results

We compared the performance of ℓ_1-homotopy and SpaRSA for the recovery of `HeaviSine` and `Piece-Regular` signals from streaming, compressive measurements. We performed 5 independent trials at different values of the compression factor R. In each experiment, we estimated the time-varying signal using all the algorithms, according to the procedures described above, and recorded the corresponding signal-to-error ratios, the number of matrix-vector products, and MATLAB runtime. The results, averaged over all the trials, are presented in Figures 8.1–8.2.

Figure 8.1 presents results for experiments with `HeaviSine` signal. Figure 8.1a, top-left image presents the `HeaviSine` signal, where p^{th} column represents x_p. Next to the image, we have plotted three examples for $x_1,\ x_{20},\ x_{60}$ as three colored lines.

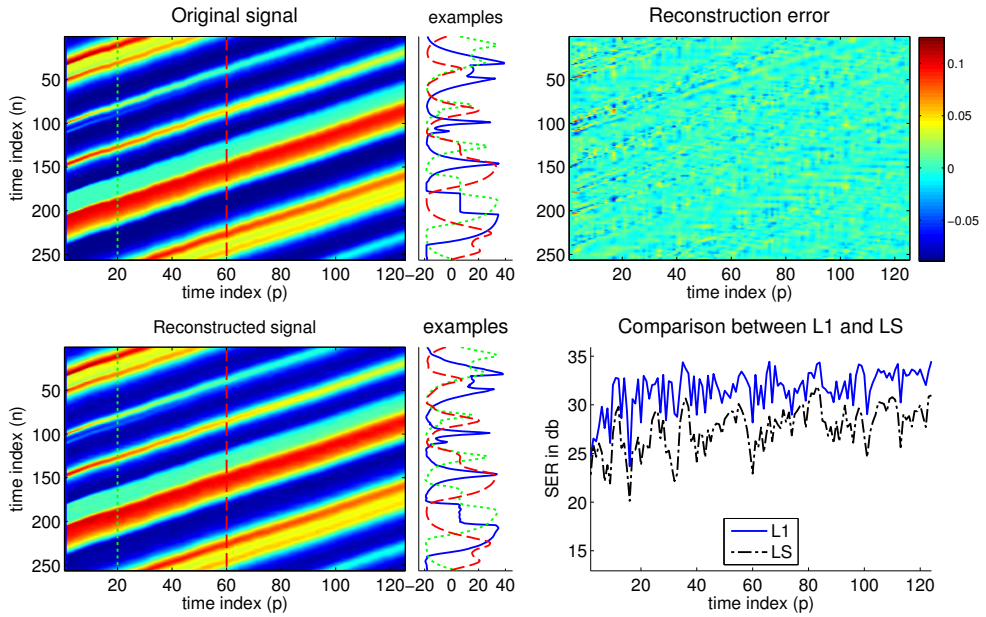


(a) Snapshot of the original and the reconstructed signal, error in the reconstruction, and the comparison of ℓ_1- and ℓ_2-regularized reconstructions. **Top left:** `HeaviSine` signal $x[n]$ drawn as an image; p^{th} column represents x_p; x_1, x_{20} and x_{60} are plotted on the right. **Bottom left:** Reconstructed signal at $R = 4$. **Top right:** Error in the reconstructed signal. **Bottom right:** Comparison between SERs for the solution of the ℓ_1-regularized problem in (8.9) (solid-blue line, labeled L1) and the solution of the ℓ_2-regularized (Kalman filter smoothing) problem in (8.13) (broken-black line, labeled LS).

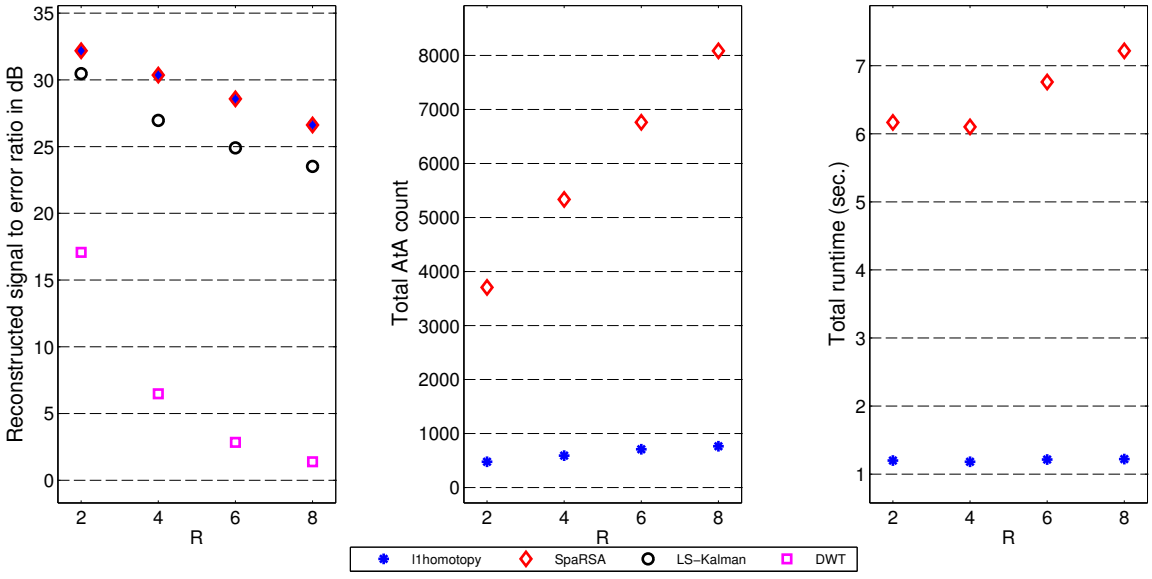


(b) Results for the recovery of `HeaviSine` signal from random, compressive measurements in the presence of noise at 35dB SNR. **Left:** SER at different compression rate R. **Middle:** Approximate count of matrix-vector multiplications. **Right:** Matlab execution time in seconds.

Figure 8.1: Experiments on the time-varying `HeaviSine` signal reconstruction from streaming, compressed measurements in a dynamical system.



(a) Snapshot of the original and the reconstructed signal, error in the reconstruction, and the comparison of ℓ_1- and ℓ_2-regularized reconstructions. **Top left:** `Piece-Regular` signal $x[n]$ drawn as an image; p^{th} column represents x_p; x_1, x_{20} and x_{60} are plotted on the right. **Bottom left:** Reconstructed signal at $R = 4$. **Top right:** Error in the reconstructed signal. **Bottom right:** Comparison between SERs for the solution of the ℓ_1-regularized problem in (8.9) (solid-blue line, labeled L1) and the solution of the ℓ_2-regularized (Kalman filter smoothing) problem in (8.13) (broken-black line, labeled LS).



(b) Results for the recovery of `Piece-Regular` signal from random, compressive measurements in the presence of noise at 35dB SNR. **Left:** Signal to error ratio at different compression rate R. **Middle:** Approximate count of matrix-vector multiplications. **Right:** Matlab execution time in seconds.

Figure 8.2: Experiments on the time-varying `Piece-Regular` signal reconstruction from streaming, compressed measurements in a dynamical system.

Bottom-left image is the reconstructed signal at $R = 4$ along with the examples of the reconstructed x_p on its right. Top-right image represents errors in the reconstruction. Bottom-right plot presents a comparison between the SER for the solution of the ℓ_1-regularized problem in (8.9) and the solution of the following ℓ_2-regularized (Kalman filtering and smoothing) problem using the systems in (8.4) and (8.5):

$$\underset{\mathbf{x}}{\text{minimize}}\ (x_p - \widehat{x}_{p|p-1})^T P_{p|p-1}^{-1}(x_p - \widehat{x}_{p|p-1}) + \lambda\|\tilde{\mathbf{F}}\mathbf{x}\|_2^2 + \|\bar{\mathbf{\Phi}}\mathbf{x} - \bar{\mathbf{y}}\|_2^2, \tag{8.13}$$

where $\mathbf{x}$ denotes a vector that consists of $x_p, \ldots, x_{p+P-1}$, $\tilde{\mathbf{F}}$ denotes a submatrix of $\bar{\mathbf{F}}$ (without its first N rows), and $P_{p|p-1}$ denotes the error covariance matrix for the Kalman filter estimate $\widehat{x}_{p|p-1}$ given all the previous measurements [78, 126]. Three plots in Figure 8.1b compare performances of the ℓ_1-homotopy ($*$) and SpaRSA ($\diamond$). The left plot in Figure 8.1b compares the SER for the two solvers. Since both of them solve the same convex program, SERs in reconstructed signals are almost identical. To demonstrate the advantage of our proposed recovery framework (8.9), we present results for the solution of two related recovery problems, for identical signal and measurement settings: 1) Kalman filtering and smoothing problem (8.13) (labeled as LS-Kalman and plotted as $\circ$), which does not take into account the sparsity of the signal. As the results indicate, the Kalman filter estimate is not as good as the one for the ℓ_1-regularized problem in (8.9). 2) Weighted ℓ_1-regularized problem in (8.9) without the dynamic model, which is equivalent to solving (8.9) with $\lambda = 0$, and it exploits only the sparse representation of each x_p in wavelets (results labeled as DWT and plotted as $\times$). We observed a significant degradation in the signals reconstructed without the dynamic model; the results are indeed inferior to the LS-Kalman. The middle plot in Figure 8.1b compares the computational cost of all the algorithms in terms of the total number of matrix-vector multiplications used in the signal reconstruction. We counted one application of the system matrix and its adjoint as one count of AtA, where we approximated the cost of one step in ℓ_1 homotopy as one application of $\mathbf{\Phi}^T\mathbf{\Phi}$. The right plot in Figure 8.1b compares the

MATLAB execution time for each solver. We observed that ℓ_1-homotopy consumed distinctly fewer matrix-vector multiplications and lesser computation time for the signal reconstruction.

Figures 8.2 presents similar results for experiments with `Piece-Regular` signal. Figure 8.2a presents a snapshot of the `Piece-Regular` signal, its reconstruction at $R = 4$ using (8.9), the error in the reconstruction, and a comparison between the reconstruction of (8.9) and (8.13). Three plots in Figure 8.2b compare performance of the two solvers. In these plots we see similar results that the reconstruction error for (8.9) using both ℓ_1-homotopy and SpaRSA is almost identical, but ℓ_1-homotopy performs significantly better in terms of computational cost and execution time. For the DWT experiments with `Piece-Regular` signal, we solved a non-weighted version of (8.9), where we fixed the value of $\mathbf{W}$ as τ.

A brief summary of the results for our experiments is as follows. We observed that combining a linear dynamic model with the ℓ_1-norm regularization for the sparse signal reconstruction provided a much better signal reconstruction compared to the Kalman filter or the ℓ_1-regularized problem without the dynamic model. The computational cost and execution time for ℓ_1-homotopy is significantly smaller than that for SpaRSA. Average number of homotopy steps for updating the solution at every iteration ranges from 3 to 10, and average time for an update ranges from 5 to 13 milliseconds.

CHAPTER IX

NON-NEGATIVE ℓ_1 HOMOTOPY

In this chapter, we discuss how a simple change in the ℓ_1-homotopy algorithm presented in Chapter 3 allows us to solve a positivity-constrained formulation of the ℓ_1-norm minimization program in (3.6). As we explain below, the only change occurs in the definition of δ^+ in (9.10) such that only the elements with positive sign enter the signal support.

Suppose $\mathbf{y}$ is a vector that obeys the following linear model: $\mathbf{y} = \mathbf{\Phi}\bar{\mathbf{x}} + \mathbf{e}$, where $\bar{\mathbf{x}}$ is a sparse signal of interest that has non-negative values, $\mathbf{\Phi}$ is an $M \times N$ system matrix, and $\mathbf{e}$ is a noise vector. We want to solve the following ℓ_1-norm minimization program with a positivity constraint on the estimate $\bar{\mathbf{x}}$:

$$\underset{\mathbf{x}}{\text{minimize}} \; \|\mathbf{W}\mathbf{x}\|_1 + \frac{1}{2}\|\mathbf{\Phi}\mathbf{x} - \mathbf{y}\|_2^2 \quad \text{subject to} \quad \mathbf{x} \succeq 0, \tag{9.1}$$

where $\mathbf{W}$ denotes a diagonal matrix that contains positive weights $\mathbf{w}$ in its diagonal and $\succeq$ denotes an element-wise inequality. Instead of solving (9.1) from scratch, we want to use a non-negative warm-start vector $\widehat{\mathbf{x}}$ (with support $\widehat{\Gamma}$) and solve the following homotopy program:

$$\underset{\mathbf{x}}{\text{minimize}} \; \|\mathbf{W}\mathbf{x}\|_1 + \frac{1}{2}\|\mathbf{\Phi}\mathbf{x} - \mathbf{y}\|_2^2 + (1-\epsilon)\mathbf{u}^T\mathbf{x} \quad \text{subject to} \quad \mathbf{x} \succeq 0. \tag{9.2}$$

We define $\mathbf{u}$ as before:

$$\mathbf{u} \stackrel{\text{def}}{=} -\mathbf{W}\,\widehat{\mathbf{z}} - \mathbf{\Phi}^T(\mathbf{\Phi}\widehat{\mathbf{x}} - \mathbf{y}), \tag{9.3}$$

where $\widehat{\mathbf{z}}$ can be any vector that is defined as $\text{sign}(\widehat{\mathbf{x}}) = 1$ on $\widehat{\Gamma}$ and strictly smaller than one elsewhere. As ϵ changes from 0 to 1, the optimization problem in (9.2) gradually transforms into the one in (9.1), and the solution of (9.2) follows a piece-wise linear homotopy path from $\widehat{\mathbf{x}}$ toward the solution of (9.1). The homotopy procedure for

solving (9.2) is identical to the one described in Algorithm 1 with the only difference that δ^+ is selected to ensure that any new element that is added to $\mathbf{x}^*$ has positive sign.

To demonstrate these facts and derive the homotopy algorithm, we analyze the optimality conditions for (9.2). We write (9.2) in the following equivalent standard form:

$$\underset{\mathbf{x}}{\text{minimize}} \;\; \mathbf{1}^T\mathbf{W}\mathbf{x} + \frac{1}{2}\|\mathbf{\Phi}\mathbf{x} - \mathbf{y}\|_2^2 + (1-\epsilon)\mathbf{u}^T\mathbf{x} \quad \text{subject to} \quad -\mathbf{x} \preceq 0. \tag{9.4}$$

where $\mathbf{1}$ is a vector of all ones. We can write the Lagrangian function for the objective in (9.4) as

$$L(\mathbf{x}, \lambda) = \mathbf{1}^T\mathbf{W}\mathbf{x} + \frac{1}{2}\|\mathbf{\Phi}\mathbf{x} - \mathbf{y}\|_2^2 + (1-\epsilon)\mathbf{u}^T\mathbf{x} - \lambda^T\mathbf{x}, \tag{9.5}$$

where $\lambda \in \mathbb{R}^N$ is the Lagrange dual vector associated with the inequality constraints $-\mathbf{x} \preceq 0$ and $\lambda \succeq 0$. The problem in (9.4) satisfies the strong duality conditions, and any optimal primal-dual solution pair $(\mathbf{x}^*, \lambda^*)$ must satisfy the following Karush-Kuhn-Tucker (KKT) conditions [26]:

$$\nabla_{\mathbf{x}} L = \mathbf{W}\mathbf{1} + \mathbf{\Phi}^T(\mathbf{\Phi}\mathbf{x}^* - \mathbf{y}) + (1-\epsilon)\mathbf{u} - \lambda^* = 0, \tag{9.6a}$$

$$-\mathbf{x}^* \preceq 0, \tag{9.6b}$$

$$\lambda^* \succeq 0, \tag{9.6c}$$

$$\mathbf{x}_i^*\lambda_i^* = 0, \quad i = 1, \ldots, N. \tag{9.6d}$$

The first equation is the stationarity condition, which states that the gradient of the Lagrangian function with respect to $\mathbf{x}$ must vanish at the optimal solution; the second and the third inequality corresponds to the primal and the dual feasibility, respectively; and the fourth one is the so-called complementary slackness condition, which states that if a primal constraint is active at index i such that $\mathbf{x}_i^* > 0$, then the corresponding $\lambda_i^* = 0$, and if $\lambda_i^* > 0$, then the corresponding $\mathbf{x}_i^* = 0$.

The KKT conditions imply that, for any given value of $\epsilon \in [0,\ 1]$, a vector $\mathbf{x}^*$ (with support Γ) is an optimal solution of (9.2) if it satisfies the following optimality

conditions:

$$\phi_i^T(\mathbf{\Phi}\mathbf{x}^* - \mathbf{y}) + (1-\epsilon)\mathbf{u}_i + \mathbf{w}_i = 0 \qquad \text{for all } i \in \Gamma \tag{9.7a}$$

$$\phi_i^T(\mathbf{\Phi}\mathbf{x}^* - \mathbf{y}) + (1-\epsilon)\mathbf{u}_i + \mathbf{w}_i \geq 0 \qquad \text{for all } i \in \Gamma^c, \tag{9.7b}$$

where ϕ_i denotes i^{th} column of $\mathbf{\Phi}$. Note that we used the stationarity condition (i.e, $\nabla_{\mathbf{x}} L = 0$) with the complementary slackness condition (i.e., $\lambda_\Gamma^* = 0$) in (9.7a) and with the dual feasibility condition (i.e., $\lambda^* \succeq 0$) in (9.7b). In contrast with the optimality conditions in (3.10) (i.e., without the positivity constraints), the conditions in (9.7) impose one-sided constraints on $\phi_i^T(\mathbf{\Phi}\mathbf{x} - \mathbf{y}) + (1-\epsilon)\mathbf{u}_i$—only on the smallest negative value and not on the absolute value.

As we increase ϵ by a small value δ, the solution moves in a direction $\partial\mathbf{x}$, which to maintain optimality must obey

$$\phi_i^T(\mathbf{\Phi}\mathbf{x}^* - \mathbf{y}) + (1-\epsilon)\mathbf{u}_i + \delta(\phi_i^T\mathbf{\Phi}\partial\mathbf{x} - \mathbf{u}_i) = -\mathbf{w}_i \qquad \text{for all } i \in \Gamma \tag{9.8a}$$

$$\underbrace{\phi_i^T(\mathbf{\Phi}\mathbf{x}^* - \mathbf{y}) + (1-\epsilon)\mathbf{u}_i}_{\mathbf{p}_i} + \delta\underbrace{(\phi_i^T\mathbf{\Phi}\partial\mathbf{x} - \mathbf{u}_i)}_{\mathbf{d}_i} \geq -\mathbf{w}_i \qquad \text{for all } i \in \Gamma^c, \tag{9.8b}$$

The update direction that keeps the solution optimal as we change δ is the same as in (3.12):

$$\partial\mathbf{x} = \begin{cases} (\mathbf{\Phi}_\Gamma^T\mathbf{\Phi}_\Gamma)^{-1}\mathbf{u}_\Gamma & \text{on } \Gamma \\ 0 & \text{otherwise.} \end{cases} \tag{9.9}$$

We can move in direction $\partial\mathbf{x}$ until either one of the constraints in (9.8b) is violated, indicating that we must add an element to the support Γ, or one of the nonzero elements in $\mathbf{x}^*$ shrinks to zero, indicating that we must remove an element from Γ. The smallest step-size that causes one of these changes in the support can be easily computed as $\delta^* = \min(\delta^+, \delta^-)$, where

$$\delta^+ = \min_{i \in \Gamma^c} \left(\frac{-\mathbf{w}_i - \mathbf{p}_i}{\mathbf{d}_i} \right)_+ \tag{9.10a}$$

and

$$\delta^- = \min_{i \in \Gamma} \left(\frac{-\mathbf{x}_i^*}{\partial\mathbf{x}_i} \right)_+, \tag{9.10b}$$

and $\min(\cdot)_+$ means that the minimum is taken over only positive arguments. Note that the definition of δ^+ is the only difference between the homotopy algorithm here and the one in Chapter 3. δ^+ is the smallest step-size that causes an inactive constraint to become active at index γ^+, indicating that γ^+ should enter the support, and δ^- is the smallest step-size that shrinks an existing element at index γ^- to zero, indicating that γ^- should leave the support. The new critical value of ϵ becomes $\epsilon + \delta^*$ and the new signal estimate $\mathbf{x}^*$ becomes $\mathbf{x}^* + \delta^* \partial\mathbf{x}$, and its support and sign sequence are updated accordingly. If γ^+ is added to the support, we check whether the value of $\partial\mathbf{x}_{\gamma^+}$ is positive at the next iteration; if $\partial\mathbf{x}_{\gamma^+}$ is negative, we immediately remove γ^+ from the support and recompute the update direction $\partial\mathbf{x}$.

At every step along the homotopy path, we compute the update direction, the step-size, and the consequent one-element change in the support. We repeat this procedure until ϵ is equal to 1.

CHAPTER X

THE DANTZIG SELECTOR ℓ_1 HOMOTOPY

In this chapter, we present a homotopy algorithm similar to the one discussed in Chapter 3 that can dynamically update the ℓ_1 program for the Dantzig selector as the system parameters change. The Dantzig selector (DS) is an ℓ_1-norm minimization program with near-optimal performance guarantees for the recovery of sparse signals from linear, incoherent measurements in the presence of noise [34]. Suppose $\mathbf{y}$ is a vector that obeys the following linear model: $\mathbf{y} = \mathbf{\Phi}\bar{\mathbf{x}} + \mathbf{e}$, where $\bar{\mathbf{x}}$ is a sparse, unknown signal of interest, $\mathbf{\Phi}$ is an $M \times N$ system matrix, and $\mathbf{e}$ is a noise vector. The DS solves the following ℓ_1-norm minimization problem to estimate $\bar{\mathbf{x}}$:

$$\underset{\mathbf{x}}{\text{minimize}} \; \|\mathbf{x}\|_1 \;\; \text{subject to} \;\; \|\mathbf{\Phi}^T(\mathbf{\Phi}\mathbf{x} - \mathbf{y})\|_\infty \leq \tau, \tag{10.1}$$

where τ denotes a regularization parameter. This is a convex optimization problem that can be recast into a linear program for real data and solved using an appropriate solver [20, 26, 41]. Homotopy algorithms for solving (10.1) have also been proposed in [7, 10, 73] and an equivalence between the LASSO and the Dantzig selector homotopy has also been discussed in [3].

10.1 Primal-dual formulation

In [10], we proposed the primal-dual pursuit homotopy algorithm for solving (10.1), which uses both the primal and the dual formulation of (10.1) and the strong duality between their objectives. In this section, we discuss a homotopy algorithm for solving the following modified form of the standard DS program in (10.1):

$$\underset{\mathbf{x}}{\text{minimize}} \; \|\mathbf{W}\mathbf{x}\|_1 \quad \text{subject to} \quad |\mathbf{\Phi}^T(\mathbf{\Phi}\mathbf{x} - \mathbf{y})| \preceq \mathbf{q}, \tag{10.2}$$

where $\mathbf{W}$ denotes a diagonal matrix that contains positive weights $\mathbf{w}$, $\mathbf{q}$ denotes a vector with a set of positive regularization parameters, and $\preceq$ denotes an element-wise inequality. Instead of solving (10.3) from scratch, we want to expedite the process by using some prior knowledge about the solution of (10.3), and for that we develop a homotopy-based dynamic updating algorithm similar to the one discussed in Algorithm 1. To design the homotopy algorithm, we will also need the dual formulation of (10.2), which we state here without derivation:

$$\underset{\lambda}{\text{maximize}} \; -\lambda^T \mathbf{\Phi}^T \mathbf{y} - \|\mathbf{Q}\lambda\|_1 \quad \text{subject to} \quad |\mathbf{\Phi}^T \mathbf{\Phi} \lambda| \preceq \mathbf{w}, \tag{10.3}$$

where $\lambda \in \mathbb{R}^N$ is the dual optimization variable and $\mathbf{Q}$ is a diagonal matrix that contains $\mathbf{q}$ in its diagonal.

We treat the given warm-start primal-dual pair $\widehat{\mathbf{x}}$, $\widehat{\lambda}$ as a starting point and solve a primal-dual pair of homotopy programs for the DS in (10.2) and (10.3): We formulate the homotopy program for the primal problem as

$$\underset{\mathbf{x}}{\text{minimize}} \; \|\mathbf{W}\mathbf{x}\|_1 + (1-\epsilon)\mathbf{u}^T\mathbf{x} \quad \text{subject to} \quad |\mathbf{\Phi}^T(\mathbf{\Phi}\mathbf{x} - \mathbf{y}) + (1-\epsilon)\mathbf{v}| \preceq \mathbf{q}, \tag{10.4}$$

for which the dual program can be written as

$$\underset{\lambda}{\text{maximize}} \; -\lambda^T(\mathbf{\Phi}^T\mathbf{y} - (1-\epsilon)\mathbf{v}) - \|\mathbf{Q}\lambda\|_1 \quad \text{subject to} \quad |\mathbf{\Phi}^T\mathbf{\Phi}\lambda + (1-\epsilon)\mathbf{u}| \preceq \mathbf{w}. \tag{10.5}$$

We define $\mathbf{u}$ and $\mathbf{v}$ such that the warm-start vectors $\widehat{\mathbf{x}}, \widehat{\lambda}$ (with respective supports $\widehat{\Gamma}_x, \widehat{\Gamma}_\lambda$) are optimal solutions for the problems in (10.4) and (10.5) at $\epsilon = 0$. Using the optimality conditions for (10.4) and (10.5), discussed in (10.15) below, we define such $\mathbf{u}, \mathbf{v}$ as

$$\mathbf{u} \stackrel{\text{def}}{=} -\mathbf{W}\,\widehat{\mathbf{z}}_x - \mathbf{\Phi}^T\mathbf{\Phi}\widehat{\lambda}, \tag{10.6a}$$

$$\mathbf{v} \stackrel{\text{def}}{=} \mathbf{Q}\,\widehat{\mathbf{z}}_\lambda - \mathbf{\Phi}^T(\mathbf{\Phi}\widehat{\mathbf{x}} - \mathbf{y}), \tag{10.6b}$$

where $\widehat{\mathbf{z}}_x, \widehat{\mathbf{z}}_\lambda$ can be any vectors that are defined as $\text{sign}(\widehat{\mathbf{x}}), \text{sign}(\widehat{\lambda})$ on their respective supports $\widehat{\Gamma}_x, \widehat{\Gamma}_\lambda$ and strictly smaller than one elsewhere. As ϵ changes from 0 to 1,

the optimization problems in (10.4) and (10.5) gradually transform into the ones in (10.2) and (10.3), and the solutions of (10.4), (10.5) follow piece-wise linear homotopy paths (with sharp discontinuities at critical values of ϵ) from $\widehat{\mathbf{x}}, \widehat{\lambda}$ toward the solutions of (10.4), (10.5). To demonstrate these facts and derive the homotopy algorithm, we discuss below a derivation of the dual formulation in (10.5) and the optimality conditions that any primal-dual solution pair must satisfy.

10.1.1 Dual problem derivation

To derive the dual problem for (10.4), we first recast it into the following linear program:

$$\begin{aligned} \underset{\mathbf{x},\mathbf{s}}{\text{minimize}}\ \mathbf{1}^T\mathbf{s} + (1-\epsilon)\mathbf{u}^T\mathbf{x} \quad \text{subject to} \quad \mathbf{W}\mathbf{x} &\preceq \mathbf{s}, \\ -\mathbf{W}\mathbf{x} &\preceq \mathbf{s}, \\ \mathbf{\Phi}^T(\mathbf{\Phi}\mathbf{x}-\mathbf{y}) + (1-\epsilon)\mathbf{v} &\preceq \mathbf{q}, \\ -\mathbf{\Phi}^T(\mathbf{\Phi}\mathbf{x}-\mathbf{y}) - (1-\epsilon)\mathbf{v} &\preceq \mathbf{q}. \end{aligned} \tag{10.7}$$

The Lagrangian function for this problem can be written as

$$\begin{aligned} L(\mathbf{x},\mathbf{s},\lambda_+,\lambda_-,\eta_+,\eta_-) = \mathbf{1}^T\mathbf{s} + (1-\epsilon)\mathbf{u}^T\mathbf{x} + \eta_+^T(\mathbf{W}\mathbf{x}-\mathbf{s}) + \eta_-^T(-\mathbf{W}\mathbf{x}-\mathbf{s}) \\ +\lambda_+^T(\mathbf{\Phi}^T(\mathbf{\Phi}\mathbf{x}-\mathbf{y}) + (1-\epsilon)\mathbf{v} - \mathbf{q}) + \lambda_-^T(-\mathbf{\Phi}^T(\mathbf{\Phi}\mathbf{x}-\mathbf{y}) - (1-\epsilon)\mathbf{v} - \mathbf{q}), \end{aligned} \tag{10.8}$$

the Lagrange dual function can be written as

$$g(\lambda_+,\lambda_-,\eta_+,\eta_-) = \inf_{\mathbf{x},\mathbf{s}}\ L(\mathbf{x},\mathbf{s},\lambda_+,\lambda_-,\eta_+,\eta_-), \tag{10.9}$$

and the Lagrange dual problem can be written as

$$\underset{\lambda_+,\lambda_-,\eta_+,\eta_-}{\text{maximize}}\ g(\lambda_+,\lambda_-,\eta_+,\eta_-) \quad \text{subject to} \quad \lambda_+ \succeq 0, \lambda_- \succeq 0, \eta_+ \succeq 0, \eta_- \succeq 0, \tag{10.10}$$

where $\lambda_+, \lambda_-, \eta_+, \eta_-$ are the dual variables associated with the inequality constraints in (10.7) [26, Ch. 5]. To compute the Lagrange dual function, we minimize L over

$\mathbf{x}, \mathbf{s}$ by setting its gradient to zero as follows.

$$\nabla_{\mathbf{x},\mathbf{s}} L = \begin{bmatrix} (1-\epsilon)\mathbf{u} + \mathbf{W}\eta_+ - \mathbf{W}\eta_- + \mathbf{\Phi}^T\mathbf{\Phi}\lambda_+ - \mathbf{\Phi}^T\mathbf{\Phi}\lambda_- \\ \mathbf{1} - \eta_+ - \eta_- \end{bmatrix} = 0, \tag{10.11}$$

which provides us with a set of additional feasibility condition. The dual problem can now be written as

$$\begin{aligned} \underset{\lambda_+,\lambda_-,\eta_+,\eta_-}{\text{maximize}} \;\; & (\lambda_+ - \lambda_-)^T(-\mathbf{\Phi}^T\mathbf{y} + (1-\epsilon)\mathbf{v}) - (\lambda_+ + \lambda_-)^T\mathbf{q} \\ \text{subject to} \;\; & \mathbf{\Phi}^T\mathbf{\Phi}(\lambda_+ - \lambda_-) + (1-\epsilon)\mathbf{u} = -\mathbf{W}(\eta_+ - \eta_-) \\ & \eta_+ + \eta_- = \mathbf{1} \\ & \lambda_+ \succeq 0, \lambda_- \succeq 0, \eta_+ \succeq 0, \eta_- \succeq 0. \end{aligned} \tag{10.12}$$

To represent the dual problem in (10.12) in the form described in (10.5), we first define two variables $\lambda \stackrel{\text{def}}{=} \lambda_+ - \lambda_-$ and $\eta \stackrel{\text{def}}{=} \eta_+ - \eta_-$, which are unconstrained in sign. Then we write the equivalent form of the second equality constraint as $\|\eta\|_\infty \leq 1$, using the fact that because of the complementary slackness condition and the non-negativity constraints, η_+, η_- cannot be nonzero for the same index. Finally, we eliminate η by writing all the constraints in (10.12) as the following set of inequality constraints: $|\mathbf{\Phi}^T\mathbf{\Phi}\lambda + (1-\epsilon)\mathbf{u}| \preceq \mathbf{w}$, which leaves λ as the only dual optimization vector. The second term in the objective of (10.12) can be written in the following equivalent form: $(\lambda_+ + \lambda_-)^T\mathbf{q} \equiv \|\mathbf{Q}\lambda\|_1$. This is because λ_+, λ_- are dual variables associated with two complementary inequality constraints in (10.7), and they are nonzero only for the active primal constraints (due to complementary slackness), λ_+ and λ_- cannot be nonzero for the same index as long as all the entries in $\mathbf{q}$ are nonzero.

10.1.2 Optimality conditions

To derive the optimality conditions, we use the strong duality between the objectives in the primal and dual problems in (10.4) and (10.5). Any primal-dual solution pair

$\mathbf{x}^*, \lambda^*$ should satisfy the following equality:

$$\|\mathbf{W}\mathbf{x}^*\|_1 + \langle \mathbf{x}^*, (1-\epsilon)\mathbf{u}\rangle = -\langle \lambda^*, \mathbf{\Phi}^T\mathbf{y} - (1-\epsilon)\mathbf{v}\rangle - \|\mathbf{Q}\lambda^*\|_1, \tag{10.13}$$

which can be equivalently written as

$$\begin{aligned}\|\mathbf{W}\mathbf{x}^*\|_1 + \|\mathbf{Q}\lambda^*\|_1 &= -\langle \mathbf{x}^*, (1-\epsilon)\mathbf{u}\rangle - \langle \lambda^*, \mathbf{\Phi}^T\mathbf{y} - (1-\epsilon)\mathbf{v}\rangle + \langle \lambda^*, \mathbf{\Phi}^T\mathbf{\Phi}\mathbf{x}^* - \mathbf{\Phi}^T\mathbf{\Phi}\mathbf{x}^*\rangle \\ &= -\langle \mathbf{x}^*, \mathbf{\Phi}^T\mathbf{\Phi}\lambda^* + (1-\epsilon)\mathbf{u}\rangle + \langle \lambda^*, \mathbf{\Phi}^T(\mathbf{\Phi}\mathbf{x}^* - \mathbf{y}) + (1-\epsilon)\mathbf{v}\rangle.\end{aligned} \tag{10.14}$$

The complementary slackness condition implies that only those elements in $\mathbf{x}^*$ would be nonzero for which the corresponding dual constraints are active (i.e., hold with equality); similarly, only those elements in λ^* would be nonzero for which the corresponding primal constraints are active. Thus, using (10.14) and the primal and dual feasibility conditions in (10.4) and (10.5), respectively, we get the following optimality conditions that any primal-dual solution pair $\mathbf{x}^*, \lambda^*$ must satisfy at any given value of ϵ:

$$\mathbf{\Phi}^T(\mathbf{\Phi}\mathbf{x}^* - \mathbf{y}) + (1-\epsilon)\mathbf{v}_i = \mathbf{Q}\mathbf{z}_\lambda \tag{10.15a}$$

$$|\phi_i^T(\mathbf{\Phi}\mathbf{x}^* - \mathbf{y}) + (1-\epsilon)\mathbf{v}_i| \le \mathbf{q}_i \qquad \text{for all } i \in \Gamma_\lambda^c, \tag{10.15b}$$

$$\mathbf{\Phi}_i^T\mathbf{\Phi}\lambda^* + (1-\epsilon)\mathbf{u}_i = -\mathbf{W}\mathbf{z}_x \tag{10.15c}$$

$$|\phi_i^T\mathbf{\Phi}\lambda^* + (1-\epsilon)\mathbf{u}_i| \le \mathbf{w}_i \qquad \text{for all } i \in \Gamma_x^c, \tag{10.15d}$$

where ϕ_i denotes i^{th} column of $\mathbf{\Phi}$, Γ_x, Γ_λ denote the supports of $\mathbf{x}^*, \lambda^*$, and $\mathbf{z}_x, \mathbf{z}_\lambda$ are vectors that are same as signs of primal-dual vectors on their respective supports and strictly smaller than one elsewhere. The conditions in (10.15a) are the constraints that the primal solution $\mathbf{x}^*$ satisfies with equality over the support of the dual solution λ^* and strict inequality elsewhere, whereas the conditions in (10.15c) are the constraints that the dual solution λ^* satisfies with equality on the support of the primal solution $\mathbf{x}^*$ and strict inequality elsewhere. Equivalently, active primal constraints determine the support and the sign sequence for the dual solution and the dual active constraints

determine the support and the sign sequence of the primal solution. Note that, the definitions of $\mathbf{u}, \mathbf{v}$ in (10.6) ensure that the primal-dual warm-start pair $\widehat{\mathbf{x}}, \widehat{\lambda}$ satisfies the optimality conditions in (10.15) at $\epsilon = 0$; hence, they are valid initial solutions. It is also evident from (10.15) that at any value of ϵ the solution pair $\mathbf{x}^*, \lambda^*$ is completely described by the supports Γ_x, Γ_λ and the sign sequences $\mathbf{z}_x, \mathbf{z}_\lambda$ (assuming that $(\mathbf{\Phi}_{\Gamma_\lambda}^T \mathbf{\Phi}_{\Gamma_x})^{-1}$ exists). The support of the primal and the dual vector changes only at certain critical values of ϵ, when either a new element enters the support or an existing nonzero element shrinks to zero. These critical values of ϵ are easy to calculate at any point along the homotopy path, and the entire path (parameterized by ϵ) can be traced in a sequence of computationally inexpensive homotopy steps.

10.2 Primal-dual ℓ_1-homotopy

Every step in the homotopy algorithm for the primal-dual formulation consists of two phases. In the first phase, we increase ϵ while updating both the primal and the dual solutions, until there is a change in the support of either one of them, which indicates the critical value of ϵ. In the second phase, we fix ϵ and update either the primal or the dual vector, depending on the change in the support during the first phase, so that the size of the primal and the dual support remains the same. We repeat this procedure until ϵ is equal to 1. A pseudocode outlining the homotopy procedure is presented in Algorithm 4.

10.2.1 Phase 1

In the first phase of every homotopy step, as we increase ϵ by a small value δ, the primal-dual solutions move in directions $\partial\mathbf{x}, \partial\lambda$, which to maintain optimality must satisfy the following conditions:

$$\mathbf{\Phi}^T(\mathbf{\Phi}\mathbf{x}^* - \mathbf{y}) + (1-\epsilon)\mathbf{v}_i + \delta(\mathbf{\Phi}^T\mathbf{\Phi}\partial\mathbf{x} - \mathbf{v}) = \mathbf{Q}\mathbf{z}_\lambda \tag{10.16a}$$

$$|\underbrace{\phi_i^T(\mathbf{\Phi}\mathbf{x}^* - \mathbf{y}) + (1-\epsilon)\mathbf{v}_i}_{\mathbf{p}_i} + \delta\underbrace{(\phi_i^T\mathbf{\Phi}\partial\mathbf{x} - \mathbf{v}_i)}_{\mathbf{d}_i}| \leq \mathbf{q}_i \qquad \text{for all } i \in \Gamma_\lambda^c, \tag{10.16b}$$

$$\mathbf{\Phi}^T\mathbf{\Phi}\lambda^* + (1-\epsilon)\mathbf{u}_i + \delta(\mathbf{\Phi}^T\mathbf{\Phi}\partial\lambda - \mathbf{u}) = -\mathbf{W}\mathbf{z}_x \tag{10.16c}$$

$$|\underbrace{\phi_i^T\mathbf{\Phi}\lambda^* + (1-\epsilon)\mathbf{u}_i}_{\alpha_i} + \delta\underbrace{(\phi_i^T\mathbf{\Phi}\partial\lambda - \mathbf{u}_i)}_{\beta_i}| \leq \mathbf{w}_i \qquad \text{for all } i \in \Gamma_x^c, \tag{10.16d}$$

The update directions that keep the primal-dual solution pair optimal as we change δ can be written as

$$\partial\mathbf{x} = \begin{cases} (\mathbf{\Phi}_{\Gamma_\lambda}^T\mathbf{\Phi}_{\Gamma_x})^{-1}\mathbf{v}_{\Gamma_\lambda} & \text{on } \Gamma_x \\ 0 & \text{otherwise.} \end{cases} \tag{10.17}$$

$$\partial\lambda = \begin{cases} (\mathbf{\Phi}_{\Gamma_x}^T\mathbf{\Phi}_{\Gamma_\lambda})^{-1}\mathbf{u}_{\Gamma_x} & \text{on } \Gamma_\lambda \\ 0 & \text{otherwise.} \end{cases} \tag{10.18}$$

We can change $\mathbf{x}^*$ in the direction $\partial\mathbf{x}$ until either one of the constraints in (10.16b) is violated, indicating that we must add an element to the dual support Γ_λ, or one of the nonzero elements in $\mathbf{x}^*$ shrinks to zero, indicating that we must remove an element from Γ_x. Similarly, we can change λ^* in the direction $\partial\lambda$ until either one of the constraints in (10.16d) is violated, indicating that we must add an element to the primal support Γ_x, or one of the nonzero elements in λ^* shrinks to zero, indicating that we must remove an element from Γ_λ. The smallest step-size that causes one of these changes in the support can be easily computed as $\delta^* = \min(\delta_x^+, \delta_x^-, \delta_\lambda^+, \delta_\lambda^-)$, where[1]

$$\delta_\lambda^+ = \min_{i\in\Gamma_\lambda^c}\left(\frac{\mathbf{q}_i - \mathbf{p}_i}{\mathbf{d}_i}, \frac{-\mathbf{q}_i - \mathbf{p}_i}{\mathbf{d}_i}\right)_+, \tag{10.19a}$$

$$\delta_x^- = \min_{i\in\Gamma_x}\left(\frac{-\mathbf{x}_i^*}{\partial\mathbf{x}_i}\right)_+, \tag{10.19b}$$

$$\delta_x^+ = \min_{i\in\Gamma_x^c}\left(\frac{\mathbf{w}_i - \alpha_i}{\beta_i}, \frac{-\mathbf{w}_i - \alpha_i}{\beta_i}\right)_+, \tag{10.19c}$$

and

$$\delta_\lambda^- = \min_{i\in\Gamma_\lambda}\left(\frac{-\lambda_i^*}{\partial\lambda_i}\right)_+, \tag{10.19d}$$

[1]To include the positivity constraint in the optimization problem (10.4), we initialize the homotopy with a non-negative (feasible) warm-start vector $\widehat{\mathbf{x}}$ and define $\delta_x^+ = \min_{i\in\Gamma_x^c}\left(\frac{-\mathbf{w}_i-\alpha_i}{\beta_i}\right)_+$ so that only positive elements are added to $\mathbf{x}^*$. See Section 10.3 for further explanation.

and $\min(\cdot)_+$ means that the minimum is taken over only positive arguments. δ_λ^+ and δ_x^+ are the smallest step-sizes that would cause an inactive primal and dual constraint to become active and indicate the new element that should enter the support Γ_λ or Γ_x and its sign. δ_x^- and δ_λ^- are the smallest step-sizes that would shrink an existing element in $\mathbf{x}^*$ and λ^* to zero and indicate the element that should leave the support Γ_x or Γ_λ. We compute the smallest stepsize δ^* and identify the associated change in the support of the primal or the dual vector. The new critical value of ϵ becomes $\epsilon + \delta^*$ and the new primal-dual estimates become $\mathbf{x}^* + \delta^* \partial\mathbf{x}$, $\lambda^* + \delta^* \partial\lambda$.

10.2.2 Phase 2

In the second phase of the homotopy step, we keep ϵ fixed and identify the change in either the primal or the dual support by updating λ^* or $\mathbf{x}^*$ such that the sizes of Γ_x and Γ_λ remain the same. For instance, if a new element is selected to add into Γ_λ or an existing element is selected to remove from Γ_x during the first phase, we must either add a new element in Γ_x or remove an element from Γ_λ during the second phase, for which we fix $\mathbf{x}^*$ and update λ^*. Alternatively, if a new element is selected to add into Γ_x or an element is selected to remove from Γ_λ during the first phase, we must either add a new element in Γ_λ or remove an element from Γ_x during the second phase, for which we fix λ^* and update $\mathbf{x}^*$. To update $\mathbf{x}^*$ during the second phase, we use the primal optimality conditions in (10.15a); to update λ^*, we use the dual optimality conditions in (10.15c). We discuss the four possible scenarios below.

$\boxed{\delta^* = \delta_\lambda^+ :}$ Suppose an element at index γ^+ with sign $\mathbf{z}_{\gamma^+}$ is identified as the new element that should enter Γ_λ during the first phase, which happens if $\delta^* = \delta_\lambda^+$ in (10.19). Thus, we fix $\mathbf{x}^*$ and change λ^* on the updated support $[\Gamma_\lambda \ \gamma^+]$ to identify either a new element to add into Γ_x or an element to remove from Γ_λ. The dual feasibility condition in (10.15c) provides us one degree of freedom, because of the newly identified element γ^+ in λ^* for which the sign is also available, such that we

can change λ^* on the updated support without violating the active constraints on Γ_x. To maintain the active constraints on Γ_x in (10.15c), the update direction $\partial\lambda$ must satisfy the following condition:

$$\mathbf{\Phi}_{\Gamma_x}^T \mathbf{\Phi}_{\Gamma_\lambda} \partial\lambda_{\Gamma_\lambda} + \theta \mathbf{\Phi}_{\Gamma_x} \phi_{\gamma^+} \mathbf{z}_{\gamma^+} = 0, \tag{10.20}$$

from which we can compute the update direction as

$$\partial\lambda = \begin{cases} -(\mathbf{\Phi}_{\Gamma_x}^T \mathbf{\Phi}_{\Gamma_\lambda})^{-1} \mathbf{\Phi}_{\Gamma_x}^T \phi_{\gamma^+} \mathbf{z}_{\gamma^+} & \text{on } \Gamma_\lambda \\ \mathbf{z}_{\gamma^+} & \text{on } \gamma^+ \\ 0 & \text{elsewhere,} \end{cases} \tag{10.21}$$

where $\theta > 0$ denotes the step size. As we change λ^* in the direction $\partial\lambda$ by changing the stepsize θ, either one of the following dual constraints becomes active:

$$| \underbrace{\phi_i^T \mathbf{\Phi} \lambda^* + (1-\epsilon)\mathbf{u}_i}_{\alpha_i} + \theta \underbrace{\phi_i^T \mathbf{\Phi} \partial\lambda}_{\beta_i} | \leq \mathbf{w}_i \text{ for all } i \in \Gamma_x^c, \tag{10.22}$$

indicating the new element that must be added to Γ_x, or an existing element in λ^* shrinks to zero. The smallest step-size that causes one of these changes in the support can be easily computed as $\theta^* = \min(\theta_x^+, \theta_\lambda^-)$, where[2] $\theta_x^+, \theta_\lambda^-$ are defined same as $\delta_x^+, \delta_\lambda^-$ in (10.19), but with a different definition of α, β:

$$\theta_x^+ = \min_{i \in \Gamma_x^c} \left(\frac{\mathbf{w}_i - \alpha_i}{\beta_i}, \frac{-\mathbf{w}_i - \alpha_i}{\beta_i} \right)_+ \tag{10.23a}$$

and

$$\theta_\lambda^- = \min_{i \in \Gamma_\lambda} \left(\frac{-\lambda_i^*}{\partial\lambda_i} \right)_+ . \tag{10.23b}$$

We compute the smallest stepsize θ^*, identify the associated change in the support of the primal or the dual vector, and update Γ_x and Γ_λ accordingly. The new optimal dual solution becomes $\lambda^* + \theta^* \partial\lambda$.

[2]Similarly, for non-negative $\mathbf{x}^*$, we define $\theta_x^+ = \min_{i \in \Gamma_x^c} \left(\frac{-\mathbf{w}_i - \alpha_i}{\beta_i} \right)_+$ so that only positive elements are added to the signal support. See Section 10.3 for further explanation.

$\boxed{\delta^* = \delta_x^- :}$ Suppose instead of selecting an element to add into Γ_λ, we identified an element γ^- to remove from Γ_x during the first phase, which happens if $\delta^* = \delta_x^-$ in (10.19). The dual feasibility condition in (10.15c) still provides us one degree of freedom, because the constraint on $\gamma^- \in \Gamma_x$ has now become inactive, and we can change λ^* on Γ_λ without violating the active constraints on $\Gamma_x \backslash \gamma^-$. We remove γ^- from Γ_x, pick an element γ^+ out of Γ_λ and treat that as the new element to be included in Γ_λ. While selecting γ^+, we must ensure that the matrix $\mathbf{\Phi}_{\Gamma_x}^T \mathbf{\Phi}_{\Gamma_\lambda}$ remains well-conditioned and invertible, which can be confirmed by checking that the inverse of its Schur complement must be nonzero [10]. Using the updated Γ_x, Γ_λ, and $\mathbf{z}_{\gamma^+} = 1$, we compute the update direction, $\partial\lambda$, in (10.21). Then we can compute θ^*, as described in (10.23), and identify the associated change in the support of the primal or the dual vector, and update Γ_x and Γ_λ accordingly. The new optimal dual solution becomes $\lambda^* + \theta^* \partial\lambda$. Since there is an ambiguity about the *actual* sign of $\partial\lambda$ on γ^+, we have to flip the sign of $\partial\lambda$ while computing θ^* if $\text{sign}(\alpha_{\gamma^-}) = \text{sign}(\beta_{\gamma^-})$. Note that $\lambda_{\gamma^+}^*$ is not zero, and it can shrink to zero as well.

$\boxed{\delta^* = \delta_x^+ :}$ If an element at index γ^+ with sign $\mathbf{z}_{\gamma^+}$ is identified as a new element for the support Γ_x during the first phase, which happens if $\delta^* = \delta_x^+$ in (10.19), we fix λ^* and update $\mathbf{x}^*$ on the updated support $[\Gamma_x \ \gamma^+]$, using the procedure similar to that we discussed for updating λ^* if $\delta^* = \delta_\lambda^+$. The only difference is that here we use the primal optimality conditions in (10.15a) to compute the update direction and the step size. The update direction can be computed as

$$\partial\mathbf{x} = \begin{cases} -(\mathbf{\Phi}_{\Gamma_\lambda}^T \mathbf{\Phi}_{\Gamma_x})^{-1} \mathbf{\Phi}_{\Gamma_\lambda}^T \phi_{\gamma^+} \mathbf{z}_{\gamma^+} & \text{on } \Gamma_x \\ \mathbf{z}_{\gamma^+} & \text{on } \gamma^+ \\ 0 & \text{elsewhere.} \end{cases} \tag{10.24}$$

As we change $\mathbf{x}^*$ in the direction $\partial\mathbf{x}$ by changing the stepsize θ, either one of the

following primal constraints becomes active:

$$|\underbrace{\phi_i^T(\mathbf{\Phi}\mathbf{x}^* - \mathbf{y}) + (1-\epsilon)\mathbf{v}_i}_{\mathbf{p}_i} + \theta \underbrace{\phi_i^T \mathbf{\Phi}\partial\mathbf{x}}_{\mathbf{d}_i}| \leq \mathbf{q}_i \text{ for all } i \in \Gamma_\lambda^c, \tag{10.25}$$

indicating the new element that must be added to Γ_λ, or an existing element in $\mathbf{x}^*$ shrinks to zero. The smallest step-size that causes one of these changes in the support can be easily computed as $\theta^* = \min(\theta_\lambda^+, \theta_x^-)$, where $\theta_\lambda^+, \theta_x^-$ are defined same as $\delta_\lambda^+, \delta_x^-$ in (10.19), but with a different definition of $\mathbf{p}, \mathbf{d}$

$$\theta_\lambda^+ = \min_{i \in \Gamma_\lambda^c} \left(\frac{\mathbf{q}_i - \mathbf{p}_i}{\mathbf{d}_i}, \frac{-\mathbf{q}_i - \mathbf{p}_i}{\mathbf{d}_i} \right)_+ \tag{10.26a}$$

and

$$\theta_x^- = \min_{i \in \Gamma_x} \left(\frac{-\mathbf{x}_i^*}{\partial \mathbf{x}_i} \right)_+ . \tag{10.26b}$$

We compute the smallest stepsize θ^*, identify the associated change in the support of the primal or the dual vector, and update Γ_x and Γ_λ accordingly. The new optimal primal solution becomes $\mathbf{x}^* + \theta^* \partial\mathbf{x}$.

$\boxed{\delta^* = \delta_\lambda^- :}$ If instead of selecting an element to add into Γ_x, we identified an element γ^- to remove from Γ_λ during the first phase, which happens if $\delta^* = \delta_\lambda^-$ in (10.19). The primal feasibility condition in (10.15a) still provides us one degree of freedom, because the constraint on $\gamma^- \in \Gamma_\lambda$ has now become inactive, and we can change $\mathbf{x}^*$ on Γ_x without violating the active constraints on $\Gamma_\lambda \backslash \gamma^-$. We remove γ^- from Γ_λ, pick an element γ^+ out of Γ_x and treat that as the new element to be included in Γ_x. Using the updated Γ_x, Γ_λ, and $\mathbf{z}_{\gamma^+} = 1$, we compute the update direction, $\partial\mathbf{x}$, in (10.24). Then we can compute θ^*, as described in (10.26), and identify the associated change in the support of the primal or the dual vector, and update Γ_x and Γ_λ accordingly. The new optimal dual solution becomes $\mathbf{x}^* + \theta^* \partial\mathbf{x}$. Since there is an ambiguity about the *actual* sign of $\partial\mathbf{x}$ on γ^+, we have to flip the sign of $\partial\mathbf{x}$ while computing θ^* if $\text{sign}(\mathbf{p}_{\gamma^-}) = \text{sign}(\mathbf{d}_{\gamma^-})$. Note that $\mathbf{x}_{\gamma^+}^*$ is not zero, and it can shrink to zero as well.

10.3 Non-negative Dantzig selector

To impose a positivity constraint on the estimate of the DS program in (10.4), we can solve the following modified problem:

$$\begin{aligned} \underset{\mathbf{x}}{\text{minimize}} \;\; & \mathbf{1}^T\mathbf{W}\mathbf{x} + (1-\epsilon)\mathbf{u}^T\mathbf{x} \qquad (10.27) \\ \text{subject to} \;\; & |\mathbf{\Phi}^T(\mathbf{\Phi}\mathbf{x}-\mathbf{y}) + (1-\epsilon)\mathbf{v}| \preceq \mathbf{q}, \;\; \mathbf{x} \succeq 0, \end{aligned}$$

for which the dual program can be written as

$$\begin{aligned} \underset{\lambda}{\text{maximize}} \;\; & -\lambda^T(\mathbf{\Phi}^T\mathbf{y} - (1-\epsilon)\mathbf{v}) - \|\mathbf{Q}\lambda\|_1 \qquad (10.28) \\ \text{subject to} \;\; & \mathbf{\Phi}^T\mathbf{\Phi}\lambda + (1-\epsilon)\mathbf{u} \succeq -\mathbf{w}. \qquad (10.29) \end{aligned}$$

We can derive this dual formulation using the same procedure described in Section 10.1.1. The only change that appears in the non-negative homotopy is the dual optimality condition, where instead of inequality constraints on the absolute values of $\mathbf{\Phi}^T\mathbf{\Phi}\lambda + (1-\epsilon)\mathbf{u}$ (as in (10.15d)), we have one-sided constraints on the smallest negative values. This, in turn, changes the definition of δ_x^+ in (10.19) and θ_x^+ in (10.23), while the rest of the homotopy procedure remains the same.

Algorithm 4 ℓ_1 homotopy for the Dantzig selector

Input: $\mathbf{\Phi}$, $\mathbf{y}$, $\mathbf{W}$, $\mathbf{Q}$, $\widehat{\mathbf{x}}$, $\widehat{\lambda}$, $\mathbf{u}$, and $\mathbf{v}$ (optional: inverse or QR factors of $\mathbf{\Phi}_{\widehat{\Gamma}_\lambda}^T \mathbf{\Phi}_{\widehat{\Gamma}_x}$)
Output: x^*

Initialize: $\epsilon = 0$, $\mathbf{x}^* \leftarrow \widehat{\mathbf{x}}$, $\lambda^* \leftarrow \widehat{\lambda}$
Repeat:

Phase 1: ▷ Change ϵ in the first phase
Compute $\partial\mathbf{x}, \partial\lambda$ in (10.17),(10.18) ▷ Primal-dual update directions
Compute $\mathbf{p}$,$\mathbf{d}$, α, β in (10.16) ▷ Primal-dual constraints
Compute $\delta^* = \min(\delta_x^+, \delta_x^-, \delta_\lambda^+, \delta_\lambda^-)$ in (10.19) ▷ Step size
if $\epsilon + \delta^* > 1$ **then**
  $\delta^* \leftarrow 1 - \epsilon$ ▷ Last iteration
  $\mathbf{x}^* \leftarrow \mathbf{x}^* + \delta^* \partial\mathbf{x}$ ▷ Final primal solution
  $\lambda^* \leftarrow \lambda^* + \delta^* \partial\lambda$ ▷ Final dual solution
  `break`
end if
$\mathbf{x}^* \leftarrow \mathbf{x}^* + \delta^* \partial\mathbf{x}$ ▷ Update the primal solution
$\lambda^* \leftarrow \lambda^* + \delta^* \partial\lambda$ ▷ Update the dual solution
$\epsilon \leftarrow \epsilon + \delta^*$ ▷ Update the homotopy parameter

Phase 2: ▷ Fix ϵ in the second phase
if $\delta^* = \delta_\lambda^+$ or $\delta^* = \delta_x^-$ **then** ▷ Fix $\mathbf{x}^*$ and update λ^* only
  Compute $\partial\lambda$ in (10.21) ▷ Dual update direction
  Compute α, β in (10.22)
  Compute $\theta^* = \min(\theta_x^+, \theta_\lambda^-)$ in (10.23)
  $\lambda^* \leftarrow \lambda^* + \theta^* \partial\lambda$ ▷ Update the dual solution
  Update the supports Γ_x, Γ_λ
else ▷ Fix λ^* and update $\mathbf{x}^*$ only
  Compute $\partial\mathbf{x}$ in (10.24) ▷ Primal update direction
  Compute $\mathbf{p}$, $\mathbf{d}$ in (10.25)
  Compute $\theta^* = \min(\theta_\lambda^+, \theta_x^-)$ in (10.26)
  $\mathbf{x}^* \leftarrow \mathbf{x}^* + \theta^* \partial\mathbf{x}$ ▷ Update the primal solution
  Update the supports Γ_x, Γ_λ
end if

until $\epsilon = 1$

CHAPTER XI

ℓ_1 DECODING

In this chapter we discuss dynamic updating for an ℓ_1-norm minimization program that can recover an arbitrary vector $\bar{x}$ from corrupted measurements $y = A\bar{x} + \bar{e}$, given A satisfies some properties and $\bar{e}$ is a sparse error vector. This problem appears in the context of decoding by linear programming [38], which is closely related to the recovery of a sparse vector from an underdetermined system and compressive sensing. While we can compress a sparse signal by applying an underdetermined incoherent matrix, we can also protect a general signal against sparse errors by applying an overdetermined incoherent matrix. Suppose we take $M = CN$ incoherent measurements of an arbitrary signal $\bar{x}$ of length N as $A\bar{x}$, where $C > 1$, and add a sparse error $\bar{e}$ that has fewer than $\rho(C) \cdot M$ non-zero terms, where $\rho(C)$ is a constant that depends on C. Solving the following ℓ_1 *decoding* program (for $y = A\bar{x} + \bar{e}$) can exactly recover $\bar{x}$ [38, 118]:

$$\underset{x}{\text{minimize}} \ \|Ax - y\|_1. \tag{11.1}$$

There exist a number of strong performance guarantees for (11.1) that relate the number of errors we can correct (number of non-zero entries in $\bar{e}$) to the number of measurements we have collected (rows in A) [38, 118]. If the matrix A consists of independent Gaussian random variables, then the number of errors we can correct (and hence recover $\bar{x}$ exactly) scales with the amount of oversampling $M - N$.

In this chapter, we will discuss a homotopy algorithm that can update the solution to the ℓ_1 decoding problem (11.1) as new measurements are added. This algorithm has been published in [14].

11.1 Problem formulation

We will use the language of a communications system: a transmitter is trying to send a message $\bar{x}$ to a receiver. The message is turned into a codeword by applying A, and the received signal $y = A\bar{x} + \bar{e}$ is corrupted by a sparse error vector $\bar{e}$. The receiver recovers the message by solving (11.1). If the codeword is long enough (A has enough rows) and the error is sparse enough (not too many entries of $\bar{e}$ are non-zero), the message will be recovered exactly. The receiver will assume that the true message has been recovered when the error $e = Ax - y$ for the solution to (11.1) has fewer than $M - N$ nonzero terms (in general, the error for the solution will contain exactly $M - N$ terms, and so this degeneracy indicates that the receiver has locked on to something special). If the recovered error has exactly $M - N$ non-zero terms, the receiver asks the transmitter for more measurements (codeword elements).

Suppose that the receiver has just solved (11.1) to estimate a decoded message, and then P new measurements of x are received. The updated system of equations is

$$\begin{bmatrix} y \\ w \end{bmatrix} = \begin{bmatrix} A \\ B \end{bmatrix} \bar{x} + \begin{bmatrix} \bar{e} \\ \bar{d} \end{bmatrix}, \tag{11.2}$$

where w represents P new entries in the received codeword, B denotes P new rows in the coding matrix, and $\bar{d}$ is the error vector for the new codeword entries. The receiver now solves the updated ℓ_1 decoding problem

$$\underset{x}{\text{minimize}} \ \|Ax - y\|_1 + \|Bx - w\|_1. \tag{11.3}$$

Instead of solving (11.3) from scratch, the new measurements can be worked into the solution gradually by solving the following homotopy program:

$$\underset{x}{\text{minimize}} \ \|Ax - y\|_1 + \epsilon\|Bx - w\|_1 \tag{11.4}$$

as ϵ (the homotopy parameter) is increased from 0 to 1.

We will find it convenient to trace the path of both the primal and dual solutions as ϵ increases from 0 to 1. We begin by writing the dual problem for (11.4) as

$$\begin{aligned} &\underset{\lambda,\nu}{\text{maximize}} \quad -\lambda^T y - \epsilon \nu^T w \\ &\text{subject to} \quad A^T\lambda + \epsilon\, B^T\nu = 0, \;\; \|\lambda\|_\infty \leq 1, \; \|\nu\|_\infty \leq 1, \end{aligned} \tag{11.5}$$

where $\lambda \in \mathbb{R}^M$ and $\nu \in \mathbb{R}^P$ are the dual optimization variables [26, 38].

The optimality conditions for (x^*, λ^*, ν^*) to be a primal-dual solution set at a given value of ϵ can be derived as follows. Let $e^* = Ax^* - y$ and $d^* = Bx^* - w$ be the error estimates for the first and second part of the codeword; their supports denoted by Γ_e and Γ_d respectively. Using the fact that, because of the strong duality in a linear program [26], the primal and dual objectives in (11.4) and (11.5) will be equal for the optimal solutions, we get the following conditions for (x^*, λ^*, ν^*):

$$\lambda^* = \text{sign}(Ax^* - y) \quad \text{on } \Gamma_e, \qquad \|\lambda^*\|_\infty < 1 \quad \text{on } \Gamma_e^c \tag{11.6a}$$

$$\nu^* = \text{sign}(Bx^* - w) \quad \text{on } \Gamma_d, \qquad \|\nu_k\|_\infty < 1 \quad \text{on } \Gamma_d^c \tag{11.6b}$$

$$A^T\lambda^* + \epsilon B^T\nu^* = 0. \tag{11.6c}$$

These conditions tell us that the dual vectors lie in the left null space of the combined coding matrix (ignoring the presence of ϵ here), and whenever an entry in the error estimate is non-zero the corresponding dual element is equal to the sign of the error at that location, while the absolute value of dual vectors at all other indices is smaller than 1. In the homotopy scheme for ℓ_1 decoding, we update the active set for the error estimate $\Gamma = \{\Gamma_e \cup \Gamma_d\}$ while satisfying the constraints on the dual vectors as we increase ϵ from 0 to 1.

11.2 Homotopy algorithm

The algorithm for updating the solution for (11.4), (11.5) as ϵ moves from 0 to 1 consists of an initialization procedure followed by alternating updates of the primal

and the dual solutions. The critical points along the homotopy path correspond to the values of ϵ when an element enters or leaves the support of the sparse error vectors (e^* and d^*) corresponding to the optimal solution x^*. We describe each of these stages below.

11.2.1 Initialization

We initialize x^* and λ^* with the primal and dual solutions at $\epsilon = 0$; the old error estimate for the first M codeword elements as $e^* = Ax^* - y$. We initialize the error estimate for the next P elements as $d^* = Bx^* - w$. In general, if we have not yet recovered the underlying message, all of the terms in d^* will be non-zero. Throughout the algorithm, we will use Γ as the index set for the error locations over all $M + P$ codeword elements; we initialize it with $\Gamma = \{\Gamma_e \cup \Gamma_d\}$, where Γ_e is the support of e^*, and Γ_d is the support of d^*. We initialize the dual variable ν^* corresponding to the new errors as $\nu^* = \text{sign}(Bx^* - w)$. Apart from keeping track of the support of the entire error estimate, we will also find it necessary to keep track of which elements from the second part of the error (i.e., d) leave the support at some time. To this end, we initialize a set $\Gamma_n = \Gamma_d$ and if an element in d shrinks to zero, we remove its location from Γ_n (we will never grow Γ_n), which is equivalent to removing the homotopy parameter from in front of the measurements in (11.4) corresponding to that index in d.

Every step of the homotopy algorithm for ℓ_1 decoding can be divided into a primal and a dual update. Assume that we already have primal-dual solutions (x^*, λ^*, ν^*) for the problems in (11.4) and (11.5) for a given value of ϵ between 0 and 1, with supports Γ (corresponding to all the nonzero entries in the error estimate) and Γ_n (corresponding to the entries of d which have remained non-zero so far).

11.2.2 Dual update

Assume that the current error estimate has exactly N terms which are zero, which implies that Γ has size $M + P - N$ and Γ^c has size N, and exactly N entries in the dual vector (λ^*, ν^*) have magnitudes smaller than 1. There are N degrees of freedom for which the dual solution can move during one step of the update; we will exercise this freedom by manipulating the dual coefficients on the set Γ^c.

We combine both parts of the coding matrix together as $G = [A^T \; B^T]$ and both parts of the dual vector together as $\xi^* = \begin{bmatrix} \lambda^* \\ \nu^* \end{bmatrix}$. By dividing ξ into components that correspond to Γ_n (those error locations in d that have remained nonzero so far) and the rest, the optimality condition (11.6c) can be written as

$$G_{\Gamma_n^c} \xi^*_{\Gamma_n^c} + \epsilon \, G_{\Gamma_n} \xi^*_{\Gamma_n} = 0. \tag{11.7}$$

Note that if at any point an element in d shrinks to zero, the corresponding element is removed from Γ_n. Increasing ϵ to $\epsilon + \delta$, the dual solution changes in direction $\partial\xi$ and the optimality conditions change as

$$\begin{aligned} G_{\Gamma_n^c}(\xi^* + \partial\xi)_{\Gamma_n^c} + (\epsilon + \delta) G_{\Gamma_n}(\xi^* + \partial\xi)_{\Gamma_n} &= 0 \\ G_{\Gamma_n^c} \partial\xi_{\Gamma_n^c} + \delta \, G_{\Gamma_n}(\xi^* + \partial\xi)_{\Gamma_n} &= 0. \end{aligned} \tag{11.8}$$

Since $\partial\xi$ can change only on the set Γ^c (i.e., N indices where error vectors e^* and d^* are zero), $\Gamma_n \subset \Gamma$, and $\Gamma^c \subset \Gamma_n^c$, $\partial\xi_{\Gamma_n} = 0$. We can compute the update direction $\partial\xi$ and the step size δ required to change ϵ to $\epsilon + \delta$ as

$$\partial\xi = \begin{cases} -(G_{\Gamma^c})^{-1} G_{\Gamma_n} \xi^*_{\Gamma_n} & \text{on } \Gamma^c \\ 0 & \text{otherwise,} \end{cases} \tag{11.9}$$

As we increase ϵ to $\epsilon + \delta$, the dual solution (ξ^*) changes in the direction $\partial\xi$ with step size δ until one of its element becomes active (equal to +1 or -1) on Γ^c. The

smallest step size for this to happen can be computed as

$$\delta^+ = \min_{i \in \Gamma^c} \left(\frac{1 - \xi_i^*}{\partial \xi_i}, \frac{1 + \xi_i^*}{-\partial \xi_i} \right)_+ . \tag{11.10}$$

The new critical value for ϵ becomes $\epsilon + \delta^+$ and dual solution vector ξ^* becomes $\xi^* + \delta^+ \partial \xi$. Let γ^+ be the index for the minimizer in (11.10). This tells us that we have a new element in the estimated error vector at index γ^+ with the sign, z_{γ^+}, same as $\xi^*_{\gamma^+}$.

11.2.3 Primal update

The dual update provides us with a new element in the support of the error estimate. As the error estimate will have exactly N zero entries until we have recovered the message, we know that one of the elements currently in Γ must shrink to zero. This is accomplished by the primal update.

We have the following system of equations at ϵ

$$\underbrace{\begin{bmatrix} A \\ B \end{bmatrix}}_{G^T} x^* - \underbrace{\begin{bmatrix} y \\ w \end{bmatrix}}_{s} = \underbrace{\begin{bmatrix} e^* \\ d^* \end{bmatrix}}_{c^*}, \tag{11.11}$$

where the current optimal error estimate c^* is supported on the set Γ. The dual update has indicated that our new error estimate will have a new active term at index γ^+, and that the sign of this new term should be z_{γ^+}. We update our estimate of the message x^* such that the updated error estimate c^* has $\text{sign}[c^*_{\gamma^+}] = z_{\gamma^+}$ and c^* remains zero at the remaining indices in Γ^c. In other words, for an update direction ∂x, the system in (11.11) needs to be satisfied with the following conditions on the set Γ^c for a small step size $\delta > 0$:

$$(G^T(x^* + \delta \partial x) - s)_{\Gamma^c} = (c^* + \delta \partial c)_{\Gamma^c}, \tag{11.12}$$

where ∂c is constrained on the set Γ^c as

$$\partial c_{\Gamma^c} = \begin{cases} z_{\gamma^+} & \text{on } \gamma^+ \\ 0 & \text{on } \Gamma^c \backslash \{\gamma^+\}. \end{cases} \tag{11.13}$$

We choose δ above as the smallest value which shrinks an existing element in c^* to zero, which will also be the value for the new element in c^* at index γ^+.

Using (11.12) and (11.13) we can write the following system of equations to compute the update direction ∂x

$$[G^T]_{[\Gamma^c]} \partial x = \begin{cases} z_{\gamma^+} & \text{on } \gamma^+ \\ 0 & \text{on } \Gamma^c \backslash \{\gamma^+\}, \end{cases} \tag{11.14}$$

where $[G^T]_{[\Gamma^c]}$ corresponds to the rows of G^T indexed by elements in the set Γ^c, which is same as $(G_{\Gamma^c})^T$. We solve (11.14) to compute ∂x and consequently $\partial c = G^T \partial x$. The step size associated with ∂x is δ, and as we increase δ from 0, one of the elements in $c^* + \delta \partial c$ will eventually shrink to zero. The value of this step size can be computed as

$$\delta^- = \min_{i \in \Gamma} \left(\frac{-c_i^*}{\partial c_j} \right)_+, \tag{11.15}$$

which is also the new value of $c^*_{\gamma^+}$. Let us denote γ^- as the index that leaves the support of c^*. The new estimates for the message x and the error vector c can be written as $x^* + \delta^- \partial x$ and $c^* + \delta^- \partial c$, respectively.

The support set can be updated as $\Gamma = [\Gamma \cup \gamma^+] \backslash \{\gamma^-\}$. If at some point during the primal update, an element from within Γ_n is removed, set $\Gamma_n = \Gamma_n \backslash \{\gamma^-\}$ and (to maintain the optimality of the dual solution) $\xi^*_{\gamma^-} = \epsilon \, \xi^*_{\gamma^-}$. This is equivalent to removing the homotopy parameter ϵ for that element in d and ν for which the error shrunk to zero. We repeat the alternating dual and primal updates until ϵ becomes equal to 1.

The procedure outlined above used two working assumptions. The first is that the error estimate will have exactly N zero entries until we recover the original message

x. The second is that any $N \times N$ submatrix formed by picking N rows from the $M + P \times N$ coding matrix will be nonsingular. The second assumption allows us to calculate the update directions for both the primal and dual; the first ensures that this update direction is unique. Both of these assumptions are true with probability 1 if the coding matrix is Gaussian or a random projection, and they are true with very high probability if the coding matrix A is Bernoulli [93]. In addition to this, the condition number of these submatrices will be *fairly* controlled [119]. The algorithm can be extended to properly handle situations where these assumptions do not hold, but we will not discuss this here.

The main computational cost in this algorithm also comes from solving a system of equations to compute the update directions. At every homotopy step we have one element swap in the support set Γ^c, and consequently a column g_{γ^+} in G_{Γ^c} is replaced with g_{γ^-} (a column of G at index γ^-). Note that we can represent the change in G_{Γ^c} as a rank-one update as

$$G_{\Gamma^c} \leftarrow G_{\Gamma^c} + (g_{\gamma^-} - g_{\gamma^+})\mathbf{1}_\gamma,$$

where $\mathbf{1}_\gamma$ represents a row vector with all zeros except at index γ that corresponds to the location of γ^+ in Γ^c. Instead of computing the inverse of G_{Γ^c} from scratch at every step, we can simply update the old inverse using a rank-one update [23, 66].

11.2.4 Numerical experiments

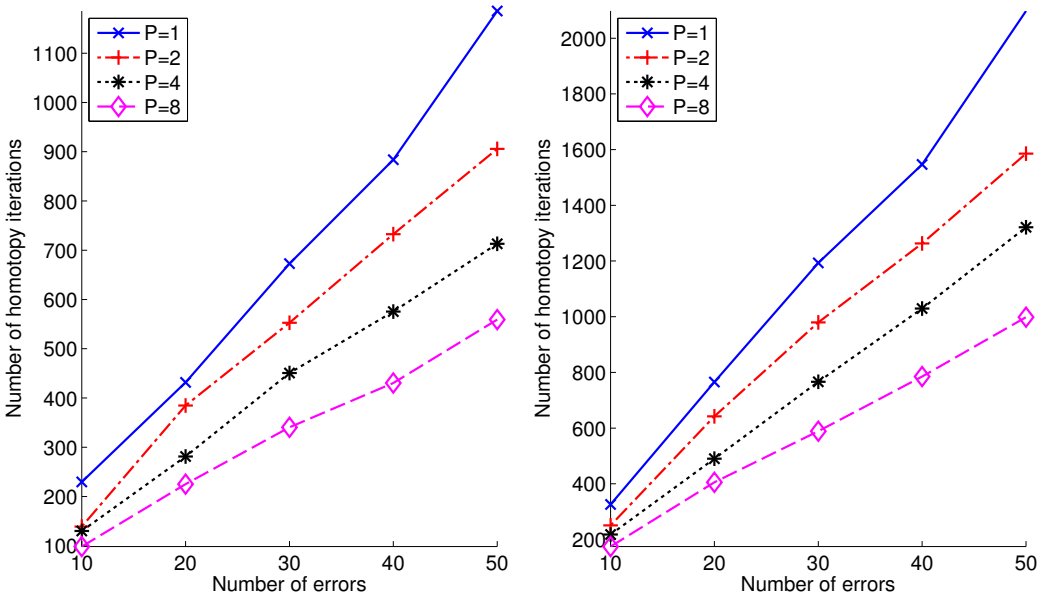
We evaluated the performance of dynamic updating for ℓ_1 decoding using the following experiment. We generated a message $\bar{x}$ of length N by selecting each entry from i.i.d. $\mathcal{N}(0, 1)$ distribution; encoded $\bar{x}$ using an $N \times N$ Gaussian matrix A, whose entries were generated from i.i.d. $\mathcal{N}(0, 1)$ distribution; and added error e by selecting S nonzero entries from i.i.d. $\mathcal{N}(0, 1)$ distribution, at random locations. We computed the initial estimate as $\widehat{x} = A^{-1}y$. Afterwards, we started adding P error-free measurements to the system and solved (11.3) using Algorithm 5 until the exact message

is recovered. Results for signal recovery for different values of N and S, averaged over 20 independent experiments, are presented in Table 11.1 and Figures 11.1 and 11.2.

Table 11.1 presents the results for the average number of measurements (length of codeword, M) required for perfect reconstruction of messages of length $N = 64,\ 128$ when $S = [10,\ 20,\ 30,\ 40,\ 50]$ sparse errors are present in the starting codeword. As it can be seen that the redundancy $(M - N)$ required for perfect recovery ranges within $2S$–$4S$. Figure 11.1 illustrates the total number of homotopy iterations that was used to recover the signal exactly. Figure 11.2 illustrates the approximate number of homotopy iterations used to update the solution every time P new measurement were added to the system. We observed that the average number of homotopy iterations required for adding a new measurement range from 3–11 for $N = 64$ and 5–15 for $N = 128$, which is a small fraction of the problem size and the number of errors in the measurements. Thus, instead of solving a new problem from scratch after receiving new measurements, we can quickly update the solution in a small number of homotopy iterations.

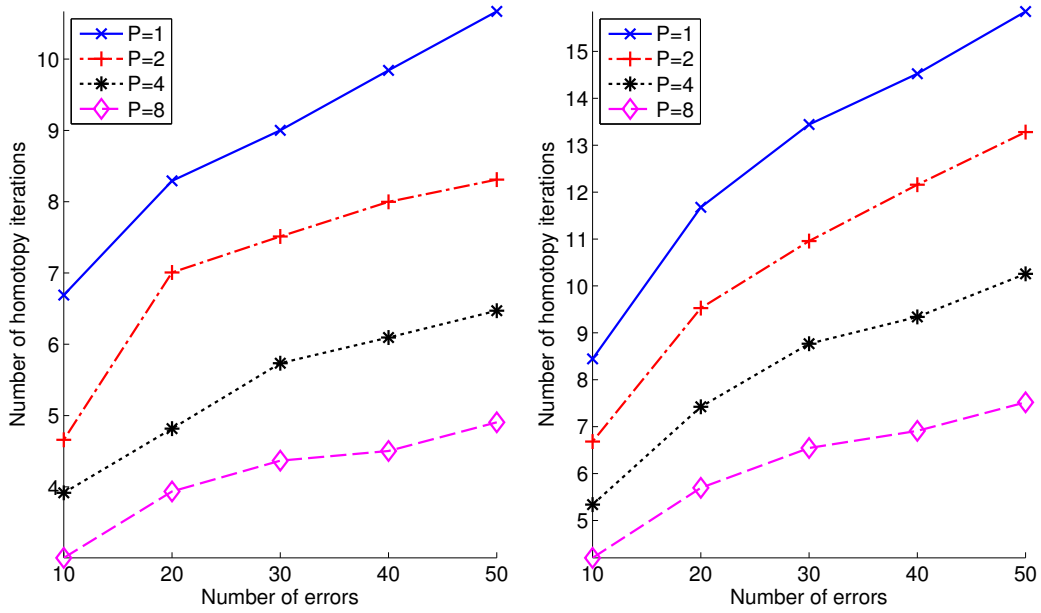
Table 11.1: Comparison of recovery performance at different levels of error: Average length of codeword (M) and redundancy $(M - N)$ required for perfect signal recovery in the presence of S sparse errors using ℓ_1 decoding.

Signal length	No. of errors (S)	Average redundancy $(M - N)$	Average codeword length (M)
$N = 64$	10	33	97
	20	56	120
	30	76	140
	40	93	157
	50	111	175
$N = 128$	10	40	168
	20	68	196
	30	89	217
	40	109	237
	50	128	256



(a) $N = 64$ **(b)** $N = 128$

Figure 11.1: Total number of homotopy iterations required for signal recovery, while adding P new measurements at a time until the exact signal is recovered.



(a) $N = 64$ **(b)** $N = 128$

Figure 11.2: Average number of homotopy iterations required to add one new measurement, while adding P new measurements at a time until the exact signal is recovered.

Algorithm 5 ℓ_1 decoding

Input: y, w, A, B and primal-dual solution pair: x^*, λ^*.
Output: x^*

Initialize: $e^* = Ax^* - y$ with support Γ_e; $d^* = Bx^* - w$ with support Γ_d; $\nu^* = z_d$; $\Gamma_n \leftarrow \Gamma_d$ and $\Gamma \leftarrow [\Gamma_e \cup \Gamma_n]$; $c^* = \begin{bmatrix} e^* \\ d^* \end{bmatrix}$; $\xi^* = \begin{bmatrix} \lambda^* \\ \nu^* \end{bmatrix}$; and $G = [A^T \; B^T]$.

Repeat:

Dual update:
Compute $\partial\xi$ in (11.9) ▷ Update direction
Compute δ^+, γ^+ and z_γ in (11.10) ▷ Step size and change in support
if $\epsilon + \delta^+ > 1$ **then**
$\delta^+ \leftarrow 1 - \epsilon$ ▷ Last iteration
$\xi^* = \xi^* + \delta^+ \partial\xi$
`break;` ▷ Quit without any further update
end if
$\xi^* \leftarrow \xi^* + \delta^+ \partial\xi$
$\epsilon \leftarrow \epsilon + \delta^+$

Primal update:
Compute ∂x in (11.14) and set $\partial c = G^T \partial x$ ▷ Update directions
Compute δ^- and γ^- in (11.15) ▷ Step size and change in support
$x^* \leftarrow x^* + \delta^- \partial x$
$c^* = c^* + \delta^- \partial c$

$\Gamma \leftarrow [\Gamma \cup \gamma^+] \backslash \{\gamma^-\}$ ▷ Swap outgoing and incoming elements
if $\gamma^- \in \Gamma_n$ **then**
$\Gamma_n \leftarrow \Gamma_n \backslash \{\gamma^-\}$ ▷ Remove homotopy on zero error locations
$\xi^*_{\gamma^-} = \epsilon \, \xi^*_{\gamma^-}$ ▷ Maintains optimality of the dual solution
if Γ_n becomes empty **then**
`break;` ▷ Lucky breakdown
end if
end if
until $\epsilon = 1$

PART II

Dynamic models in video

CHAPTER XII

LOW-COMPLEXITY VIDEO CODING

In this chapter, we discuss an application of compressive sensing in video coding, where the encoder is very simple and most of the computational complexity is shifted to the decoder. Conventional encoders compress video signals by exploiting temporal and spatial redundancies using motion estimation and transform coding blocks—the same blocks are the main source of complexity in standardized video encoders. To reduce the encoder complexity, we eliminate these blocks in our proposed video encoder. We assume that the encoder compresses a video sequence by recording a small number of non-adaptive measurements for each image (frame) in the sequence. At the decoder, we reconstruct the video sequence from the compressed measurements by exploiting the inherent spatial and temporal structure in video signals. We model the measurements and the underlying video sequence as a linear dynamical system in which adjacent frames are related to each other via (unknown) inter-frame motion. We assume that the images in the video sequence and the motion-compensated differences have sparse representations and alternately estimate images and inter-frame motion during the recovery process. We present experiments to demonstrate the performance of our recovery framework for different test sequences at different compression levels.

Conventional video coding schemes rely on the availability of fully sampled, high-resolution images for motion estimation and subsequent compression. Our focus in this chapter is on low-complexity video coding where the encoder is restricted to capture only a small number of non-adaptive measurements, and it is the task of the decoder to provide quality reconstruction by extracting as much information about the

video sequence as possible from the available measurements. The recovery framework described in this chapter can be utilized in many other situations where only a small number of non-adaptive measurements of the video sequence are available for the reconstruction. One such example is the accelerated dynamic magnetic resonance imaging, which we discuss in Chapter 13.

12.1 Background

Current video coding technology has developed assuming that a high-complexity encoder in a broadcast tower would support millions of low-complexity decoders in receiving devices. However, with the proliferation of inexpensive video recording devices, such as camcorders and mobile phones, user-generated content has become commonplace. Therefore, there is a need for low-complexity encoding technology that can be deployed in these low-cost, low-power devices [112]. Because power consumption is proportional to the encoder complexity, current high-complexity encoders consume too much power to provide high compression ratios. To increase the battery life in mobile devices, a low-complexity encoder with good coding efficiency is highly desirable. Since the decoders are usually located in mains-connected devices such as set-top boxes, television sets, and computers, their complexity and high-power consumption are tolerable. We propose an encoding and decoding scheme that leverages ideas from standard video coding and compressive sensing.

Conventional video coding schemes achieve compression by exploiting the spatial and temporal structure in the video sequence [115, 128]. At the encoder, typically we first estimate motion between adjacent frames (e.g., using block matching), and then we use transform coding (e.g., DCT or wavelets) on the inter-frame motion-compensated differences and a reference frame. Motion estimation and transform coding blocks often dominate the computational complexity of the encoder. The decoder, in contrast, is much simpler. Its only task is to use the inter-frame motion

information (transmitted by the encoder) to combine the reference frame with the motion-compensated residuals to reconstruct the video sequence.

A typical video encoder divides a video sequence into disjoint groups of T frames. Out of the T frames in each group, one frame is designated to be the I (intra-coded) frame and the rest are designated to be P (predictive) frames (or some times B frames for bi-directional prediction). The I-frame is encoded as a static image. P-frames are encoded in terms of motion-compensated residuals between the original P-frames and their respective motion-compensated predictions from the neighboring frames. Since adjacent frames in a video sequence are very similar to each other, the prediction error is usually small and allows efficient encoding. Inter-frame motion between pairs of images is typically estimated using block-matching. To predict an image A from an image B, block matching first divides A into non-overlapping blocks of equal size (e.g., 8×8 or 16×16), and then finds the closest matching block of the same size in B for every block in A. The motion-compensated predicted image A is constructed by replacing each block in A with its closest matching block from B. The relative locations of the blocks are stored in the form of motion vectors. I-frame, motion-compensated residuals, and the associated motion vectors constitute the compressed data for a group of T frames.

Compressive sensing (CS) provides a signal acquisition framework in which only a small number of (non-adaptive) linear measurements are sufficient for the recovery of a (structured) sparse signal [30]. Conventional compression schemes often require fully sampled signals at the encoder, which they then compress by retaining a small subset of data after some processing (e.g., motion estimation and transform coding in MPEG and H.264 video coding) and discarding the rest. In contrast, CS combines compression with acquisition by acquiring only as many measurements as would be necessary for the signal recovery. The number of measurements required for such recovery depends on the sparse structure of the signal (also related to the degrees

of freedom in the signal). This combined acquisition and compression reduces the burden on sensing devices in two ways: full signal acquisition and any additional compression are not required. Furthermore, the computational burden shifts from the encoder to the decoder. It is a task for the decoder to reconstruct the signal from the compressed measurements, using any available information about the signal structure. CS decoders typically recover signals by solving an optimization problem that promote sparsity in the signals while maintaining fidelity to the measurements.

To recover a video from compressed measurements, our proposed decoder exploits both spatial and temporal structures in a video signal. The inter-frame motion provides a good model for the temporal structure; however, in our compression framework, the inter-frame motion is not readily available at the decoder. Therefore, we use a two-step approach to iteratively update the estimates for the images in the video sequence and the inter-frame motion. At every iteration, we use available motion (or temporal) information to reconstruct images in the video sequence from the available measurements, and then refine the inter-frame motion estimates using the recovered images. Our two-step approach is similar to those appeared in [108, 114]; however, it is different from those in [76, 99], which assume availability of a high-quality reference frame and reconstruct motion-compensated residuals with respect to the reference frame.

12.2 Video compressive sensing

12.2.1 Compressive acquisition

Our goal is to have a simple encoder with low computational complexity. We assume that the encoder records each image in a video sequence using a small number of non-adaptive measurements. This task can be performed in an imaging hardware (e.g., a single-pixel camera [129, 143]) without explicitly capturing and storing the original images or by post-processing the images recorded with conventional CCD sensors.

Consider a video sequence $x_1, x_2, \ldots$ where each $x_i \in \mathbb{R}^N$ is an N-pixel image in the video sequence. The encoder generates a sequence of measurements as

$$y_i = \Phi_i x_i + e_i, \tag{12.1}$$

where y_i denotes measurements of x_i, Φ_i denotes the measurement matrix of size $M_i \times N$, and e_i denotes noise in the measurements. The ratio of N and M_i determines the compression rate for the i^{th} image.

One important feature that we borrow from conventional video coding in the design of our encoder is that we allow non-uniform compression for images in the video sequence. Suppose our desired compression rate allows M measurements per T frames. We allow the encoder to distribute the available measurement count to T frames in any desired fashion. Figure 12.1 depicts three possible choices for the distribution of measurements in a group of T frames: (a) Every image gets an equal number of measurements. (b) One boundary image in every group of T images gets significantly more measurements than the rest. This scheme is analogous to the I-frame and P-frames in the standard video coding. (c) An arbitrary non-uniform distribution of measurements. The motivation behind using non-uniform distribution of measurements is that a few good quality images help improve the quality of the heavily compressed neighboring images during the reconstruction. In the case of non-uniform measurements, we define the compression ratio as TN/M, where M is the total number of measurements utilized by T frames.

The quality of reconstruction depends on two factors: the number of measurements (more the better) and the type of measurements. We do not make any assumption on the specific type of measurements. A general rule for CS applications is to use the measurements that are spread out in the transform domain in which the signal of interest is sparse [32]. Some commonly used measurements for images in the CS framework include subsampled DCT, subsampled noiselets, and random convolution [117].

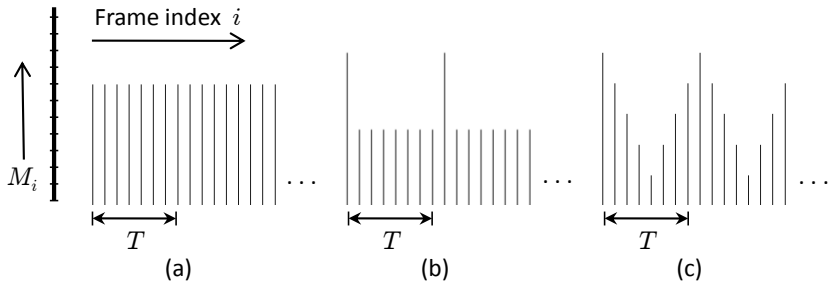


Figure 12.1: Distribution of measurements in a group of T frames. (a) Uniform distribution. (b) One boundary frame gets more measurements (analogous to I and P frames). (c) Custom distribution.

12.2.2 Motion-adaptive reconstruction

To recover the video sequence from compressive measurements in (12.1), we exploit the spatial and temporal structured sparsity in the video.

Individual images in a natural video sequence have sparse representation in a variety of spatial transforms; for example, wavelet transforms or total variation [94, 120]. Each image in the video sequence can be reconstructed independently, using one of the existing sparse recovery methods for images [117]; but such an approach ignores similarities between neighboring images in the video sequence. To exploit the temporal structure, one can assume that the inter-frame differences are sparse [97]. However, the temporal variations in video sequences are not fully captured by the inter-frame differences. Inter-frame motion provides a much superior representation for the variations in a video sequence.

To model the temporal variations, we represent video frames in the form of a linear dynamical system, where each frame is related to its immediate neighbors via inter-frame motion, as depicted in Figure 12.2. The following linear system provides a combined model for the linear measurements and the inter-frame relationship:

$$y_i = \Phi_i x_i + e_i \tag{12.2a}$$

$$x_i = F_{i-1} x_{i-1} + f_i \tag{12.2b}$$

$$x_i = B_{i+1} x_{i+1} + b_i, \tag{12.2c}$$

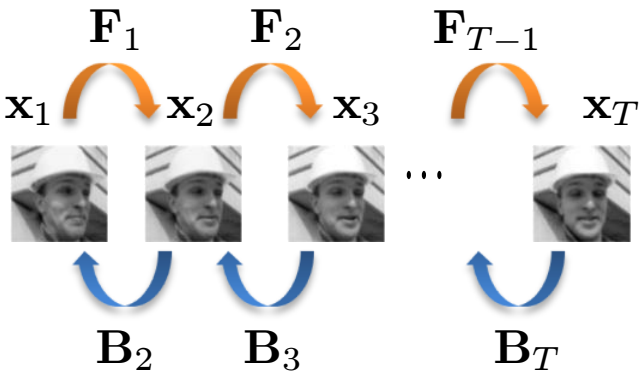


Figure 12.2: Bi-directional inter-frame motion interpretation.

where F_{i-1} and B_{i+1} denote the forward and the backward motion operators that couple x_i to its immediate neighbors x_{i-1}, x_{i+1}, and f_i and b_i denote the forward and the backward motion-compensated residuals, respectively. Motion operators can be viewed as interpolation operators that move locations of the pixels according to the inter-frame motion. The motion-compensated residual images $f_i = F_{i-1}x_{i-1} - x_i$ and $b_i = B_{i+1}x_{i+1} - x_i$ may exhibit sparsity, either in the canonical representation or after an appropriate spatial transformation. We exploit the sparsity of original images and motion-compensated residuals in the recovery process.

Note that F_{i-1} and B_{i+1} require information about motion between x_i and its immediate neighbors x_{i-1} and x_{i+1}, respectively. To estimate inter-frame motion we require images, but we only have their compressed measurements at the decoder. Therefore, we adopt an iterative approach for the reconstruction, where we alternate between estimating video frames using the available motion information, and using the estimated video frames to refine the motion estimate. Video recovery algorithms with similar alternating motion update principles have also appeared in [108, 114].

In our proposed recovery algorithm, we jointly recover a group of images from the linear dynamic system in (12.2). It helps to select the groups following the measurement distribution at the encoder. For instance, if the encoder uses the measurement

distribution in Figure 12.1(b), we divide the sequence at the decoder into overlapping groups of $T+1$ images, where the two boundary frames in each group have more measurements, and they are shared by the adjacent groups. Such a group division provides one "free" additional high-quality frame in each group, which can be beneficial with motion regularization in both forward and backward directions.

Our recovery algorithm consists of the following two-step iterative procedure: 1) Initialization. 2) Motion adaptation.

Initialization: Solve the following ℓ_1-regularized optimization problem for an initial estimate of images:

$$\text{minimize } \sum_i \tau\|x_i\|_\Psi + \lambda\|x_{i-1} - x_i\|_\Psi + \|\Phi_i x_i - y_i\|_2^2, \tag{12.3}$$

where $\|\cdot\|_\Psi$ denotes the ℓ_1 norm in the Ψ domain (i.e., $\|z\|_\Psi \stackrel{\text{def}}{=} \|\Psi^T z\|_1$); for example, wavelets or total-variation. The first term promotes sparsity in the spatial transform of each image, the second term promotes sparsity in the inter-frame difference, and the last term keeps the solution close to the measurements.

Motion adaptation: This step can be further divided into two intermediate steps, and it can be repeated multiple times to improve the reconstruction quality:

i. *Motion estimation:* Use reconstructed frames to estimate or update the inter-frame motion, and define or update the forward and backward motion operators F_i and B_i, for all i.

ii. *Motion compensation:* Solve the following optimization problem for the dynamical system in (12.2):

$$\text{minimize } \sum_i \tau\|x_i\|_\Psi + \alpha\|F_{i-1}x_{i-1} - x_i\|_\Psi + \beta\|B_{i+1}x_{i+1} - x_i\|_\Psi + \|\Phi_i x_i - y_i\|_2^2. \tag{12.4}$$

Figure 12.3: Images from video test sequences.

The regularization parameters: τ, λ, α, and β can be adapted according to the problem at hand. We found it useful to start α and β with a small value in the first iteration and increase them as the motion estimates improve. We have written all the regularization terms as a general norm $\|\cdot\|_\Psi$ to emphasize that any suitable transform Ψ can be used with the ℓ_1-norm; or if the residuals are dense, we may use the ℓ_2 norm. We do not restrict ourselves to any particular motion estimation scheme either. However, since we estimate motion from the reconstructed images instead of the original images, the motion estimation scheme should be robust to noise and other artifacts. In our experiments, we found that block matching algorithms did not perform very well. We found the phase-based motion estimation [92] and the optical flow-based schemes [86] to be more useful.

12.3 Experiments and Results

To evaluate the performance of our proposed scheme for video compressive sensing, we performed various experiments on four standard test sequences: coastguard, container, foreman, and hall[1]. Figure 12.3 presents one image from each of the four sequences. The coastguard sequence contains the most abrupt temporal variations among the four, foreman ranks second, followed by hall, and the container sequence has the least and the slowest scene variations. In all our experiments, we used 128×128 center portion of the first 129 frames of the four sequences.

[1]Video test sequences downloaded from the following webpage: http://trace.eas.asu.edu/yuv/

Our experiment setup is as follows: We divided 129 frames into 16 overlapping groups of 9 frames, where the frames at indices $9, 17, \ldots, 121$ are shared by two adjacent groups. We encoded each group according to the scheme in Figure 12.1(b), where the number of measurements for every boundary frame was twice the number of measurements for each of the remaining frames. Linear measurements for each frame consist of two parts: the first 16×16 measurements consist of the scaling coefficients of the discrete wavelet transform of the image (using Daubechies 4 filters), while the remaining measurements consist of subsampled noiselet coefficients of the image. Every group of 9 images was then reconstructed from compressed measurements using the recovery method outlined in Section 12.2.2. We used 2-D dual-tree complex wavelet transform [122] as the sparse representation basis Ψ for the spatial ℓ_1-norm regularization in (12.3) and (12.4). We estimated motion using phases of the subband coefficients from the complex dual-tree wavelets, using the hierarchical method described in [92]. We solved the optimization problems in (12.3) and (12.4) using the ℓ_1-analysis formulation in the NESTA toolbox [19].

The recovery results for the four sequences at different compression rates are presented in Figure 12.4. The performance curves plot average peak signal to noise ratio (PSNR) of all the reconstructed images at the given compression ratios. Results for our proposed motion-compensation based recovery method after 3 motion-adaptation iterations are labeled as CS-MC (solid blue line with $*$ marker). Results for the initial reconstruction with frame-difference are labeled as CS-DIFF (broken blue line with $+$ marker). Results for the frame-by-frame reconstruction (without the frame difference in (12.3)) are labeled as CS (broken blue line $\times$ marker). We also present results for videos compressed with the following two low-complexity encoders: 1) Linear DCT approximation for which the results are presented as ldct (dashed red line with $\triangledown$ marker). Linear DCT compressed an image by keeping only a small number of low-frequency 2D-DCT coefficients, selected in a predefined zigzag order. The

measurements in this compression scheme are non-adaptive and can be recorded by acquiring a predefined set of DCT coefficients for all the images. 2) JPEG compression without the entropy coding for which the results are presented as qJPEG (broken magenta line with o marker). In the qJPEG scheme, we compressed every image by thresholding its 8×8 block-DCT coefficients using the predefined quantization masks for different quality factors. The compression ratio for qJPEG at any value of quality factor was calculated as the ratio of the number of pixels in the image to the number of nonzero block-DCT coefficients retained. Although the qJPEG compression is computationally simple, it is data dependent and requires full resolution images to be available for compression.

As we can see in Figure 12.4, adding temporal regularization in the recovery process increases the PSNR of the reconstructed videos significantly (see the improvement of CS-MC (or CS-DIFF) over CS). Moreover, CS-MC outperforms qJPEG, even though qJPEG is a data-adaptive compression scheme. qJPEG performs closer to CS-MC only for the coastguard sequence, which has a complex motion, especially around frame 70.

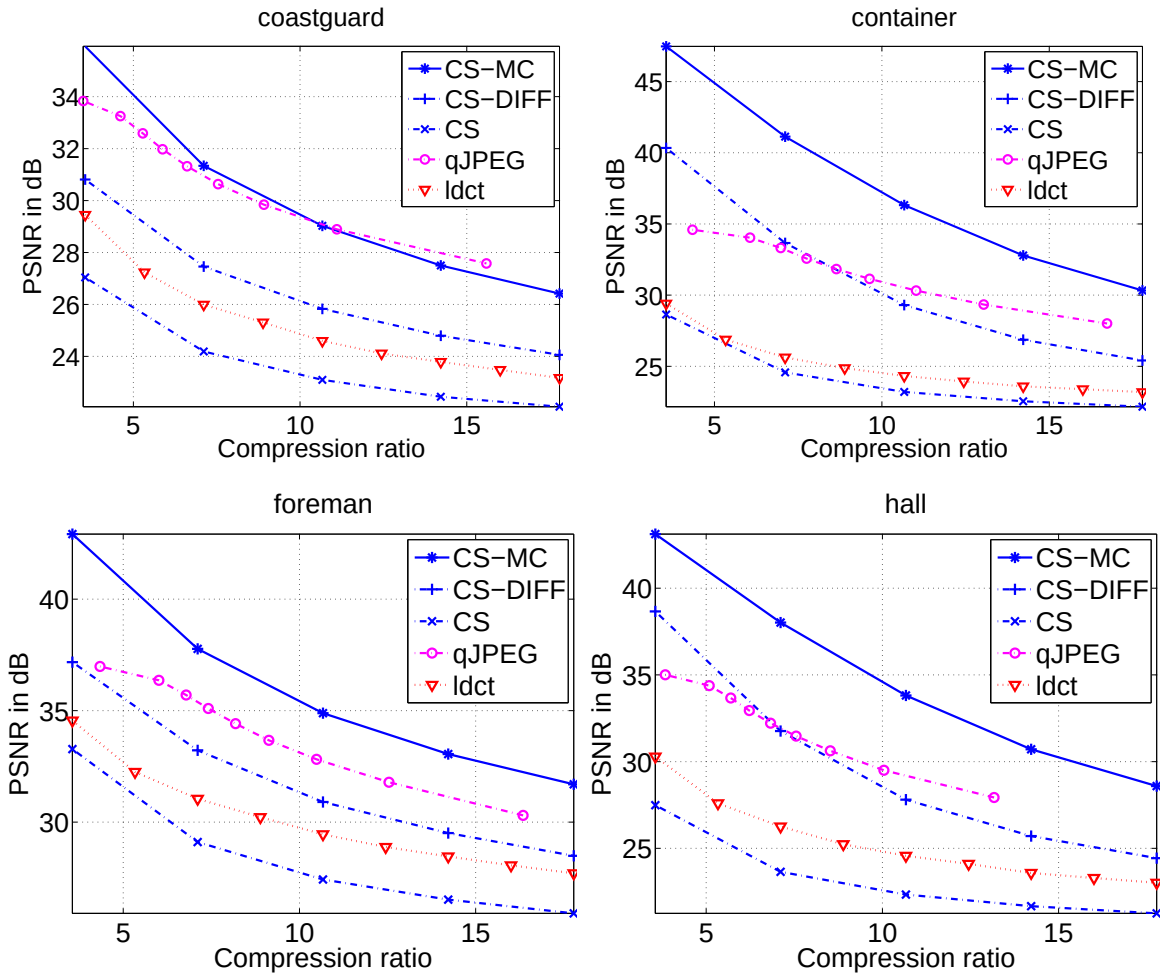


Figure 12.4: Video reconstruction from linear, non-adaptive measurements (wavelet scaling + noiselet coefficients). Motion-compensated CS (CS-MC), frame-difference (CS-DIFF), frame-by-frame (CS). Comparison with JPEG without entropy coding (qJPEG) and linear DCT (ldct) approximation.

CHAPTER XIII

ACCELERATED DYNAMIC MRI

In this chapter, we present a new recovery algorithm for highly accelerated dynamic magnetic resonance imaging (MRI) with multiple receiver coils. To recover high-resolution images from undersampled k-space data, our algorithm exploits spatial and temporal structured sparsity in MR images. To model spatial sparsity, we use wavelet transforms. To model temporal sparsity, we use inter-frame motion, which is a useful tool for representing temporal variations in video frames. In our recovery framework, we treat the underlying image sequence as a group of video frames in which an image can be predicted from its neighboring images through a motion-adaptive interpolation. The difference between an original image and its motion-adaptive interpolation is called a motion-compensated residual, which often provides a (structured) sparse image. The recovery algorithm solves an optimization problem that involves cost functions for data mismatch, spatial sparsity of each image, and temporal sparsity of motion-compensated residuals. To promote spatial and temporal structured sparsity in the solution, we use the ℓ_1-norm regularization. We call this method *motion-adaptive spatio-temporal regularization* (MASTeR).

13.1 Background

Magnetic resonance imaging (MRI) is a versatile and highly accurate imaging modality. Although MRI is used to diagnose numerous medical conditions, the current technology has several practical limitations, one of which is the slow imaging speed. Imaging speed in MRI is limited by physical and physiological constraints associated with rapidly switching magnetic field gradients. Lack of speed poses particular challenges for applications in dynamic cardiac MRI. To reduce artifacts related to the

respiratory motion, most routine cardiac MRI techniques acquire images during a patient's breath-holds. A complete acquisition takes a few cardiac cycles and may require multiple breath-holds by the patient. Thus, the patient's ability to sustain breath-holds and the imaging speed determine the duration of the examination, which in turn, determines the spatial and the temporal resolution of the images. However, since many patients are unable to sustain breath-holds, the acquisition process must be accelerated to reduce scan time.

With the introduction of reduced-data imaging methods, reduction in the scan time does not require an increase in the imaging speed (via enhanced gradient performance), but instead, a reduction in the number of phase-encoding lines that results in an undersampled k-space. A fundamental challenge in reduced-data imaging is to recover high-resolution images from undersampled k-space data—a problem that is often underdetermined. One common approach to solve such an underdetermined problem is to utilize the spatial and temporal structures in the images and the redundancies in the acquisition process.

Based on different sources of prior information, current state-of-the-art methods for dynamic MRI can be categorized as follows: parallel imaging methods, reduced field of view methods, compressed sensing-based methods, and a combination of these methods that can use multiple independent sources of prior information.

Parallel imaging methods such as SENSE [111], SPACE-RIP [82], SMASH [125], and GRAPPA [67] utilize information provided by multiple receiver coils and recover images from undersampled k-space data provided by each receiver. The main idea behind reduced field of view (rFOV) methods such as UNFOLD [91] and Noquist [28] is to exploit the spatiotemporal redundancy that is often available in dynamic imaging because parts of the field of view remain static over time. Parallel imaging and rFOV methods utilize independent prior information sources to accelerate imaging speed and they can be combined for even higher acceleration or reduced image artifacts.

Examples of methods that integrate parallel imaging with rFOV are TSENSE [80], k-t SENSE [135], and PINOT [69].

A recent addition to reduced-data imaging is compressed sensing (CS)-based methods [74, 84, 85, 88, 140]. CS theory has shown that, under certain conditions, a *sparse* signal can be recovered from a small number of linear, incoherent measurements [37, 55]. However, signal recovery from the resulting underdetermined system involves solving an optimization problem [46, 132]. An effective convex optimization program, which comes with various theoretical guarantees, minimizes the ℓ_1-norm of the sparse signal under some data-fidelity constraints [32, 38]. Sparse MRI [88] exploits spatial sparsity in MR images, such as sparsity of angiography images in the image domain and sparsity of brain or cardiac images in the wavelet or total-variation domain. CS-SENSE [84] and SparseSENSE [85] combine parallel imaging with CS. k-t SPARSE [90] and k-t FOCUSS [74] use a temporal discrete Fourier transform (DFT) for sparse representation of temporal variations in cardiac images. k-t FOCUSS can also be used with data-driven transforms such as the Karhunen-Loeve transform or the principal component analysis.

The rFOV (with static and dynamic partition) and temporal-DFT methods use *fixed* transforms along straight lines in the temporal direction without taking spatial dependencies into account. For every pixel location, these methods impose a model that does not use (nor does it provide) any information about neighboring pixels within the same frame or across different frames. However, variations in dynamic MR images arise because physical changes occur over time. For instance, a scan of a beating heart reveals different states of the heart in a cardiac cycle. Intensities of pixels overlapping the heart wall change as the heart beats (possibly in a near-periodic fashion), but at the same time, the wall of the heart changes its position from one frame to the next. These changes, correlated in both space and time, appear as a displacement of pixels in images and provide direct information about the temporal

structure in the image sequence. The displacement of pixels in a sequence of images is commonly referred to as inter-frame motion, which plays a fundamental role in modern video compression (e.g., in H.264 and MPEG standards) [128, 146]. Inter-frame motion provides an efficient and direct representation for variations as well as dependencies in a sequence of closely-related images.

k-t FOCUSS with motion estimation/motion compensation (ME/MC) [74, 75] is another recently proposed method for dynamic MRI that uses inter-frame motion during the recovery process. k-t FOCUSS with ME/MC reconstructs motion-compensated residuals of the entire image sequence with respect to a reference image by assuming that the residuals have sparse temporal-DFT. To estimate the inter-frame motion, k-t FOCUSS with ME/MC recommends using a fully sampled image as a reference frame. In the absence of a fully-sampled image, a reference image can be generated from the temporal average of k-t measurements for the entire image sequence or for only those images that correspond to the diastole phase, where the latter approach would reduce blurring in the reference image. The quality of reconstruction for k-t FOCUSS with ME/MC directly depends on the quality of the reference image and the accuracy with which it can model the true motion-compensated residuals.

The main contribution of this chapter is the use of motion-adaptive transforms that model temporal dependencies between adjacent images in both forward and backward directions. Our model can be interpreted as a linear dynamical system in which neighboring images are linked through a motion-adaptive transform that interpolates pixel values of an image to a new set of locations described by the inter-frame motion. In contrast to k-t FOCUSS with ME/MC, our proposed method, MASTeR, does not require any reference frame. Instead, all the inter-frame dependencies are embedded in the linear dynamical system formulation. As a result, it provides a practical and versatile recovery scheme that combines parallel imaging with the spatial and the temporal regularization in a unified manner. MASTeR consists of two main

steps: initialization and motion adaptation. The initialization step estimates the image sequence without any motion information. The motion-adaptation step, which can be repeated multiple times, estimates inter-frame motion in the reconstructed images and applies it to further refine image estimates. Experiments demonstrate that our method provides good reconstructed MR images from highly undersampled k-space measurements. A comparison with k-t FOCUSS with ME/MC [74] shows that MASTeR reconstructs images with a better spatio-temporal resolution, a better signal-to-noise ratio, and fewer artifacts.

13.2 Problem formulation

13.2.1 Imaging model

Consider a dynamic MRI setting in which data consist of T images in a cardiac cycle. The vector form of the imaging system for an i^{th} image can be written as

$$y_i = \Phi_i x_i + e_i, \tag{13.1}$$

where x_i is an N-length vector that denotes the underlying two-dimensional complex-valued MR image, y_i denotes the vector with k-space measurements of x_i, and e_i denotes noise in the measurements. Encoding matrix Φ_i consists of the Fourier transform coefficients weighted by coil sensitivity maps. Suppose the system has C receiver coils and at time i we receive M k-space samples from every coil at locations specified by the set Ω_i. The expanded form of Φ_i in (13.1) can be written as

$$\Phi_i \equiv \begin{bmatrix} \mathcal{F}_{\Omega_i} S_i^1 \\ \vdots \\ \mathcal{F}_{\Omega_i} S_i^C \end{bmatrix}, \tag{13.2}$$

where $S_i^1, \ldots, S_i^C$ denote the sensitivity profiles of C receiver coils and $\mathcal{F}_{\Omega_i}$ denotes an operator that computes Fourier coefficients only at locations indexed by set Ω_i.

We can write the overall dynamic MRI system as

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_T \end{bmatrix} = \begin{bmatrix} \Phi_1 & 0 & \dots & 0 \\ 0 & \Phi_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \Phi_T \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_T \end{bmatrix} + \begin{bmatrix} e_1 \\ e_2 \\ \vdots \\ e_T \end{bmatrix}$$

$$\equiv y = \Phi x + e, \tag{13.3}$$

where x denotes a vector of length TN (with all of the T images stacked on top of each other), y is a vector of length TCM (consisting of all the k-space measurements), Φ is a $TCM \times TN$ system matrix, and e denotes noise in the measurements. We call $R = N/M$ the reduction or the acceleration factor.

In accelerated MRI, increasing the acceleration factor (R) causes the system in (13.3) to become highly ill-conditioned and eventually underdetermined. To recover the underlying images from such an underdetermined system, we need additional information about the structure of the underlying images. Compressed sensing theory comprehensively addresses such problems.

13.2.2 Structure and recovery

Compressed sensing theory provides a general sensing and reconstruction framework in which a *sparse* signal can be recovered from a small number of linear, *incoherent* measurements [37, 55]. It is a well-known fact that most natural signals inherently contain redundant information that can be represented using a small number of important features [94]. For example, images can be represented using a small number of discrete cosine transform (DCT) or wavelet transform coefficients; adjacent frames in a video can be represented using a reference frame and motion-compensated residuals. In principle, CS schemes use such prior information about sparse structures to enable the recovery of signals from a small number of measurements. The recovery process involves solving an optimization problem that promotes the desired structure

in the solution while maintaining fidelity towards the measurements. An example of such an optimization program is the following ℓ_1-regularized least-squares problem:

$$\underset{x}{\text{minimize}} \ \underbrace{\|\Phi x - y\|_2^2}_{\text{data fidelity}} + \tau \underbrace{\|\Theta x\|_1}_{\text{structure}}, \tag{13.4}$$

where the ℓ_2 term keeps the solution close to the measurements, the ℓ_1 term encourages the solution to be sparse with the Θ transform, and $\tau > 0$ is the so-called regularization parameter that controls the tradeoff between the data fidelity and the signal sparsity.

CS principles can be easily applied to the dynamic MRI because MR images exhibit sparsity in a variety of spatial transforms along with significant redundancies in the temporal direction. Moreover, to satisfy the incoherence requirement of the CS system, k-space measurements for each frame can be selected at random (e.g., uniform or variable-density phase-encoded sampling) [88]. To extract the best results from the least number of measurements, a recovery algorithm must exploit both the spatial and temporal structures in MR images. Previous applications of CS in dynamic MRI have used spatial transform sparsity [84, 85, 88] as well as temporal transform sparsity [74, 90, 107] in MR images. In the rest of this section, we summarize several state-of-the-art recovery methods that use the spatial and temporal structures of MR images.

We start our discussion with the simple least squares (LS) formulation in which we will later add regularization terms that promote certain desired structures in the reconstructed signals. The LS method—one of the simplest algorithms for MR image recovery—minimizes the data mismatch without assuming any specific structure in the signal. If Φ in (13.3) has full column rank, x can be reliably estimated by solving the following LS problem:

$$\underset{x}{\text{minimize}} \ \sum_i \|\Phi_i x_i - y_i\|_2^2. \tag{13.5}$$

The problem in (13.5) can be solved separately for each x_i [66]. However, if Φ becomes ill-conditioned or underdetermined, the LS method does not provide a reliable solution [88], necessitating the use of additional information in the recovery process.

MR images exhibit spatial sparsity in a variety of transforms such as wavelets and finite differences. In the CS framework, we can easily incorporate spatial sparsity in the recovery process by adding a new regularization term to (13.5). One candidate is the following convex program:

$$\underset{x}{\text{minimize}} \sum_i \|\Phi_i x_i - y_i\|_2^2 + \tau \|\Psi x_i\|_1, \tag{13.6}$$

where Ψ denotes a sparsity inducing transform applied to each image. The program in (13.6) can be solved using a variety of fast and efficient solvers [19, 20, 26, 62, 68].

In dynamic MRI, significant gains can be achieved by exploiting the temporal structure across different images in the sequence [110, 140]. A number of recently proposed methods for cardiac MRI use Fourier transform to model sparsity in the temporal direction [74, 90, 135]. An example of a recovery program that incorporates a temporal DFT can be written as

$$\underset{x}{\text{minimize}} \|\Phi x - y\|_2^2 + \tau \|\mathcal{F}_t x\|_1, \tag{13.7}$$

where $\mathcal{F}_t x$ generates a T-point DFT along the temporal direction for every pixel location in the image sequence x. Fourier transform appears to be an attractive choice for temporal sparsity in cardiac MRI for several reasons. It is a simple, global model that does not require any additional information about the signal. It can also be viewed as a method that automatically selects static and dynamic partitions in an image sequence. For instance, a temporal DFT for any pixel in the static region will have just one nonzero component. However, temporal DFT may not be sparse for the pixels in dynamic regions, especially those with sharp amplitude variations, which may have all the DFT coefficients to be nonzero. Therefore, although temporal DFT performs very well in many cases, it does not fully exploit the temporal structure.

Since temporal variations are linked with changes in the spatial domain across different frames and dependencies among neighboring pixels, a temporal transform that adapts to the underlying spatial variations can provide a more effective model for the temporal structure. One such adaptive transform can be constructed using inter-frame motion.

13.3 Motion-adaptive spatio-temporal regularization

Inter-frame motion plays an integral role in modern video compression schemes (e.g., MPEG and H.264 codecs) [128, 146]. The fundamental insight is that inter-frame motion provides an effective way to predict neighboring frames from one another. In standard video compression, we divide a video sequence into disjoint groups of frames in which we usually designate one frame as the reference frame and the remaining ones as prediction frames. Starting with the frame after the reference frame, we estimate motion between every frame and its previous neighbor and encode the motion-compensated residuals (constructed by subtracting the original frame from its prediction) using some spatial transform. At the decoder, the reference frame and motion-compensated residuals are combined to reconstruct all the frames in the group.

In dynamic MRI, physical changes (e.g., a beating heart in cardiac imaging) govern the transformation between any two adjacent images. Inter-frame motion provides an efficient model for the underlying temporal structure. However, video compression principles are not directly applicable in dynamic MRI. First, a fully sampled reference frame is not readily available in cardiac MRI. To acquire images at a desired temporal resolution in cardiac MRI, we can observe only a small amount of k-space data (i.e., a small number of phase-encoding lines) per frame per heartbeat. Although we can generate a reference frame by combining the downsampled k-space data, a fully sampled image at the same temporal resolution cannot be acquired in an accelerated

cardiac MRI. This is because the number of cardiac cycles required for filling up the k-space of either one frame or all of the frames in the cardiac cycle is the same. Second, we do not have motion information readily available during reconstruction.

In the following section, we show that these problems can be circumvented by describing dynamic MRI in the form of a linear dynamical system in which neighboring frames predict each other using motion-adaptive transforms. The underlying inter-frame motion can be estimated and iteratively refined from available data.

13.3.1 Motion-adaptive linear dynamical system

The displacement of image features in space appears as motion across different frames. Suppose the pixel values in a small neighborhood of location (u, v) in x_i are closest to the pixel values in the neighborhood of location $(u + \Delta u, v + \Delta v)$ in x_{i-1}. The collection of $(\Delta u, \Delta v)$ for all pixels constitutes the so-called *motion vectors*. We use motion vectors to define a transform that approximates x_i from x_{i-1} as

$$x_i = F_{i-1}x_{i-1} + f_i, \tag{13.8}$$

where F_{i-1} denotes a *forward motion operator* and f_i denotes a *forward motion-compensated residual*. F_{i-1} can be considered an operator that uses motion information to interpolate the pixel values in x_{i-1} to displaced locations in x_i. Similarly, we can approximate images in the reverse direction (i.e., x_i from x_{i+1}) as

$$x_i = B_{i+1}x_{i+1} + b_i, \tag{13.9}$$

where B_{i+1} denotes a *backward motion operator* and b_i denotes a *backward motion-compensated residual*. These forward and backward motion operators construct our so-called *motion-adaptive transforms* that use inter-frame motion to represent an image sequence x in the form of forward and backward motion-compensated residuals f_i and b_i, respectively.

We combine the imaging system in (13.1) and the motion compensation equations in (13.8) and (13.9) to write the following motion-adaptive linear dynamical system:

$$y_i = \Phi_i x_i + e_i \tag{13.10a}$$

$$x_i = F_{i-1} x_{i-1} + f_i \tag{13.10b}$$

$$x_i = B_{i+1} x_{i+1} + b_i. \tag{13.10c}$$

To recover the image sequence x, we solve (13.10) by exploiting sparse structures in x_i, f_i, and b_i for all i.

13.3.2 Recovery algorithm

Now we discuss the details of our proposed recovery algorithm: motion-adaptive spatio-temporal regularization (MASTeR). MASTeR uses the dynamical system described in (13.10), in which we need inter-frame motion to define operators F and B so that we can recover the image sequence, yet we need images to compute the inter-frame motion. A common approach to mitigate such a problem is to alternately update estimates of the image sequence and inter-frame motion [108, 114]. We adopt a two-step approach in which we estimate the image sequence with any available motion information and then use the estimated image sequence to refine the motion information. A pseudocode for the recovery algorithm is as follows.

MASTeR consists of the following two-step iterative procedure:
(1) initialization and (2) motion adaptation.

---

1. **Initialization:** Solve the following spatial ℓ_1-regularization problem to recover initial image estimates from their respective k-space measurements:

$$\underset{x}{\text{minimize}} \sum_i \|\Phi_i x_i - y_i\|_2^2 + \tau \|\Psi x_i\|_1, \tag{13.6}$$

where Ψ denotes the spatial sparsifying transform.

2. **Motion adaptation:** This step can be further divided into two intermediate steps and repeated multiple times to improve the reconstruction quality.

   i. *Motion estimation:* Use the reconstructed image sequence to estimate or refine inter-frame motion and define forward and backward motion operators F_i and B_i for all i[1].

   ii. *Motion compensation:* Solve the following optimization problem following the dynamical system in (13.10):

$$\underset{x}{\text{minimize}} \sum_i \|\Phi_i x_i - y_i\|_2^2 + \alpha\|F_{i-1}x_{i-1} - x_i\|_1 + \beta\|B_{i+1}x_{i+1} - x_i\|_1. \tag{13.11}$$

A few remarks about MASTeR are in order. During the initialization step, we do not have any motion information; however, a temporal regularization can be easily added to the optimization problem (e.g., a temporal DFT or a static/dynamic partition of the field-of-view). We used ℓ_1 norms with the last two terms in (13.11) because of our assumption that motion-compensated residuals f_i and b_i are sparse in the image domain. We can easily modify these residual terms to accommodate sparsity in some transform domain, or in the case of dense residuals, we can replace the ℓ_1 norm with an ℓ_2 norm. Regularization parameters τ, α, and β can be adjusted according to the problem.

To estimate the inter-frame motion, we can use any of the existing motion estimation or optical-flow estimation schemes [17, 27, 71, 72, 105, 127]. Although our

[1]For the boundary frames, we can either couple them and treat them as neighbors (i.e., a periodic video), or we can ignore the forward motion term for the first frame (at $i = 0$) and the backward motion term for the last frame (at $i = T$).

algorithm does not depend on any particular motion estimation scheme, we must emphasize that the quality of reconstructed images directly depends on the quality of the motion estimates. Furthermore, since motion estimates come from the reconstructed images and not the original images, the motion estimation scheme should be robust against both noise and aliasing artifacts. In this regard, we found that compared to block-matching algorithms, which do not perform very well, phase-based motion estimation [92] and optical-flow methods [86] provide significantly better results.

13.4 Methods

In our experiments, we used breath-held, prospectively-gated, steady-state free precession (SSFP) cardiac MRI scans, following the protocol approved by the Institutional Review Board. We simulated accelerated imaging system by decimating fully-sampled k-space data from multiple receiver coils according to a desired sampling pattern. We used downsampled k-space measurements to reconstruct the underlying image sequence using MASTeR. We evaluated the performance of MASTeR reconstruction for two in vivo cardiac MRI scan datasets at different reduction factors. We also compared our results against that of k-t FOCUSS with ME/MC [74].

A short-axis MRI scan (images shown in Figure 13.1) was acquired using a GE 1.5T TwinSpeed scanner (R12M4) with a 5-element cardiac coil and a FIESTA-FastCARD cine SSFP sequence. Scan parameters were selected as follows: TE: 2.0 ms, TR: 4.1 ms, flip angle: 45°, FOV: 350 × 350 mm, slice thickness: 12 mm, 8 views per segment, 224 phase-encoding lines, 256 read-out samples, and 16 temporal frames. To emulate the estimation of sensitivity maps from a prescan, we acquired a separate scan (which we assumed to be a prescan) with identical scan parameters and estimated sensitivity maps as follows. Half of the (high frequency) k-space samples from each coil were removed from the prescan via a smoothing filter followed by an inverse Fourier transform to obtain smoothed images for each coil. To estimate the

sensitivity maps, we divided each smoothed coil image by the root-sum-of-squares of all coil images.

A two-chamber view cine MRI scan (images shown in Figure 13.4) was acquired using a Philips Intera 1.5T scanner (R10.3) with a 5-element cardiac synergy coil and a balanced fast field echo SSFP sequence. Scan parameters were selected as follows: TE: 2.2 ms, TR: 4.4 ms, flip angle: 45°, slice thickness: 8 mm, 240 phase-encoding lines, 200 read-out samples, and 16 temporal frames. To simulate perfectly registered sensitivity maps, we estimated them from the same data. Although this approach introduces a positive bias in the signal-to-noise ratio of measurements, it eliminates errors that may arise because of the misregistration of sensitivity profiles.

In our experiments, we primarily used a 2-D Cartesian downsampling pattern with a fully sampled low-frequency region and a randomly sampled high-frequency region. To achieve a desired reduction factor, we constructed the downsampled measurements by selecting eight low-frequency phase-encoding lines around the center of the k-space and the remaining lines at random from the high-frequency region, according to a standard Gaussian distribution. We would like to point out that although we employed the sampling pattern with dense sampling in the low-frequency region, our algorithm does not impose any such restriction on the sampling pattern.

We reconstructed MR image sequences from the downsampled k-space data using MASTeR. We used NESTA toolbox [19] to solve the ℓ_1-norm minimization problems in (13.6) and (13.11). For the initialization, we solved (13.6) using wavelet transforms as the spatial sparsifying transform Ψ for every frame in the sequence. For the subsequent motion adaptation iterations, we estimated inter-frame motion from the reconstructed images, updated motion-operators F and B, and solved (13.11). We coupled the boundary frames such that the forward motion operator for the first frame (at $i = 1$) used the last frame and the backward motion operator for the last frame (at $i = T$) used the first frame during motion estimation and compensation

steps. We performed three motion-adaptation iterations of MASTeR for the results presented in this chapter.

We used 2-D dual-tree complex-wavelet transform (DT-CWT) [122] as the spatial transform because it provided significantly better images compared with the commonly used orthogonal wavelets. We also used DT-CWT coefficient for estimating inter-frame motion, where we used the fact that a local displacement (motion) in the image domain appears as a phase shift in the CWT coefficients [92]. The DT-CWT is a redundant, nearly shift-invariant wavelet transform, which decomposes each image into directional, multiscale subbands. Each scale of the wavelet tree produces two complex-valued lowpass and six complex-valued bandpass subimages. A benefit of this redundancy is that local displacements in the image domain cause predictable changes in the wavelet coefficients. In particular, a phase shift of a complex coefficient in each bandpass subimage is approximately linearly proportional to a local displacement in the input image in a certain direction. Starting with the CWT coefficients of two images at the coarsest scale, we can estimate a local displacement vector for every subpixel in the lowpass subimage at that scale using phase shifts of respective coefficients from all the bandpass subimages. We pass the estimated displacement field to the next scale by interpolating and scaling it up by two, where each subpixel at the coarser scale would correspond to four subpixels at the next scale. At the next scale, we start with the interpolated displacement field and use phase shifts of the coefficients at that scale to further refine the displacement field. In this manner, we use a coarse-to-fine refinement strategy to produce a displacement field for each pixel. A MATLAB implementation of MASTeR, along with scripts for the DT-CWT and motion estimation, is available at http://users.ece.gatech.edu/∼sasif/dynamicMRI.

To compare the results of MASTeR and k-t FOCUSS with ME/MC [74], we implemented iterative reweighted least-squares problem for k-t FOCUSS algorithm with multiple receiver coils using the conjugate gradient (CG) method. We selected the

regularization parameters for each dataset such that the RMS error between the original and reconstructed sequence is minimized. We selected the error threshold for CG termination as 10^{-6} and allowed the CG method to run for a maximum of 200 iterations. To report the best possible results for k-t FOCUSS, we recorded the RMS error at every CG iteration and selected the CG estimate with the minimum RMS error. We generated a reference frame by taking average of six images out of 16 reconstructed images. We identified these (almost static) frames in the diastole phase by visually inspecting all the images in each dataset. We used the generated reference frame for the motion estimation and compensation steps in k-t FOCUSS with ME/MC. We performed two iterations of motion-estimation and motion-compensated residual reconstruction with three reweighting iterations each, which provided us overall good performance with k-t FOCUSS with ME/MC.

13.5 Results

13.5.1 Short Axis dataset

Figure 13.1 illustrates the comparison of MASTeR and k-t FOCUSS with ME/MC for the short axis MRI dataset at reduction factors 4 and 8. Figure 13.1(a) shows frames 1, 8, and 13 (from left to right) out of the 16 frames in the sequence, calculated from the fully sampled k-space data. The region of interest (ROI), enclosed by the white square box, is the heart region where most of the changes occur. Figure 13.1(b) presents cropped and zoomed ROI from the three frames in (a). Figures 13.1(c) and (d) present MASTeR reconstructions at reduction factors 4 and 8, respectively. The first row shows reconstructed images, and the second row shows five times amplified differences between the original and reconstructed images. The results for k-t FOCUSS with ME/MC at reduction factors 4 and 8 are presented in Figures 13.1(e) and (f), respectively.

The MASTeR reconstruction shows a significant improvement over the k-t FOCUSS with ME/MC reconstruction at both reduction factors. MASTeR reconstructions consistently contain less random noise than k-t FOCUSS with ME/MC reconstructions. More importantly, preservation of sharp myocardial edges at high reduction factors, critically important for clinical interpretation of ventricular dynamics, is clearly superior in the MASTeR reconstruction. This perhaps may be best observed in the sharply reduced residual errors practically everywhere along the endo- and epicardial borders, but most prominently visible in the frames (8 and 13) with the fastest systolic and diastolic myocardial motion. Furthermore, k-t FOCUSS with ME/MC reconstructions contain a number of aliasing artifacts (visible in bright smooth regions), while the MASTeR reconstructions are much cleaner.

In Figure 13.2, we illustrate similar observations in three temporal slices taken from selected dynamic locations in the original image sequence. Figures 13.2(c)–(f) present MASTeR and k-t FOCUSS with ME/MC results. We observe that the MASTeR reconstructions follow temporal variations very closely whereas k-t FOCUSS with ME/MC results are noisy and tend to lose fine details. White arrows in Figure 13.2(f) illustrate such regions where k-t FOCUSS with ME/MC results show artifacts and fail to follow the temporal variations accurately, whereas MASTeR remains close to the ground truth.

A quantitative comparison of MASTeR and k-t FOCUSS with ME/MC for a range of reduction factors is presented in Figure 13.3. We evaluated the performance of both methods in terms of signal-to-error ratio (SER) in dB, defined as

$$\text{SER} = 10 \log_{10} \frac{\|x\|_2^2}{\|x - \widehat{x}\|_2^2},$$

where x and $\widehat{x}$ denote the original images (constructed from full k-space data) and reconstructed images, respectively. Solid lines in Figure 13.3 denote SER over the ROI and dashed lines denote SER over the entire image. SER curves show that MASTeR outperforms k-t FOCUSS with ME/MC at all the reduction factors with

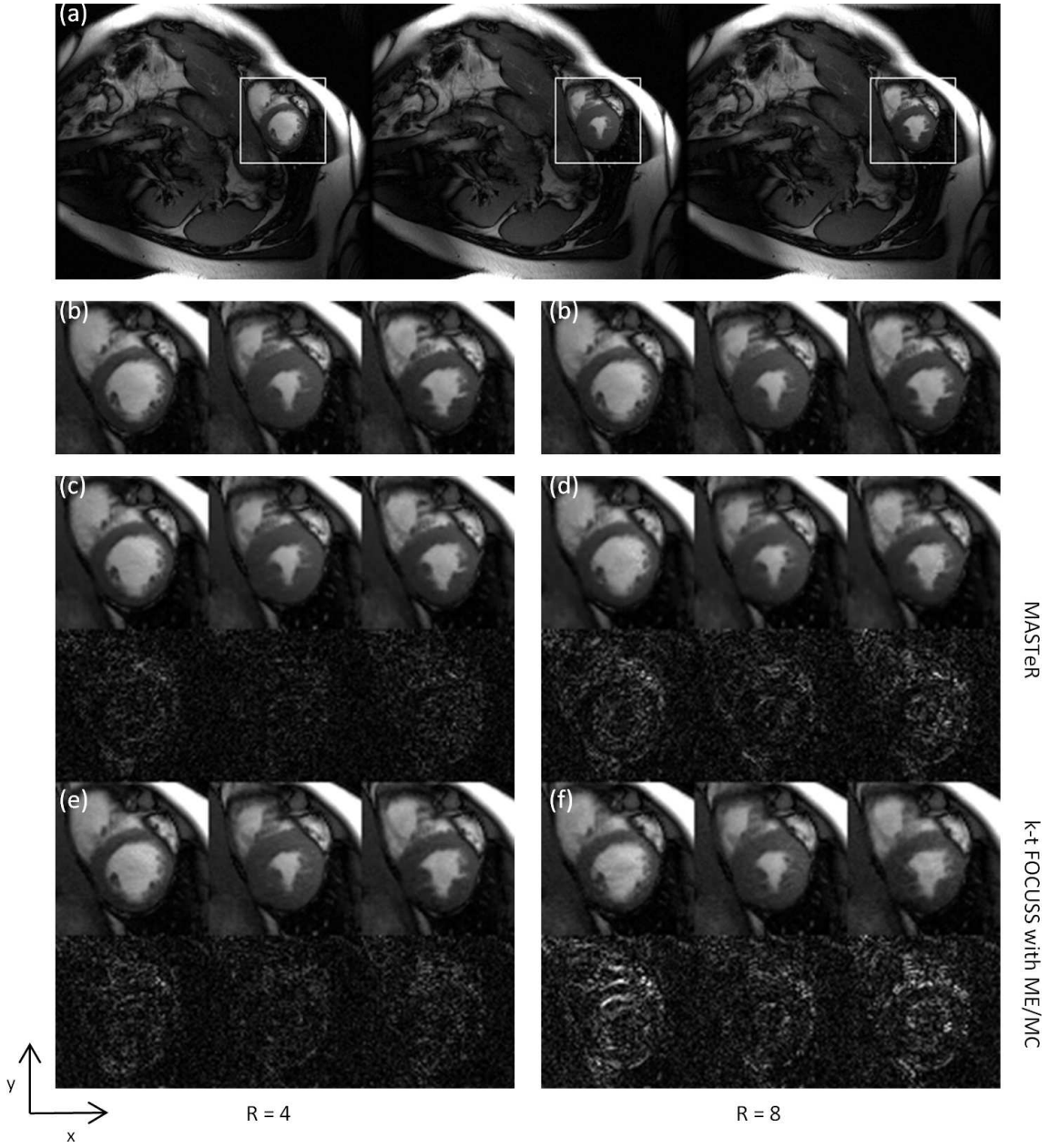


Figure 13.1: A comparison of MASTeR and k-t FOCUSS with ME/MC for the short-axis MRI scan: frames 1, 8, and 13 (left to right). (a) Conventional full-grid ground truth images from full k-space. (b) Enlarged spatial ROI. Left column: (c) MASTeR reconstruction at $R = 4$ and (e) k-t FOCUSS with ME/MC reconstruction at $R = 4$. Right column: (d) MASTeR reconstruction at $R = 8$ and (f) k-t FOCUSS with ME/MC reconstruction at $R = 8$. Bottom rows in (c)–(f) show difference images that are amplified by a factor of 5.

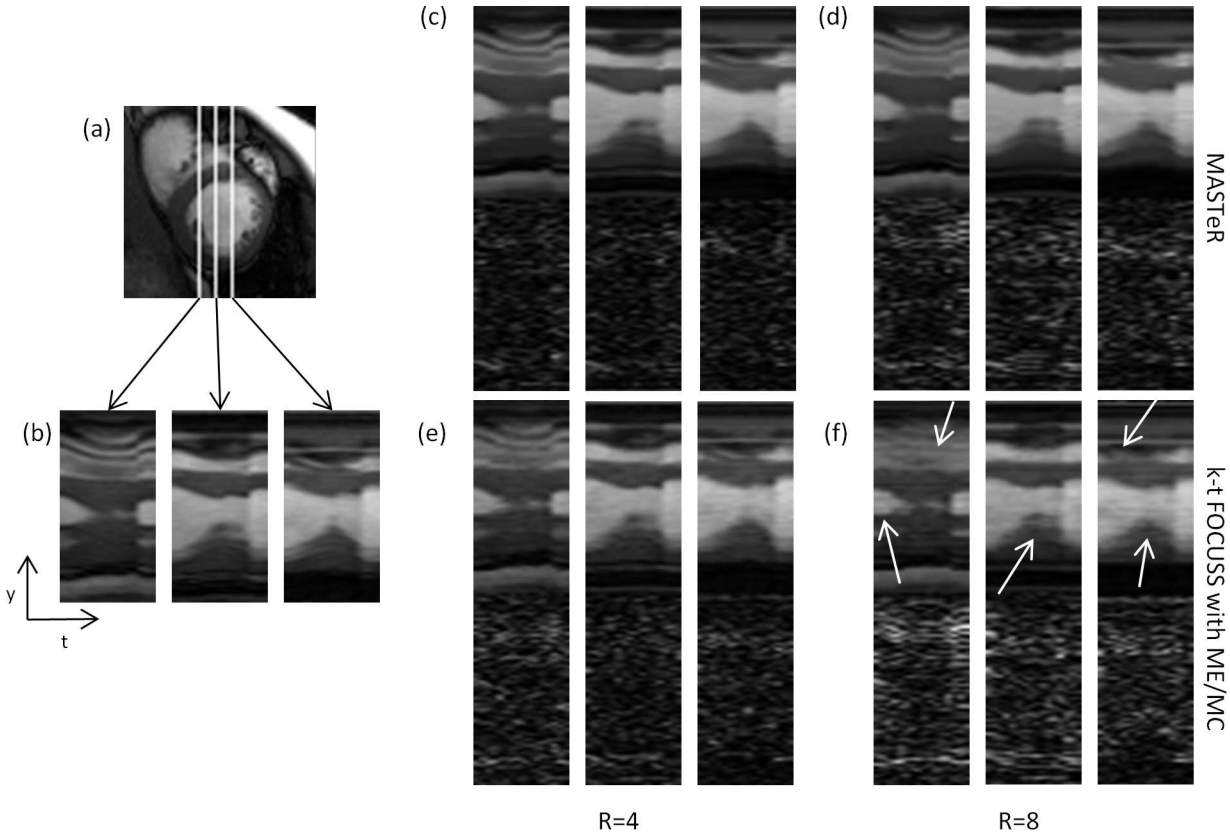


Figure 13.2: A comparison of MASTeR and k-t FOCUSS with ME/MC for the short-axis MRI scan: temporal variations. (a) ROI with lines illustrating the three locations for the temporal slices. (b) Temporal profiles in y-t space at three different locations along x direction. Left column: (c) MASTeR and (e) k-t FOCUSS with ME/MC reconstruction at $R = 4$. Right column: (d) MASTeR and (f) k-t FOCUSS with ME/MC reconstruction at $R = 8$. Bottom rows in (c)–(f) show difference images that are amplified by a factor of 5. White arrows point to regions where we see straight lines instead of smooth variations.

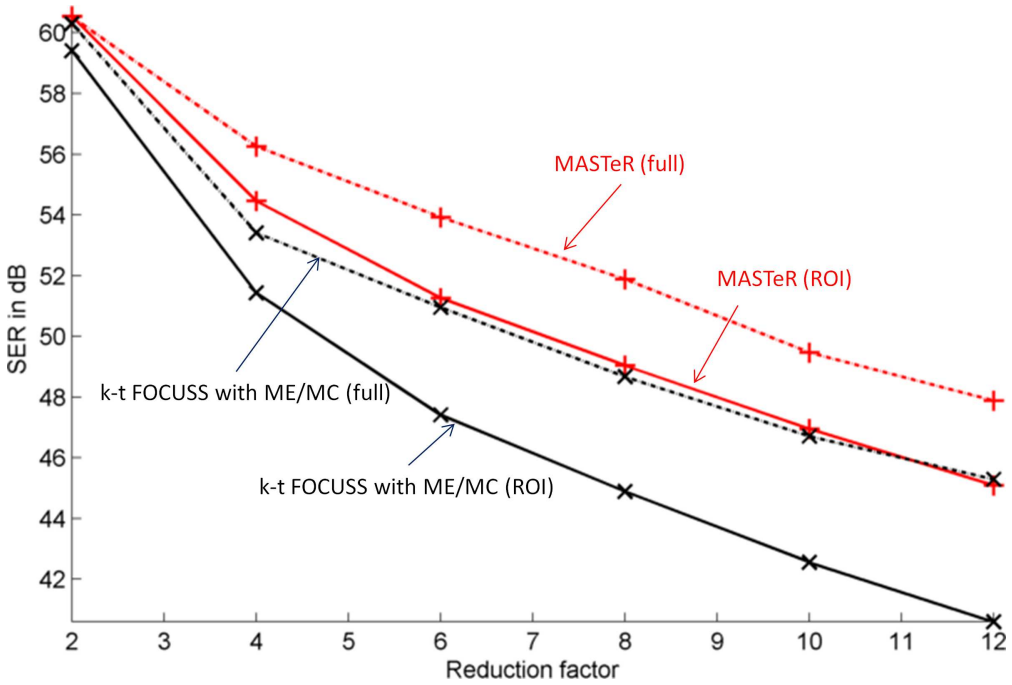


Figure 13.3: SER comparison of the MASTeR (red,+) and k-t FOCUSS with ME/MC (black,×) for the short-axis MRI dataset at different reduction factors. Solid lines represent SER in the region of interest (ROI) and dashed lines show SER over the entire image.

SER gains in the range of 4–6 dB.

13.5.2 Two-chamber results

Figure 13.4 presents a similar comparison of the reconstruction results of MASTeR and k-t FOCUSS with ME/MC for the two-chamber MRI dataset at reduction factors 6 and 10. Figure 13.4(a) presents frames 1, 3, and 9 (from top to bottom) out of total 16 frames, constructed from fully sampled k-space data. Figure 13.4(b) shows the ROI in the three frames. Figures 13.4(c) and (d) present MASTeR reconstructions at $R = 6$ and $R = 10$, and Figures 13.4(e) and (f) present k-t FOCUSS with ME/MC results at $R = 6$ and $R = 10$, respectively. At both reduction factors the MASTeR reconstructions have significantly better image quality than k-t FOCUSS with ME/MC. Even though levels of random image noise appear fairly low in all reconstructions shown from this dataset, levels of structured noise are clearly lower in

MASTeR compared with k-t FOCUSS with ME/MC. This finding is consistent with our qualitative observations and quantitative SER measurements in the short-axis scan. Moreover, delineation of the mitral valve (of clinical importance for left ventricular valve function assessment) is still adequate at reduction factor $R = 10$ with MASTeR, while the k-t FOCUSS with ME/MC reconstruction has lost most of the image details at this location and acceleration factor (white arrows in Figure 13.4(f)).

Figure 13.5 illustrates three temporal slices from the two-chamber dataset. Figures 13.5(c)–(f) present MASTeR and k-t FOCUSS with ME/MC reconstruction results. We observe a phenomenon similar to the one observed in Figure 13.2: MASTeR reconstructions follow temporal variations very closely, but k-t FOCUSS with ME/MC results tend to lose the fine details. The white arrows in Figure 13.5(f) indicate the regions where original temporal information is lost.

13.6 Discussion

13.6.1 Source of improvement in MASTeR

The fundamental difference between the motion-based temporal models of MASTeR and k-t FOCUSS with ME/MC is that instead of modeling the inter-frame motion with respect to a single reference frame and reconstructing resultant motion-compensated residuals, we modeled the temporal dependencies within neighboring frames in the form of a linear dynamical system and reconstructed the image sequence using motion-adaptive transforms. We believe that the main source of improvement for MASTeR is this difference in the temporal models. Because even if we generate a good reference frame from the available k-space data, that reference frame may fail to provide a good correspondence for all the other frames in the sequence. We observed this in our experiments on the two-chamber MRI scan where the fine details that were absent in the reference frame also vanished in the k-t FOCUSS with ME/MC reconstructions.

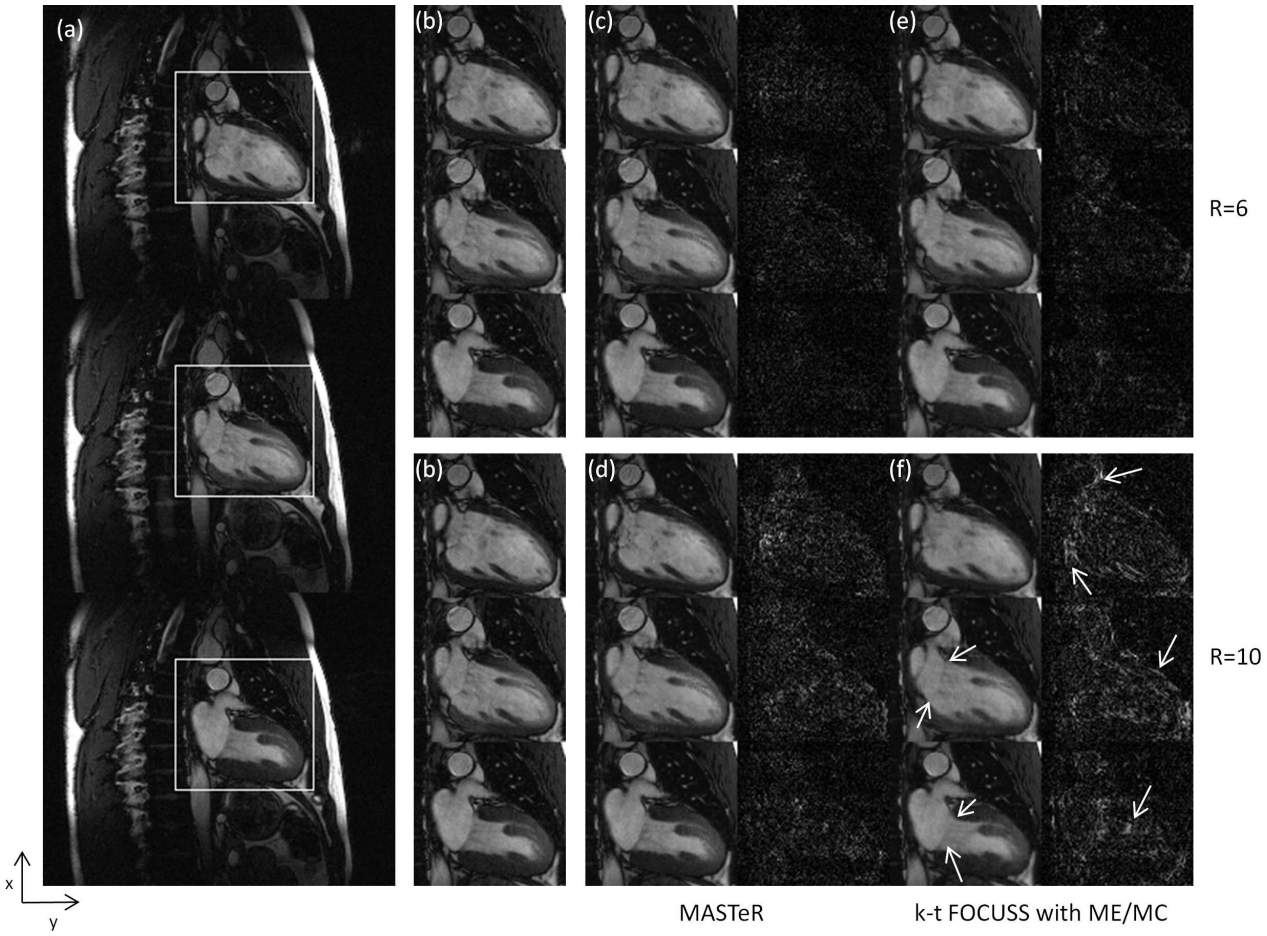


Figure 13.4: A comparison of MASTeR and k-t FOCUSS with ME/MC for the two-chamber MRI scan: frames 1, 3, and 9 (top to bottom). (a) Conventional full-grid ground truth images from the full k-space. (b) Enlarged spatial ROI. Top rows: (c) MASTeR reconstruction at $R = 6$ and (e) k-t FOCUSS with ME/MC reconstruction at $R = 6$. Bottom rows: (d) MASTeR reconstruction at $R = 10$ and (f) k-t FOCUSS with ME/MC reconstruction at $R = 10$. Right-side columns in (c)–(f) show difference images that are amplified by a factor of 5. White arrows in (f) point to the regions where heart structure is missing in the k-t FOCUSS with ME/MC reconstruction.

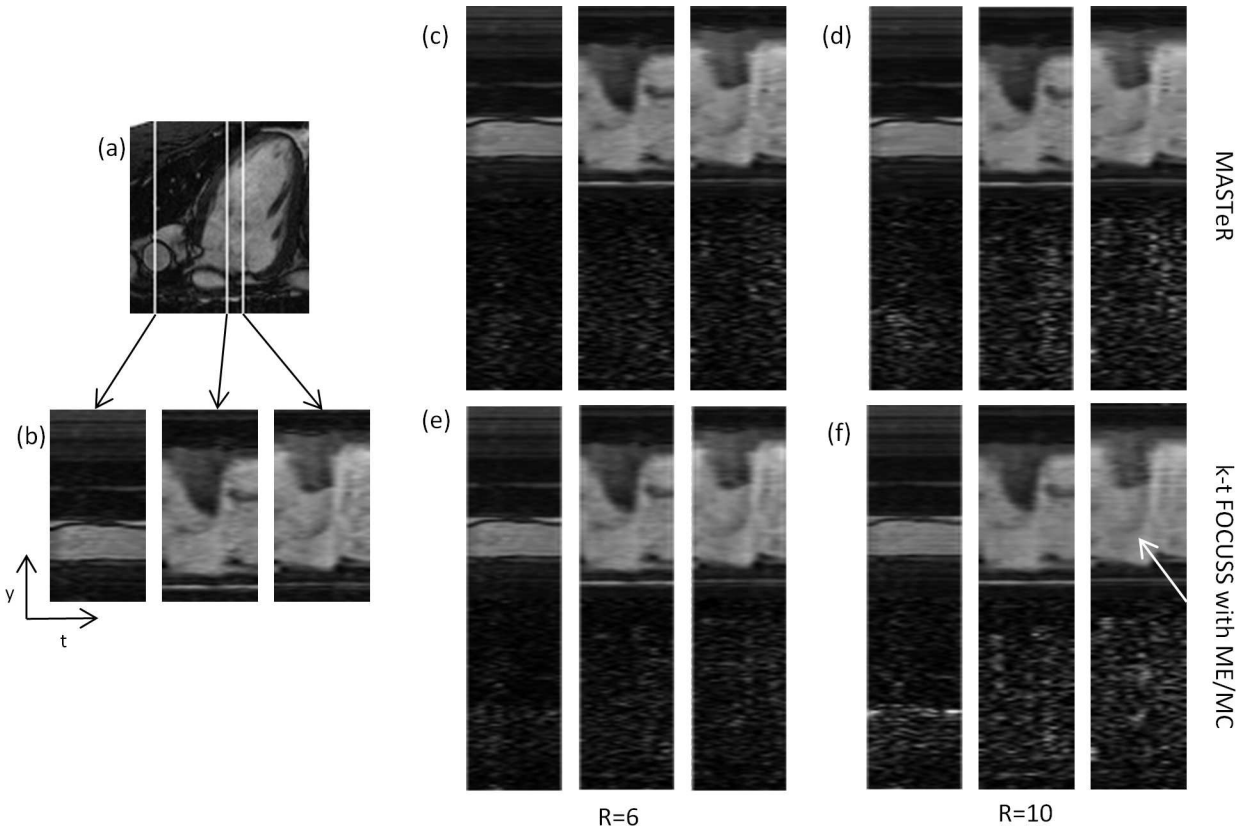


Figure 13.5: A comparison of MASTeR and k-t FOCUSS with ME/MC for the two-chamber MRI scan: temporal variations. (a) ROI with lines illustrating the three locations of the temporal slices. (b) Temporal profiles in y-t space at three different locations along x direction. Left column: (c) MASTeR and (e) k-t FOCUSS with ME/MC reconstruction at $R = 6$. Right column: (d) MASTeR and (f) k-t FOCUSS with ME/MC reconstruction at $R = 10$. Bottom rows in (c)–(f) show difference images that are amplified by a factor of 5. White arrows point to the regions where temporal information is lost in the k-t FOCUSS with ME/MC reconstruction.

We support this claim by comparing k-t FOCUSS with ME/MC and a variant of that in which we replaced the reference frame-based motion-compensated residuals terms with the motion-adaptive transforms from MASTeR. We call this variant k-t FOCUSS with MASTeR. In both the methods, the initial estimate for the image sequence is identical and is computed by solving k-t FOCUSS (i.e., a least-squares problem with iteratively reweighted temporal-DFT). For the motion-compensation step, k-t FOCUSS with ME/MC solves the following least-squares problem with iterative reweighting:

$$\underset{\Delta x}{\text{minimize}} \ \|\Phi \Delta x - y\|_2^2 + \lambda \|W \mathcal{F}_t \Delta x\|_2^2, \tag{13.12}$$

where Δx denotes motion-compensated residuals for the entire image sequence with respect to a reference frame, $\mathcal{F}_t$ denotes a temporal-DFT operator, and W denotes a diagonal matrix that is used for iterative reweighting. In contrast, k-t FOCUSS with MASTeR solves the following least-squares problem with iterative reweighting:

$$\underset{x}{\text{minimize}} \ \|\Phi x - y\|_2^2 + \lambda \|W \mathcal{M} x\|_2^2, \tag{13.13}$$

where x denotes the image sequence, W denotes a diagonal reweighting matrix, and $\mathcal{M}$ denotes a motion-compensation operator that uses the forward and the backward motion operators, F_i and B_i, to compute the respective motion-compensated differences: $f_i = F_i x_i - x_{i+1}$ and $b_i = B_i x_i - x_{i-1}$, for each image x_i in the sequence x and stacks them on top of one another. In fact, (13.13) uses iterative reweighting to approximately solve the following ℓ_1 norm problem of the MASTeR (also presented in (13.11)):

$$\underset{x}{\text{minimize}} \ \|\Phi x - y\|_2^2 + \lambda \|\mathcal{M} x\|_1. \tag{13.14}$$

We used identical procedure for optimizing regularization parameters and for iterative reweighting, while solving (13.12) and (13.13).

We followed the experimental setup described in the Methods section and compared the reconstructions for MASTeR, k-t FOCUSS with ME/MC, and k-t FOCUSS with MASTeR. The results are presented in Figure 13.6 (for short-axis scan at reduction factors 6 and 10) and Figure 13.7 (for two-chamber scan at reduction factors 10 and 12). The results show that, under identical settings of recovery framework, the motion-adaptive model outperforms the reference frame-based residual reconstruction. The results for k-t FOCUSS with MASTeR are distinctly better than those of k-t FOCUSS with ME/MC; they are less noisy and preserve the fine details in the reconstructions that are lost in k-t FOCUSS with ME/MC. The results for MASTeR are still superior to those of k-t FOCUSS with MASTeR, which can be due to the use of ℓ_1 norm instead of iteratively reweighted ℓ_2 norm and the use of spatial regularization instead of temporal regularization for the initialization.

13.6.2 Comparison of motion estimation schemes

Motion estimation is a central component of our method, and we desire a robust motion estimation scheme that can perform well with noisy, distorted images. Although we can use any motion estimation scheme in MASTeR framework, we found that CWT phase-based motion estimation performed significantly better than block-matching and overlapped block-matching schemes in our experiments.

In Figure 13.8 we present an experiment where we reconstructed a short-axis MRI scan at reduction factor $R = 8$ using MASTeR. In the initialization step, we estimated each image by solving (13.6) using DT-CWT as the sparse spatial transform Ψ. The second column in Figure 13.8 presents the ROI of the reconstructed images, which are blurry and contain several artifacts. We used three different schemes for estimating motion: phase shifts of CWT coefficients (in the third column), overlapped block-matching (in the fourth column), and standard block-matching (in the last column). The results clearly show that the CWT-based motion estimation scheme is more

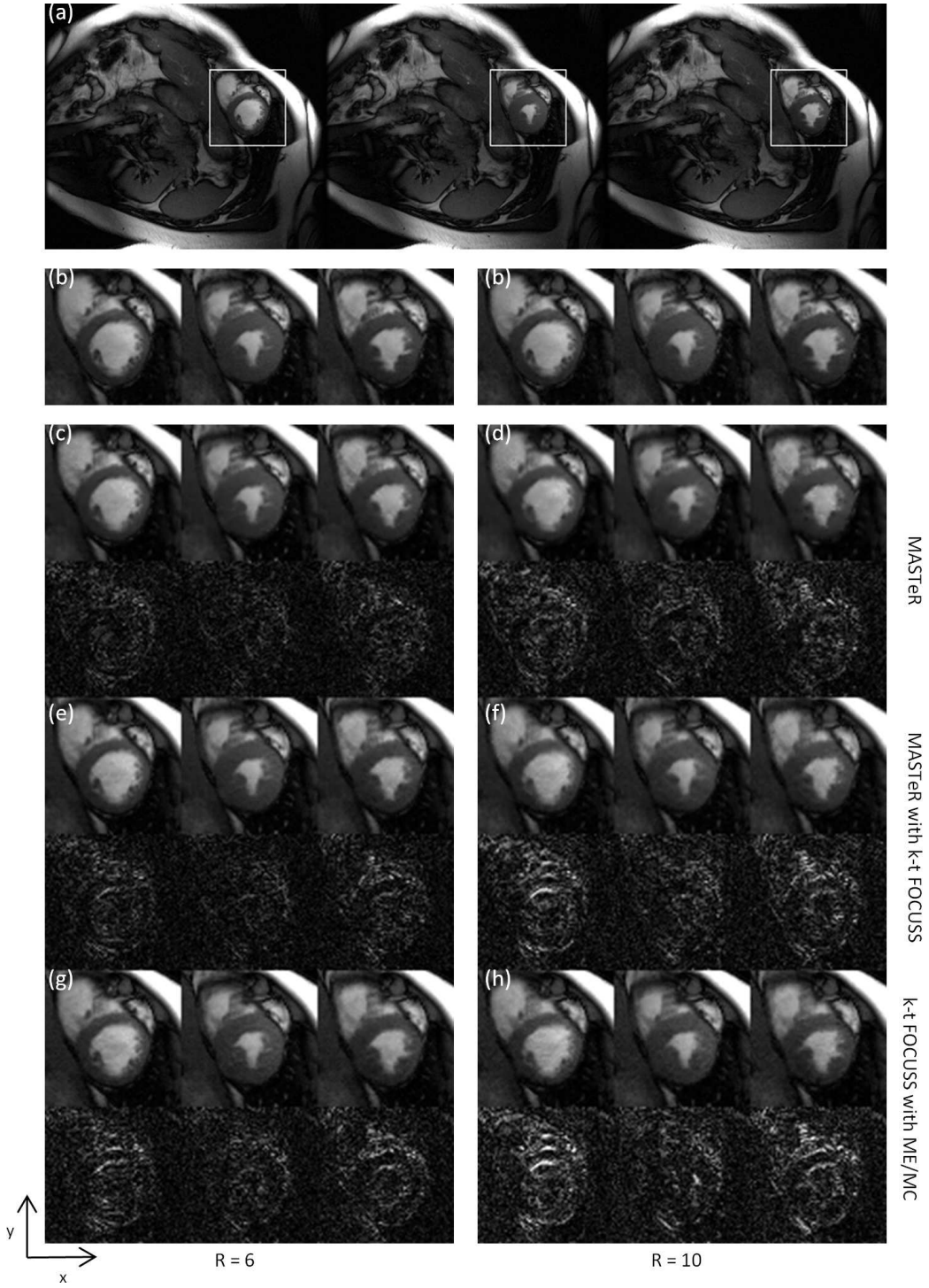


Figure 13.6: A comparison of MASTeR, MASTeR with k-t FOCUSS, and k-t FOCUSS with ME/MC for the short-axis MRI scan: frames 1, 8, and 13 (left to right). (a) Conventional full-grid ground truth images from the full k-space. (b) Enlarged spatial ROI. MASTeR reconstruction at $R = 6$ in (c) and $R = 10$ in (d). k-t FOCUSS with MASTeR at $R = 6$ in (e) and $R = 10$ in (f). k-t FOCUSS with ME/MC reconstruction at $R = 6$ in (g) and $R = 10$ in (h). Bottom rows in (c)–(h) show difference images that are amplified by a factor of 5.

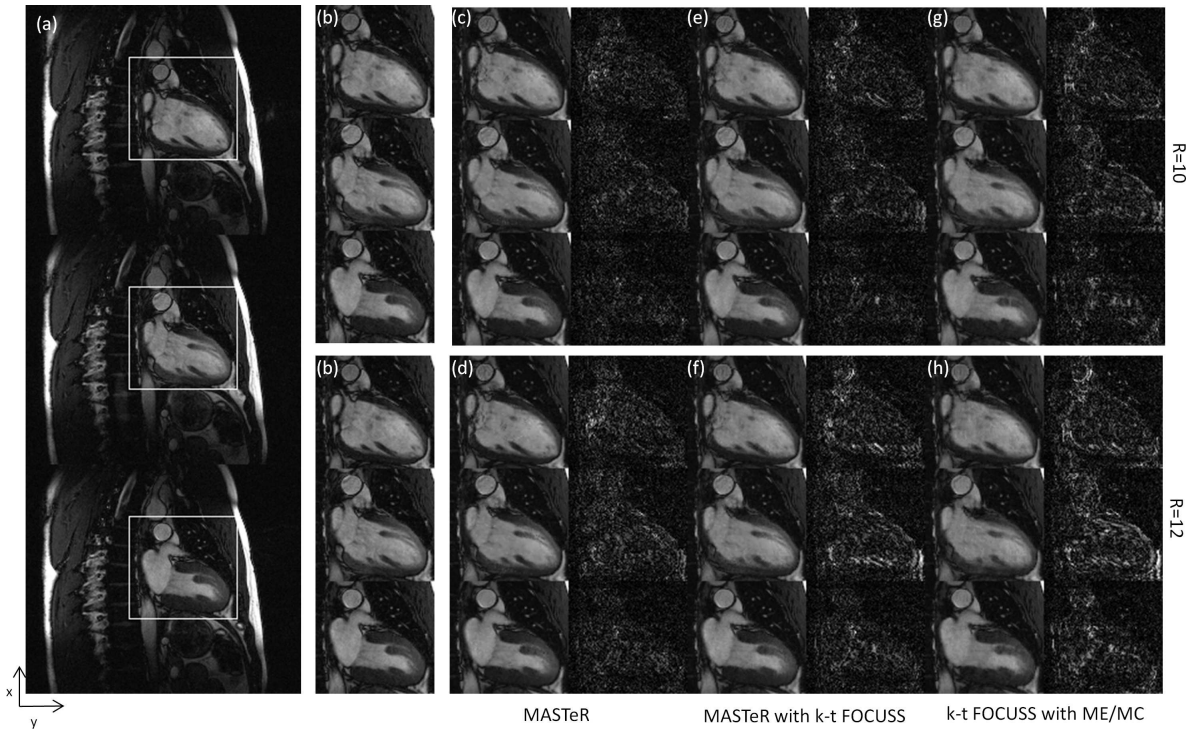


Figure 13.7: A comparison of MASTeR, MASTeR with k-t FOCUSS, and k-t FOCUSS with ME/MC for the two-chamber MRI scan: frames 1, 3, and 9 (top to bottom). (a) Conventional full-grid ground truth images from the full k-space. (b) Enlarged spatial ROI. MASTeR reconstruction at $R = 10$ in (c) and $R = 12$ in (d). k-t FOCUSS with MASTeR at $R = 10$ in (e) and $R = 12$ in (f). k-t FOCUSS with ME/MC reconstruction at $R = 10$ in (g) and $R = 12$ in (h). Right-side columns in (c)–(h) show difference images that are amplified by a factor of 5.

robust than the block matching and the overlapped block matching schemes in this setting.

We would like to point out that if we add temporal regularization (e.g., temporal-DFT) in the initialization step, the quality of initial reconstruction improves and OBM-based motion estimation scheme works well. However, we have presented these result to justify the reason for using CWT-based motion estimation scheme in our experiments, in which block-matching based schemes did not perform very well.

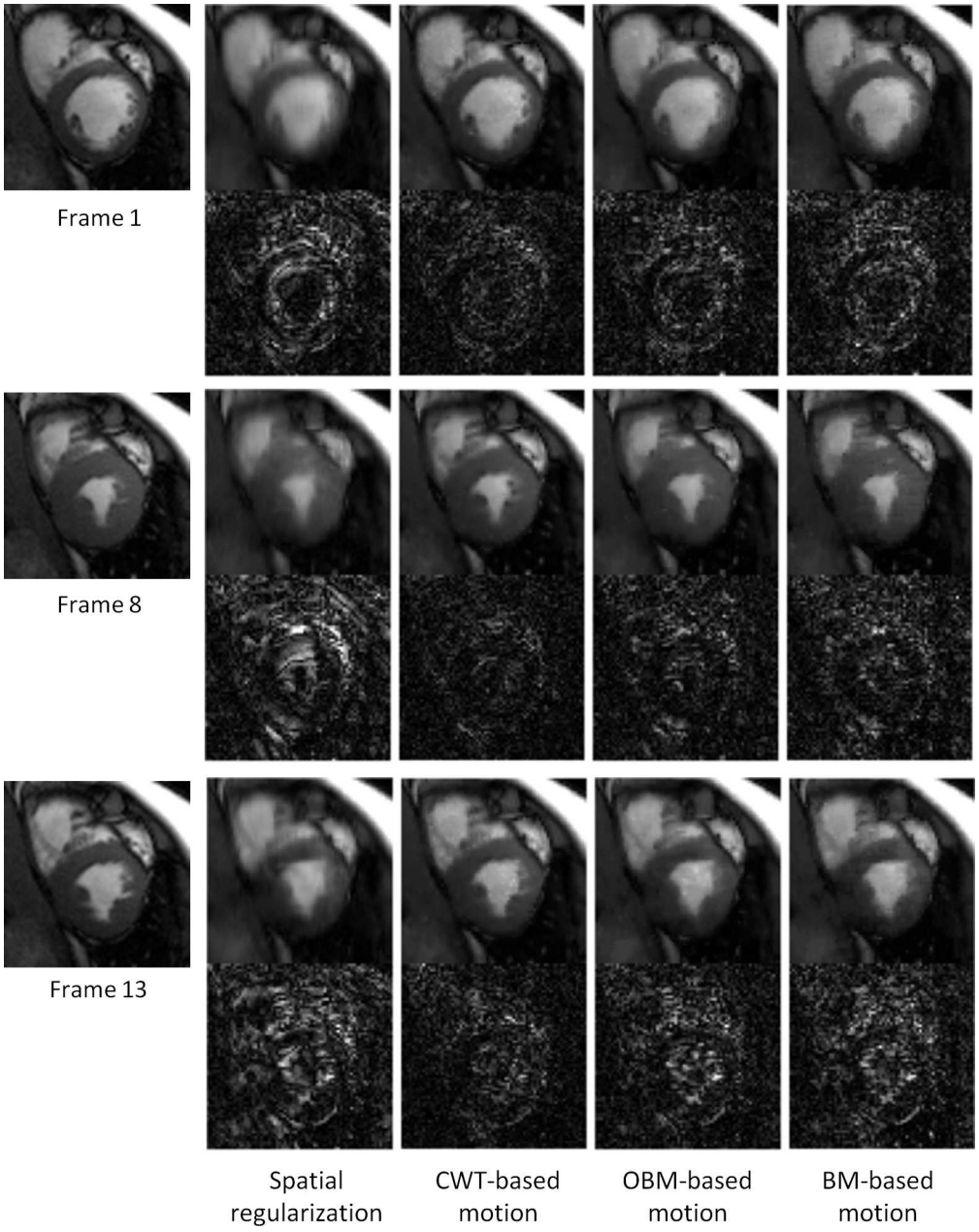


Figure 13.8: A comparison of different motion estimation schemes in MASTeR. Column 1: ROIs from frames 1, 8, and 13 of the short-axis MRI scan. The images are reconstructed from k-space data at $R = 8$. Column 2: Initial results for frame-by-frame reconstruction using ℓ_1 regularization with DT-CWT. Column 3–5: Results after thee motion adaptation iterations when we estimated motion using CWT-based method, overlapped block-matching (OBM)-based method, or standard block-matching (BM)-based method, respectively. Bottom rows in each plot show difference images that are amplified by a factor of 5.

CHAPTER XIV

CONCLUSION

The work presented in this thesis builds on the principles of sparse signal recovery from linear, non-adaptive measurements. A number of efficient algorithms and recovery guarantees have been proposed in optimization and compressive sensing literature for such problems. However, most of these results assume a static model for the signal and the measurement system. The main emphasis in this thesis is on the algorithms that can efficiently use dynamics in the signal and the measurements for solving the sparse recovery problems. In *dynamic ℓ_1 updating*, our goal is to use any available knowledge about the signal estimate and the variations in the signal or the system in a systematic manner so that the sparse recovery process can be accelerated. In building the *dynamic model for video*, our goal is to utilize knowledge about the spatial and the temporal structure inherent in a video sequence so that a high-quality video can be reconstructed from the available compressed, non-adaptive measurements.

14.1 Dynamic ℓ_1 updating

We presented a suite of homotopy algorithms that can quickly update solutions for various ℓ_1-norm minimization programs. The homotopy methods we discussed are simple and inexpensive, and promise significantly lower marginal cost than solving entirely new optimization programs from scratch. These methods break the update procedure into a series of linear steps, and the computational cost of each step is a few matrix-vector multiplications. We presented a general homotopy algorithm in Chapter 3 that caters to a variety of dynamical settings; for instance, time-varying signals, streaming measurements, iterative reweighting in ℓ_1 norm, and dynamic signal models. In Chapters 4–11, we discussed dynamic updating for different ℓ_1 programs

that usually arise in streaming and dynamical settings, and for which the homotopy update can substantially expedite the recovery process. We presented experimental evidence to demonstrate that our proposed homotopy algorithms perform significantly better than other state-of-the-art solvers.

14.2 Dynamic models in video

Low-complexity video coding: We presented a video coding framework that is inspired by compressive sensing principles, where instead of sampling the video scene at full resolution, we capture a small number of indirect, non-adaptive measurements. Such an encoder is particularly desired when the video recording device is constrained and can only provide incomplete measurements. In that case, conventional video coding tasks such as motion-estimation and residual compression cannot be performed at the encoder. The constraints can be for various reasons; for instance, limitations of the sensors, excessive power consumption, or prolonged imaging time. In such a case, the burden of reconstruction is shifted to the decoder, which needs to extract as much information about the video sequence as possible and provide its quality reconstruction from the available measurements. To recover the video from the compressed measurements, we used a motion-adaptive dynamical system to describe measurements and temporal variations in the video; where we used inter-frame motion to couple neighboring frames in the video sequence. Since motion information is not readily available from compressed measurements of the video, we adopt an alternating minimization approach in which we iteratively estimate images using any available motion information and then use estimated images to refine estimates of inter-frame motion. We demonstrated with an extensive set of experiments on standard test sequences that using inter-frame motion in the reconstruction provides significantly better results compared to either frame-by-frame or frame-difference based reconstruction.

Accelerated dynamic MRI: We presented a new recovery algorithm (MASTeR) for highly accelerated dynamic MRI, in which the acquisition process is accelerated by undersampling the k-space data (i.e., 2-D Fourier coefficients). MASTeR, which uses motion-adaptive transforms to model the temporal sparsity in images, was demonstrated to successfully recover cardiac MR images for a range of undersampling factors. It provided images with consistently superior spatial and temporal resolutions and signal-to-error ratios compared to k-t FOCUSS with ME/MC, which is another popular recovery method. The source of improvement in MASTeR over existing methods is the motion-adaptive temporal model, where instead of modeling the inter-frame motion with respect to a single reference frame and reconstructing resultant motion-compensated residuals, we reconstructed images using a linear dynamical system that employs motion-adaptive transforms for modeling temporal dependencies between adjacent frames in forward and backward directions. Our results show that it is feasible to recover high-quality dynamic cardiac MR images with an acceleration factor up to $R = 10$ using MASTeR.

14.3 Future directions

To conclude, we discuss some possible future directions for this research.

Dynamic updating: Homotopy methods for ℓ_1 problems are extremely fast and accurate, but they are limited to small-to-medium scale problems. One possible future work is to extend homotopy algorithms for general structured-signal recovery problems and develop online algorithms for fast updating in large-scale problems. Furthermore, a theoretical analysis of performance guarantees for the ℓ_1-homotopy algorithms is also highly desirable.

Adaptive sensing for videos: In our work on the recovery of videos from compressed measurements, we have primarily used non-adaptive measurements to acquire each image. However, since images in a video sequence are highly correlated, we can potentially infer the contents of an image before acquisition by either extrapolating previously reconstructed images or using guidance from other sources (e.g., a map or geometry of the scene that identifies regions of interest). A future work might consider an intelligent sensing mechanism in which prior information is used to design measurements for each image. We expect that such an adaptive sensing scheme can provide an improved *coded* video acquisition and reconstruction framework.

Medical imaging: We used motion-adaptive model in dynamic MRI to establish a dynamic model between adjacent images in the MRI sequence. Other problems in medical imaging can also benefit from similar models. One such example is the ultrasound imaging, where an array of transducers record reflections from a local region in the body. Since relative locations of the transducers with respect to the region under observation is known, a signal model that uses correlation between the received signals can explain the phenomenon in a better way. This can potentially improve the quality of reconstruction or help reduce the complexity of the transducer.

REFERENCES

[1] Afonso, M., Bioucas-Dias, J., and Figueiredo, M., "Fast image recovery using variable splitting and constrained optimization," *IEEE Transactions on Image Processing*, vol. 19, pp. 2345–2356, Sept. 2010.

[2] Angelosante, D., Roumeliotis, S. I., and Giannakis, G. B., "Lasso-Kalman smoother for tracking sparse signals," in *Proc. 43rd Asilomar Conference on Signals, Systems and Computers*, pp. 181–185, Nov. 2009.

[3] Asif, M. S. and Romberg, J., "On the LASSO and Dantzig selector equivalence," in *44th Annual Conference on Information Sciences and Systems*, pp. 1–6, Mar. 2010.

[4] Asif, M. S., Charles, A., Romberg, J., and Rozell, C., "Estimation and dynamic updating of time-varying signals with sparse variations," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 3908–3911, May 2011.

[5] Asif, M. S., Hamilton, L., Brummer, M., and Romberg, J., "Motion-adaptive spatio-temporal regularization for accelerated dynamic MRI," *Magnetic Resonance in Medicine*, 2012. DOI: 10.1002/mrm.24524.

[6] Asif, M. S. and Romberg, J., "Streaming measurements in compressive sensing: ℓ_1 filtering," in *42nd Asilomar conference on Signals, Systems and Computers*, pp. 1051–1058, Oct. 2008.

[7] Asif, M. S. and Romberg, J., "Dantzig selector homotopy with dynamic measurements," in *Proc. IS&T/ SPIE Computational Imaging VII*, vol. 7246, 2009.

[8] Asif, M. S. and Romberg, J., "Dynamic updating for sparse time varying signals," in *43rd Annual Conference on Information Sciences and Systems (CISS)*, pp. 3–8, Mar. 2009.

[9] Asif, M. S. and Romberg, J., "Sparse signal recovery and dynamic update of the underdetermined system," in *44th Asilomar Conference on Signals, Systems and Computers*, pp. 798–802, Nov. 2010.

[10] Asif, M. S., "Primal Dual Pursuit: A homotopy based algorithm for the Dantzig selector," Master's thesis, Georgia Institute of Technology, Aug. 2008.

[11] Asif, M. S. and Romberg, J., "ℓ_1 Homotopy : A MATLAB toolbox for homotopy algorithms in ℓ_1 norm minimization problems." http://users.ece.gatech.edu/∼sasif/homotopy.

[12] Asif, M. S. and Romberg, J., "Fast and accurate algorithms for re-weighted ℓ_1-norm minimization," [Preprint] Available: http://arxiv.org/abs/1208.0651.

[13] Asif, M. S. and Romberg, J., "Sparse recovery of streaming signals using ℓ_1-homotopy," [Preprint] Available: http://arxiv.org/abs/1306.3331.

[14] Asif, M. S. and Romberg, J., "Dynamic updating for ℓ_1 minimization," *IEEE Journal of Selected Topics in Signal Processing*, vol. 4, pp. 421–434, Apr. 2010.

[15] Baraniuk, R., Davenport, M., DeVore, R., and Wakin, M., "A simple proof of the restricted isometry property for random matrices," *Constructive Approximation*, vol. 28, no. 3, pp. 253–263, 2008.

[16] Baraniuk, R., Cevher, V., Duarte, M., and Hegde, C., "Model-based compressive sensing," *IEEE Transactions on Information Theory*, vol. 56, pp. 1982 –2001, Apr. 2010.

[17] Barron, J. L., Fleet, D. J., and Beauchemin, S. S., "Performance of optical flow techniques," *International Journal of Computer Vision*, vol. 12, pp. 43–77, Feb. 1994.

[18] Beck, A. and Teboulle, M., "A fast iterative shrinkage-thresholding algorithm for linear inverse problems," *SIAM Journal on Imaging Sciences*, vol. 2, no. 1, pp. 183–202, 2009.

[19] Becker, S., Bobin, J., and Candès., E., "NESTA: A fast and accurate first-order method for sparse recovery," *SIAM Journal on Imaging Sciences*, vol. 4, no. 1, pp. 1–39, 2011.

[20] Becker, S., Candès, E., and Grant, M., "Templates for convex cone problems with applications to sparse signal recovery," *Mathematical Programming Computation*, vol. 3, no. 3, 2011.

[21] Bertsekas, D., *Nonlinear programming.* Athena Scientific Belmont, Mass, 1999.

[22] Bioucas-Dias, J. and Figueiredo, M., "A new TwIST: two-step iterative shrinkage/thresholding algorithms for image restoration," *IEEE Transactions on Image Processing*, vol. 16, pp. 2992–3004, Dec. 2007.

[23] Björck, Å., *Numerical Methods for Least Squares Problems.* Society for Industrial and Applied Mathematics (SIAM), 1996.

[24] Blu, T., Dragotti, P.-L., Vetterli, M., Marziliano, P., and Coulot, L., "Sparse Sampling of Signal Innovations [Theory, algorithms, and performance bounds]," *IEEE Signal Processing Magazine*, vol. 25, pp. 31–40, Mar. 2008.

[25] Blumensath, T. and Davies, M., "Iterative thresholding for sparse approximations," *Journal of Fourier Analysis and Applications*, vol. 14, no. 5, pp. 629–654, 2008.

[26] Boyd, S. and Vandenberghe, L., *Convex Optimization*. Cambridge University Press, March 2004.

[27] Brox, T., Bruhn, A., Papenberg, N., and Weickert, J., "High accuracy optical flow estimation based on a theory for warping," in *European conference on computer vision (ECCV), Prague*, p. 25–36, 2004.

[28] Brummer, M. E., Moratal-Pérez, D., Hong, C., Pettigrew, R. I., Millet-Roig, J., and Dixon, W. T., "Noquist: Reduced field-of-view imaging by direct fourier inversion," *Magnetic Resonance in Medicine*, vol. 51, pp. 331–342, Feb. 2004.

[29] Buckheit, J., Chen, S., Donoho, D., and Johnstone, I., "Wavelab 850, Software toolbox." http://www-stat.stanford.edu/∼wavelab/.

[30] Candès, E., "Compressive sampling," *Proceedings of the International Congress of Mathematicians, Madrid, Spain*, vol. 3, pp. 1433–1452, 2006.

[31] Candès, E. and Donoho, D., "Ridgelets: a key to higher-dimensional intermittency?," *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 357, no. 1760, pp. 2495–2509, 1999.

[32] Candès, E. and Romberg, J., "Sparsity and incoherence in compressive sampling," *Inverse Problems*, vol. 23, no. 3, pp. 969–985, 2007.

[33] Candès, E., Romberg, J., and Tao, T., "Stable signal recovery from incomplete and inaccurate measurements," *Communications on Pure and Applied Mathematics*, vol. 59, no. 8, pp. 1207–1223, 2006.

[34] Candès, E. and Tao, T., "The Dantzig selector: Statistical estimation when p is much larger than n," *Annals of Statistics*, vol. 35, no. 6, pp. 2313–2351, 2007.

[35] Candès, E. J. and Tao, T., "Near-optimal signal recovery from random projections: Universal encoding strategies?," *IEEE Transactions on Information Theory*, vol. 52, no. 12, pp. 5406–5425, Dec. 2006.

[36] Candès, E. and Donoho, D., "New tight frames of curvelets and optimal representations of objects with piecewise C^2 singularities," *Communications on Pure and Applied Mathematics*, vol. 57, no. 2, pp. 219–266, 2004.

[37] Candès, E., Romberg, J., and Tao, T., "Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information," *IEEE Transactions on Information Theory*, vol. 52, no. 2, pp. 489–509, Feb. 2006.

[38] Candès, E. and Tao, T., "Decoding by linear programming," *IEEE Transactions on Information Theory*, vol. 51, no. 12, pp. 4203–4215, Dec. 2005.

[39] Candès, E., "The restricted isometry property and its implications for compressed sensing," *Comptes Rendus Mathematique*, vol. 346, pp. 589–592, May 2008.

[40] Candès, E., Eldar, Y., and Needell, D., "Compressed sensing with coherent and redundant dictionaries," *Applied and Computational Harmonic Analysis*, vol. 31, no. 1, pp. 59–73, 2011.

[41] Candès, E. and Romberg, J., "ℓ_1-magic: Recovery of sparse signals via convex programming." http://users.ece.gatech.edu/∼justin/l1magic/.

[42] Candès, E. J., Wakin, M. B., and Boyd, S. P., "Enhancing sparsity by reweighted ℓ_1 minimization," *Journal of Fourier Analysis and Applications*, vol. 14, no. 5-6, pp. 877–905, 2008.

[43] Carmi, A., Gurfil, P., and Kanevsky, D., "Methods for sparse signal recovery using Kalman filtering pseudo-measuremennt norms and quasi-norms," *IEEE Transactions on Signal Processing*, vol. 58, pp. 2405–2409, Apr. 2010.

[44] Charles, A., Asif, M. S., Romberg, J., and Rozell, C., "Sparsity penalties in dynamical system estimation," in *Proc. Conference on Information and System Sciences (CISS)*, pp. 1–6, Mar. 2011.

[45] Charles, A. S. and Rozell, C. J., "Re-weighted ℓ_1 dynamic filtering for time-varying sparse signal estimation," [Preprint] http://arxiv.org/abs/1208.0325.

[46] Chen, S. S., Donoho, D. L., and Saunders, M. A., "Atomic decomposition by basis pursuit," *SIAM Journal on Scientific Computing*, vol. 20, no. 1, pp. 33–61, 1999.

[47] Chen, W., Rodrigues, M., and Wassell, I., "Penalized ℓ_1 minimization for reconstruction of time-varying sparse signals," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 3988–3991, May 2011.

[48] Cohen, A., Daubechies, I., and Feauveau, J.-C., "Biorthogonal bases of compactly supported wavelets," *Communications on Pure and Applied Mathematics*, vol. 45, no. 5, pp. 485–560, 1992.

[49] Coifman, R., Geshwind, F., and Meyer, Y., "Noiselets," *Applied and Computational Harmonic Analysis*, vol. 10, no. 1, pp. 27–44, 2001.

[50] Coifman, R. and Wickerhauser, M., "Entropy-based algorithms for best basis selection," *IEEE Transactions on Information Theory*, vol. 38, no. 2 Part 2, pp. 713–718, 1992.

[51] Cotter, S. F. and Rao, B. D., "Sparse channel estimation via matching pursuit with application to equalization," *IEEE Transactions on Communications*, vol. 50, no. 3, pp. 374–377, 2002.

[52] Daubechies, I., Defrise, M., and De Mol, C., "An iterative thresholding algorithm for linear inverse problems with a sparsity constraint," *Communications on Pure and Applied Mathematics*, vol. 57, no. 11, pp. 1413–1457, 2004.

[53] Daubechies, I., *Ten Lectures on Wavelets*, vol. 61. Society for Industrial and Applied Mathematics (SIAM), 1992.

[54] Donoho, D. L. and Tsaig, Y., "Fast solution of ℓ_1-norm minimization problems when the solution may be sparse," *IEEE Transactions on Information Theory*, vol. 54, no. 11, pp. 4789–4812, 2008.

[55] Donoho, D., "Compressed sensing," *IEEE Transactions on Information Theory*, vol. 52, pp. 1289–1306, Apr. 2006.

[56] Donoho, D., Elad, M., and Temlyakov, V., "Stable recovery of sparse overcomplete representations in the presence of noise," *IEEE Transactions on Information Theory*, vol. 52, pp. 6–18, Jan. 2006.

[57] Donoho, D. and Huo, X., "Uncertainty principles and ideal atomic decomposition," *IEEE Transactions on Information Theory*, vol. 47, pp. 2845–2862, Nov. 2001.

[58] Donoho, D. and Johnstone, J., "Ideal spatial adaptation by wavelet shrinkage," *Biometrika*, vol. 81, no. 3, pp. 425–455, 1994.

[59] Efron, B., Hastie, T., Johnstone, I., and Tibshirani, R., "Least angle regression," *Annals of Statistics*, vol. 32, no. 2, pp. 407–499, 2004.

[60] Elad, M., Milanfar, P., and Rubinstein, R., "Analysis versus synthesis in signal priors," *Inverse Problems*, vol. 23, p. 947, 2007.

[61] Feng, P. and Bresler, Y., "Spectrum-blind minimum-rate sampling and reconstruction of multiband signals," in *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, vol. 3, pp. 1688–1691, s, 1996.

[62] Figueiredo, M., Nowak, R., and Wright, S., "Gradient projection for sparse reconstruction: Application to compressed sensing and other inverse problems," *IEEE Journal of Selected Topics in Signal Processing*, vol. 1, pp. 586–597, Dec. 2007.

[63] Friedlander, M., Mansour, H., Saab, R., and Yilmaz, O., "Recovering compressively sampled signals using partial support information," *IEEE Transactions on Information Theory*, vol. 58, pp. 1122 –1134, Feb. 2012.

[64] Fuchs, J., "On sparse representations in arbitrary redundant bases," *IEEE Transactions on Information Theory*, vol. 50, pp. 1341–1344, June 2004.

[65] Garrigues, P. J. and Ghaoui, L. E., "An homotopy algorithm for the Lasso with online observations," *Neural Information Processing Systems (NIPS) 21*, Dec. 2008.

[66] Golub, G. and Van Loan, C., *Matrix Computations*. Johns Hopkins University Press, 1996.

[67] Griswold, M. A., Jakob, P. M., Heidemann, R. M., Nittka, M., Jellus, V., Wang, J., Kiefer, B., and Haase, A., "Generalized autocalibrating partially parallel acquisitions (GRAPPA)," *Magnetic Resonance in Medicine*, vol. 47, pp. 1202–1210, June 2002.

[68] Hale, E., Yin, W., and Zhang, Y., "Fixed-Point Continuation for ℓ_1-minimization: Methodology and Convergence," *SIAM Journal on Optimization*, vol. 19, no. 3, pp. 1107–1130, 2008.

[69] Hamilton, L. H., Fabregat, J. A., Moratal, D., Ramamurthy, S., Lerakis, S., Parks, W. J., Sallee III, D., and Brummer, M. E., "PINOT: Time-resolved parallel magnetic resonance imaging with a reduced dynamic field of view," *Magnetic Resonance in Medicine*, vol. 65, pp. 1062–1074, Apr. 2011.

[70] Hayes, M., *Statistical Digital Signal Processing and Modeling*. John Wiley & Sons, Inc. New York, NY, USA, 1996.

[71] Horn, B. K. and Schunck, B. G., "Determining optical flow," *Artificial Intelligence*, vol. 17, pp. 185–203, Aug. 1981.

[72] Jain, J. and Jain, A., "Displacement measurement and its application in interframe image coding," *IEEE Transactions on Communications*, vol. 29, pp. 1799– 1808, Dec. 1981.

[73] James, G., Radchenko, P., and Lv, J., "The DASSO algorithm for fitting the Dantzig selector and the Lasso," *Journal of the Royal Statistical Society, Series B*, vol. 71, pp. 127–142, 2009.

[74] Jung, H., Sung, K., Nayak, K., Kim, E., and Ye, J., "k-t FOCUSS: A general compressed sensing framework for high resolution dynamic MRI," *Magnetic Resonance in Medicine*, vol. 61, no. 1, pp. 103–116, 2009.

[75] Jung, H. and Ye, J., "Motion estimated and compensated compressed sensing dynamic magnetic resonance imaging: What we can learn from video compression techniques," *International Journal of Imaging Systems and Technology*, vol. 20, no. 2, pp. 81–98, 2010.

[76] Jung, H., Sung, K., Nayak, K. S., Kim, E. Y., and Ye, J. C., "k-t FOCUSS: a general compressed sensing framework for high resolution dynamic MRI," *Magnetic Resonance in Medicine*, vol. 61, pp. 103–116, Jan. 2009.

[77] Kailath, T., Sayed, A. H., and Hassibi, B., *Linear estimation*. Prentice Hall Upper Saddle River, NJ, 2000.

[78] Kalman, R., "A new approach to linear filtering and prediction problems," *Journal of basic Engineering*, vol. 82, no. 1, pp. 35–45, 1960.

[79] Kay, S. M., *Fundamentals of statistical signal processing: Estimation theory*. Prentice Hall, 1993.

[80] Kellman, P., Epstein, F. H., and McVeigh, E. R., "Adaptive sensitivity encoding incorporating temporal filtering (TSENSE)," *Magnetic Resonance in Medicine*, vol. 45, pp. 846–852, May 2001.

[81] Khajehnejad, M., Xu, W., Avestimehr, A., and Hassibi, B., "Improved sparse recovery thresholds with two-step reweighted ℓ_1 minimization," in *IEEE International Symposium on Information Theory Proceedings (ISIT)*, pp. 1603–1607, June 2010.

[82] Kyriakos, W. E., Panych, L. P., Kacher, D. F., Westin, C., Bao, S. M., Mulkern, R. V., and Jolesz, F. A., "Sensitivity profiles from an array of coils for encoding and reconstruction in parallel (SPACE RIP)," *Magnetic Resonance in Medicine*, vol. 44, pp. 301–308, Aug. 2000.

[83] Li, W. and Preisig, J., "Estimation of rapidly time-varying sparse channels," *IEEE Journal of Oceanic Engineering*, vol. 32, no. 4, pp. 927–939, 2007.

[84] Liang, D., Liu, B., Wang, J., and Ying, L., "Accelerating SENSE using compressed sensing," *Magnetic Resonance in Medicine*, vol. 62, pp. 1574–1584, Dec. 2009.

[85] Liu, B., Sebert, F., Zou, Y., and Ying, L., "SparseSENSE: Randomly-sampled parallel imaging using compressed sensing," in *Proceedings of the 16th Annual Meeting of ISMRM, Toronto*, p. 3154, May 2008.

[86] Liu, C., *Beyond Pixels: Exploring New Representations and Applications for Motion Analysis*. Doctor of philosophy, Massachusetts Institute of Technology, 2009.

[87] Lu, Y. and Do, M., "Sampling signals from a union of subspaces [a new perspective for the extension of this theory]," *IEEE Signal Processing Magazine*, vol. 25, pp. 41–47, Mar. 2008.

[88] Lustig, M., Donoho, D., and Pauly, J., "Sparse MRI: The application of compressed sensing for rapid MR imaging," *Magnetic Resonance in Medicine*, vol. 58, no. 6, pp. 1182–1195, 2007.

[89] Lustig, M., Donoho, D., Santos, J., and Pauly, J., "Compressed Sensing MRI [A look at how CS can improve on current imaging techniques]," *IEEE Signal Processing Magazine*, vol. 25, pp. 72–82, Mar. 2008.

[90] Lustig, M., Santos, J., Donoho, D., and Pauly, J., "k-t SPARSE: High frame rate dynamic MRI exploiting spatio-temporal sparsity," in *Proceedings of the 13th Annual Meeting of ISMRM, Seattle*, p. 2420, May 2006.

[91] Madore, B., "UNFOLD-SENSE: A parallel MRI method with self-calibration and artifact suppression," *Magnetic Resonance in Medicine*, vol. 52, pp. 310–320, Aug. 2004.

[92] Magarey, J. and Kingsbury, N., "Motion estimation using a complex-valued wavelet transform," *IEEE Transactions on Signal Processing*, vol. 46, pp. 1069–1084, Apr. 1998.

[93] Malioutov, D., Sanghavi, S., and Willsky, A., "Compressed sensing with sequential observations," *IEEE International Conference on Acoustics, Speech, and Signal Processing,*, pp. 3357–3360, April 2008.

[94] Mallat, S., *A Wavelet Tour of Signal Processing*. Academic Press, second ed., 1999.

[95] Malvar, H. and Staelin, D., "The LOT: Transform coding without blocking effects," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 37, no. 4, pp. 553–559, 1989.

[96] Mansour, H., "Beyond ℓ_1-norm minimization for sparse signal recovery," in *Proc. of the IEEE Statistical Signal Processing Workshop (SSP)*, Aug. 2012.

[97] Marcia, R. and Willett, R., "Compressive coded aperture video reconstruction," in *Proc. European Signal Processing Conf.(EUSIPCO)*, Aug. 2008.

[98] Meinshausen, N. and Yu, B., "Lasso-type recovery of sparse representations for high-dimensional data," *Annals of Statistics*, vol. 37, no. 1, pp. 246–270, 2008.

[99] Mun, S. and Fowler, J. E., "Residual reconstruction for block-based compressed sensing of video," in *Data Compression Conference (DCC)*, pp. 183–192, Mar. 2011.

[100] Natarajan, B., "Sparse Approximate Solutions to Linear Systems," *SIAM Journal on Computing*, vol. 24, p. 227, 1995.

[101] Needell, D. and Tropp, J., "CoSaMP: Iterative signal recovery from incomplete and inaccurate samples.," *Applied and Computational Harmonic Analysis*, vol. 26, pp. 301–321, June 2008.

[102] Nesterov, Y. and Nemirovsky, A., "Interior Point Polynomial Methods in Convex Programming," *Studies in Applied Mathematics (SIAM)*, vol. 13, 1994.

[103] Nyquist, H., "Certain topics in telegraph transmission theory," *Transactions of the American Institute of Electrical Engineers*, vol. 47, no. 2, pp. 617–644, 1928.

[104] Olshausen, B. and Field, D., "Sparse coding with an overcomplete basis set: A strategy employed by V1?," *Vision research*, vol. 37, no. 23, pp. 3311–3325, 1997.

[105] Orchard, M. T. and Sullivan, G. J., "Overlapped block motion compensation: An estimation-theoretic approach," *IEEE Transactions on Image Processing*, vol. 3, pp. 693–699, Sept. 1994.

[106] Osborne, M., Presnell, B., and Turlach, B., "A new approach to variable selection in least squares problems," *IMA Journal of Numerical Analysis*, vol. 20, no. 3, pp. 389–403, 2000.

[107] Otazo, R., Kim, D., Axel, L., and Sodickson, D. K., "Combination of compressed sensing and parallel imaging for highly accelerated first-pass cardiac perfusion MRI," *Magnetic Resonance in Medicine*, vol. 64, pp. 767–776, Sept. 2010.

[108] Park, J. and Wakin, M., "A multiscale framework for compressive sensing of video," in *IEEE Picture Coding Symposium, Chicago*, pp. 1–4, May 2009.

[109] Pati, Y. C., Rezaiifar, R., and Krishnaprasad, P. S., "Orthogonal matching pursuit: recursive function approximation with applications to wavelet decomposition," in *27th Asilomar Conference on Signals, Systems and Computers*, vol. 1, pp. 40–44, Nov. 1993.

[110] Portniaguine, O., Bonifasi, C., DiBella, E., and Whitaker, R., "Inverse methods for reduced k-space acquisition," in *Proceedings of the 11th Meeting of ISMRM, Toronto*, p. 481, July 2003.

[111] Pruessmann, K., Weiger, M., Scheidegger, M., and Boesiger, P., "SENSE: sensitivity encoding for fast MRI," *Magnetic Resonance in Medicine*, vol. 42, no. 5, p. 952–962, 1999.

[112] Puri, R., Majumdar, A., and Ramchandran, K., "PRISM: a video coding paradigm with motion estimation at the decoder," *IEEE Transactions on Image Processing*, vol. 16, pp. 2436–2448, Oct. 2007.

[113] Radchenko, P. and James, G., "Improved variable selection with forward-lasso adaptive shrinkage," *The Annals of Applied Statistics*, vol. 5, no. 1, pp. 427–448, 2011.

[114] Reddy, D., Veeraraghavan, A., and Chellappa, R., "P2C2: programmable pixel compressive camera for high speed imaging," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Colorado Springs*, pp. 329–336, June 2011.

[115] Richardson, I. E. G., *H.264 and MPEG-4 Video Compression*. Chichester, UK: John Wiley & Sons, Ltd, Sept. 2003.

[116] Rockafellar, R. T., *Convex analysis*, vol. 28. Princeton university press, 1997.

[117] Romberg, J., "Imaging via Compressive Sampling [Introduction to compressive sampling and recovery via convex programming]," *IEEE Signal Processing Magazine*, vol. 25, pp. 14–20, Mar. 2008.

[118] Rudelson, M. and Vershynin, R., "Geometric approach to error correcting codes and reconstruction of signals," *International Mathematics Research Notices*, no. 64, pp. 4019–4041, 2005.

[119] Rudelson, M. and Vershynin, R., "The Littlewood–Offord problem and invertibility of random matrices," *Advances in Mathematics*, 2008.

[120] Rudin, L. I., Osher, S., and Fatemi, E., "Nonlinear total variation based noise removal algorithms," *Physica D: Nonlinear Phenomena*, vol. 60, pp. 259–268, Nov. 1992.

[121] Santosa, F. and Symes, W., "Linear Inversion of Band-Limited Reflection Seismograms," *SIAM Journal on Scientific and Statistical Computing*, vol. 7, p. 1307, 1986.

[122] Selesnick, I., Baraniuk, R., and Kingsbury, N., "The dual-tree complex wavelet transform," *IEEE Signal Processing Magazine*, vol. 22, no. 6, pp. 123–151, 2005.

[123] Shannon, C. E., "Communication in the presence of noise," *Proceedings of the IRE*, vol. 37, no. 1, pp. 10–21, 1949.

[124] Shannon, C. E., "A mathematical theory of communication," *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 5, no. 1, pp. 3–55, 2001.

[125] Sodickson, D. K. and Manning, W. J., "Simultaneous acquisition of spatial harmonics (SMASH): Fast imaging with radio-frequency coil arrays," *Magnetic Resonance in Medicine*, vol. 38, pp. 591–603, Oct. 1997.

[126] Sorenson, H. W., "Least-squares estimation: from Gauss to Kalman," *IEEE Spectrum*, vol. 7, no. 7, pp. 63–68, 1970.

[127] Srinivasan, R. and Rao, K., "Predictive coding based on efficient motion estimation," *IEEE Transactions on Communications*, vol. 33, pp. 888– 896, Aug. 1985.

[128] Sullivan, G. J. and Wiegand, T., "Video compression - from concepts to the H.264/AVC standard," *Proceedings of the IEEE*, vol. 93, pp. 18–31, Jan. 2005.

[129] Takhar, D., Laska, J., Wakin, M., Duarte, M., Baron, D., Sarvotham, S., Kelly, K., and Baraniuk, R., "A New Compressive Imaging Camera Architecture using Optical-Domain Compression," *Proc. IS&T/SPIE Computational Imaging IV*, 2006.

[130] Taubman, D. and Marcellin, M., *JPEG2000: Image compression fundamentals, standards and practice,*. Kluwer Academic Publishers Boston, 2002.

[131] Tibshirani, R., "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society, Series B*, vol. 58, no. 1, pp. 267–288, 1996.

[132] Tropp, J., "Just relax: Convex programming methods for identifying sparse signals in noise," *IEEE Transactions on Information Theory*, vol. 52, no. 3, pp. 1030–1051, 2006.

[133] Tropp, J. A., Laska, J. N., Duarte, M. F., Romberg, J. K., and Baraniuk, R. G., "Beyond Nyquist: Efficient sampling of sparse, bandlimited signals," *IEEE Transactions on Information Theory*, vol. 56, pp. 520–544, Jan. 2010.

[134] Tropp, J. A. and Gilbert, A., "Signal recovery from random measurements via orthogonal matching pursuit," *IEEE Transactions on Information Theory*, vol. 53, pp. 4655–4666, Dec. 2007.

[135] Tsao, J., Boesiger, P., and Pruessmann, K. P., "k-t BLAST and k-t SENSE: dynamic MRI with high frame rate exploiting spatiotemporal correlations," *Magnetic Resonance in Medicine*, vol. 50, pp. 1031–1042, Nov. 2003.

[136] Unser, M., "Sampling-50 years after shannon," *Proceedings of the IEEE*, vol. 88, pp. 569–587, Apr. 2000.

[137] van den Berg, E. and Friedlander, M. P., "Probing the pareto frontier for basis pursuit solutions," *SIAM Journal on Scientific Computing*, vol. 31, no. 2, pp. 890–912, 2008.

[138] Vanderbei, R., *Linear Programming: Foundations and Extensions*. Kluwer Academic Publishers, 2001.

[139] Vaswani, N., "Kalman filtered compressed sensing," in *15th IEEE International Conference on Image Processing*, pp. 893–896, Oct. 2008.

[140] Velikina, J., Johnson, K., Block, K., and Samsonov, A., "Design of temporally constrained compressed sensing methods for accelerated dynamic mri," in *Proceedings of the 18th Annual Meeting of ISMRM, Stockholm*, p. 4865, May 2010.

[141] Vetterli, M. and Kovacevic, J., *Wavelets and subband coding*. Prentice Hall PTR Englewood Cliffs, NJ, 1995.

[142] Vetterli, M., Marziliano, P., and Blu, T., "Sampling signals with finite rate of innovation," *Signal Processing, IEEE Transactions on*, vol. 50, no. 6, pp. 1417–1428, 2002.

[143] Wakin, M., Laska, J., Duarte, M., Baron, D., Sarvotham, S., Takhar, D., Kelly, K., and Baraniuk, R., "An architecture for compressive imaging," pp. 1273–1276, 8-11 Oct. 2006.

[144] Wen, Z. and Yin, W., "FPC_AS: A MATLAB solver for ℓ_1-regularized least-squares problems." http://www.caam.rice.edu/∼optimization/L1/FPC_AS/.

[145] Whitaker, E., "On the functions which are represented by the expansion of interpolating theory," in *Proc. Roy. Soc. Edinburgh*, vol. 35, pp. 181–194, 1915.

[146] Wiegand, T., Sullivan, G. J., Bjontegaard, G., and Luthra, A., "Overview of the H.264/AVC video coding standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, pp. 560–576, July 2003.

[147] Wright, S., Nowak, R., and Figueiredo, M., "Sparse reconstruction by separable approximation," *IEEE Transactions on Signal Processing*, vol. 57, pp. 2479–2493, July 2009.

[148] Yang, J. and Zhang, Y., "Alternating direction algorithms for ℓ_1-problems in compressive sensing," *SIAM Journal on Scientific Computing*, vol. 33, no. 1-2, pp. 250–278, 2011.

[149] Yang, T.-J., Tsai, Y.-M., Li, C.-T., and Chen, L.-G., "WarmL1: A warm-start homotopy-based reconstruction algorithm for sparse signals," in *IEEE International Symposium on Information Theory Proceedings (ISIT)*, pp. 2226–2230, July 2012.

[150] Yin, W., Osher, S., Goldfarb, D., and Darbon, J., "Bregman iterative algorithms for ℓ_1 minimization with application to compressed sensing," *SIAM Journal on Imaging sciences*, vol. 1, no. 1, pp. 143–168, 2008.

[151] Ziniel, J., Potter, L. C., and Schniter, P., "Tracking and smoothing of time-varying sparse signals via approximate belief propagation," in *44th Asilomar Conference on Signals, Systems and Computers*, pp. 808–812, Nov. 2010.

[152] Zou, H., "The adaptive lasso and its oracle properties," *Journal of the American Statistical Association*, vol. 101, no. 476, pp. 1418–1429, 2006.